THE 18TH EUROPEAN CONFERENCE ON MACHINE LEARNING
AND
THE 11TH EUROPEAN CONFERENCE ON PRINCIPLES AND PRACTICE
OF KNOWLEDGE DISCOVERY IN DATABASES

PROCEEDINGS OF THE

ECML/PKDD'2007

DISCOVERY CHALLENGE

September 17, 2007

Warsaw, Poland

Editors:
Hung Son Nguyen
Institute of Mathematics, Warsaw University

Typesetting:
Hung Son Nguyen

Preface

Knowledge discovery in real-world databases requires a broad scope of techniques and forms of knowledge. Both the knowledge and the applied methods should fit the discovery tasks and should adapt to knowledge hidden in the data. The Discovery Challenge will encourage a collaborative research effort, a broad and unified view of knowledge and methods of discovery, and emphasis on business problems and solutions to those problems.

The idea of Discovery Challenge came from Jan Żytkow, who suggested to organize such an event during PKDD'99 in Prague. The Discovery Challenge constitutes a collection of data and problems as a common ground for better comparisons and discussions of the applicability of KDD methods on a real-world problems with respect to both KDD and application viewpoints. The main goals of the Discovery Challenge are

- stimulate an open view of knowledge and discovery
- stimulate collaborative approach to KDD and research on unification of both different forms of knowledge and discovery
- integrate into KDD an emphasis on real-world problems and solutions to those problems

This year's **Discovery Challenge** was devoted to three problems: user behaviour ptrediction from web traffic logs, HTTP traffic classification, and Sumerian literature understanding. The Challenge was co-organized by Piotr Ejdys(Gemius SA), Hung Son Nguyen (Warsaw University), Pascal Poncelet (EMA-LGI2P) and Jerzy Tyszkiewicz (Warsaw University).

1. **User's behaviour prediction**
   This task is co-organized by Gemius, the leading Internet market research company in Poland. The problem objective is to predict user behaviour by characterising nature of user's visit, i.e., the list categories of the visited Internet portal and the number of page views in each category. The challenge is accomplished with use of web traffic data from Polish web sites employing gemiusTraffic study, grouped by appropriate categories.
2. **HTML traffic prediction**
   The task is co-organized by the LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, FRANCE) and the LGI2P (Ecole des Mines d'Alčs, FRANCE) and is based on a dataset of real-world web traffic in conjunction with Bee Ware, a leading provider of secure web enabled delivery solutions. The aim of this contest is to classify HTTP traffic into multiple classes, distinguishing between different attack types but also between anomalous and normal traffic. The procedure involves 3 tasks respectively handling classification, pattern isolation and performance issues.
3. **Sumerian Literature understanding**
   The challenge is related to a database of ca. 28'000 administrative documents from the kingdom of the III Dynasty of Ur, which existed in the 21st century b.C. in

Mesopotamia, present day southern Iraq. The documents were originally written in Sumerian on clay tablets, using cuneiform script.
The challenge is to discover useful information about water and spirit transport system using available transliterated documents in the text format.

There were 122 participants registering to the Discovery Challenge website. However, only the first challenge was attacked by more than five participants. The final classification for the first task is as follows:

The winner:

- "Auto-regressive and Score maximizing Approach" by Krzysztof Dembczynski and Wojciech Kotlowski from Poznan University of Technology, Poland and Marcin Sydow from Polish-Japanese Institute of Information Technology, Poland

Runners-up:

- "Bayesian Inference Approach" by Malik Tahir Hassan, Khurum Nazir Junejo and Asim Karim from Lahore University, Pakistan
- "Frequent Item Approach" by Tung-Ying Lee from National Tsing Hua University, Taiwan

I would like to express my gradtitute to Dr Petr Berka, Dr Steffen Bickel and Dr Bruno Cremilleux – the chairs of previous events of PKDD/ECML Discovery Challenge – for their help. I also indebted to Marcin Szczuka – the ECML/PKDD Local Chair – for his assistance on setting up the Discovery Challenge website. Last but not least I would like thank our sponsors for their great contributions for the success of ECML/PKDD Discovery Challenge.

Warsaw, August 2007 Hung Son Nguyen

ECML/PKDD'2007 Discovery Challenge Organization

Program Committee

Petr Berka
Steffen Bickel
Bruno Cremilleux
Piot Ejdys (co-chair)
Joanna Jaworska
Wojciech Jaworski
Hung Son Nguyen (chair)
Pascal Poncelet (co-chair)
Maguelonne Teisseire
Jerzy Tyszkiewicz (co-chair)

Table of Contents

Users' Behaviour Challenge

Joanna Jaworska[1], Hung Son Nguyen[2]

[1] Gemius SA,
Mars Building, Staircase D, 2nd Floor
Wołoska 7 Str., Warsaw 02-675, Poland
[2] Institute of Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland

Abstract. This task is co-organized by Gemius SA, the leading Internet market research agency in Central and Eastern Europe. The problem objective is to predict user behaviour by characterising nature of user's visit, i.e., the list categories of the visited Internet portal and the number of page views in each category. The challenge is accomplished with use of web trafic data from Polish web sites employing *gemiusTraffic* study, grouped by appropriate categories. This will be accomplished with use of web traffic data from Polish web sites employing *gemiusTraffic* study, grouped by appropriate categories. The above defined objective has been divided into three separate challenge problems: Problem 1 is related to the length of the visit. A visit – accordingly to the definition – is a sequence of page views by one user (cookie). As web pages are identified by their categories, during one visit user may view pages of one or more categories. The problem is to predict whether a given visit is short (1 category) or long (two or more categories). Problem 2 is related to the most probable categories. Solution of Problem 2 is a list of the most probable categories in a given visit of a given user. Problem 3 is to predict the most probable categories and ranges of numbers of page views. Solution of Problem 3 is a list of the most probable categories in a given visit of a given user with range of number of page views in each category.

1. Introduction

Gemius is the Internet market research company with a mission of providing information about, inter alia, Internet users' behaviour and their social and demographic profile. Knowledge about the Internet market, accumulated as processed statistical data, is a basis for building interactive marketing strategies and pointing to the content most desired by users. It helps in adjusting an offer to needs of given target groups and achieving increased profits from various web-related business activities.

Behavioural information in the gemiusTraffic study is acquired through use of scripts, placed in code of the monitored web page. The scripts report to the gemiusTraffic platform each Page View. Registered Page Views are the basis for calculating the further usage statistics like numbers of visits and visitors. Internet users are identified

by the cookies technology that enables merging Page Views into Visits while fully respecting users' privacy.
It is possible to define several other, more precise metrics from the basic indexes (e.g. sequence of Page Views assigned to one Visit can define Visit Path). Such metrics may deliver more accurate depiction of web traffic nature for a given site. Additionally gemiusTraffic research records technical data like versions of used web browsers and operating systems.

2. Problem specification

2.1. Useful definitions

- Page – web page participating in the research (with embedded gemiusTraffic script); pages are distinguishable only by the category they belong to,
- Page View – event of displaying the monitored web page,
- Visit - an uninterrupted series of Page Views on a given web site executed by the same Visitor (cookie), counted as a closed whole. This represents an Internet user's total "stay" on the web site in question for any individual visit. It is assumed that one Page View cannot exceed 30 minutes (a longer Page View duration / gap will result in the series being counted as two separate Visits),
- Visit Path - series of web pages visited during one Visit. This represents the click-stream that the user followed in navigating a web site.
- Category – each page is qualified as a member of a relevant category that is a group of web sites of a similar leading theme., e.g. entertainment, technology, news, communication, education, e-commerce, business, etc. These categories have been assigned certain identifiers.

2.2. General specification

The problem objective is to predict user behaviour by characterising nature of user's visit. The visit is defined by categories of visited web pages and number of page views in each category.

This will be accomplished with use of web traffic data from Polish web sites employing gemiusTraffic study, grouped by appropriate categories. The above defined objective has been divided into three separate problems:

Problem 1: Length of the visit

A visit – accordingly to the definition – is a sequence of page views by one user (cookie). As web pages are identified by their categories, during one visit user may view pages of one or more categories. Therefore we define:

- *short visit* – is a visit with page views of only one category,
- *long visit* – is a visit of with views of pages belonging to at least two categories

Solution of Problem 1 is answer on a question whether a given visit is short or long.

Problem 2: The most probable categories

Solution of Problem 2 is a list of the most probable categories in a given visit of a given user.

Problem 3: The most probable categories and ranges of numbers of page views

Solution of Problem 3 is a list of the most probable categories in a given visit of a given user with range of number of page views in each category.

3. Data format

Data processed in this problem will consist of two main parts with information about 1) users and 2) visit paths.

Both parts will be presented as two separate text files. The exact format of the files is as follows:

i) Users table:
This table consists of the following fields: *user_id, country_id, region_id, city_id, system_id, system_sub_id, browser_id, browser_ver_id*;

The meaning of fields follows from their names. An example record in Users table is as follows:

user_id	***country_id***	***region_id***	***city_id***	***system_id***	***system_sub_id***	***browser_id***	***browser_ver_id***
…	…	…	…	…	…	…	…
10	42	11	44	3	9	1	517

ii) Visit Paths table:
This table consists of the following fields:
path_id, user_id, timestamp, path{*category_id, pageviews_number*} ->{} -> ...

An example record of this table is as follows:

path_id	***user_id***	***timestamp***	***Path (category_id, pageviews_number), …***			
27	1	1169814548	7,1	16,2	17,9	16,1

During one visit there may be found Page Views of pages of different categories. In the above example there are four Categories and 13 Page Views in the visit, therefore it is an example of a long visit. Accordingly to the definition of the Visit Path the sequence of visited categories is important as well as the fact of repeated views of pages of the same category. The above sequence 7 → 16 → 17 → 16 and a new sequence 16 → 7 → 17 →16 are significantly different because of changed order of visited categories.

Explanation of fields in data files:

user_id	web user identifier (based on user's cookie),
country_id, region_id, city_id	these three numbers are based on geo-localisation data derived from user's IP address,
system_id, system_sub_id	identifier of user's operation system and version,
browser_id, browser_ver_id	identifier of user's web browser and version,
path_id	identifier of a given Visit Path,
timestamp of the Visit commencement	defines time of the start of the Visit, that is time of the first Page View in the Visit,
category_id	identifier of category of the web page visited by the user,
pageviews_number	number of Page Views during one Visit in one category (not interrupted by a Page View from a different category).

4. Training data and test data

The problem will be solved using data collected during one month of monitoring of Polish web sites. The file with Visit Paths table (see Section 3) is split into two separate files – one with practice data and second with test data.

- **Training data file** contains data gathered during first three weeks of the month
- **Test data file** contains data from one last week. Please note that information about Visit Paths has been removed from this file.

5. Input and output data specification

User's behaviour predicting will mean presenting user's future Visit Path. The Visit Path consist of categories (visited web pages are included in these categories) and number of Page Views per Visit.

Input data will be *user_id* and *timestamp* of the first Page View. With this information one should determine Visit Path giving: *category_id* and *pageviews_*number accordingly to the following formula:

$$f_{predicted}\begin{pmatrix} user\_id \\ timestamp \end{pmatrix} = \begin{pmatrix} category\_id_1, pageviews\_number_1 \\ category\_id_2, pageviews\_number_2 \\ \dots \\ category\_id_m, pageviews\_number_m \end{pmatrix} \quad (1)$$

where: *m* stands for number of different Categories in one Visit.

Entrant's task is to present text files containing solutions of individual problems:

Gemius1.txt – for problem 1,
Gemius2.txt – for problem 2,
Gemius3.txt – for problem 3,

Results for each problem should be presented in the same sequence as in the test file. The general structure of the result file is:

path_id user_id timestamp result

where *result* form depends on the problem – details are specified below in descriptions of problems.

6. Specification of problems

Problem 1:

Solution of this problem is determining length of the Visit – whether it is long or short visit. Accordingly to the definition:

$$Length = \begin{cases} short \Leftrightarrow m = 1 \\ long \Leftrightarrow m > 1 \end{cases} \quad (2)$$

Therefore one needs to estimate a value of the parameter *m* and present the length accordingly. The correct solution is:

$$Length_{predicted} = Length_{actual} \quad (3)$$

Please output to file 'Gemius1.txt' as a result value "1" or ">1" as predicted lengths of visits for all individual users (cookies).

Problem 2:

Solution of this problem is a predicted vector of the most probable categories for a given user.
Solution is three identifiers of the most probable categories for a given user that appear in the first three places of the visit path:

$$Category\_id_{predicted} = \begin{pmatrix} category\_id_{predicted\_1} \\ \dots \\ category\_id_{predicted\_3} \end{pmatrix} \quad (4)$$

Please output to file 'Gemius2.txt' as a result 3 predicted category identifiers separated by tab (\t) character.
Estimated vector will be compared with the actual vector of category identifiers, that is individual values of elements are compared:

$$Category\_id_{predicted}(i) = Category\_id_{actual}(j) \quad (5)$$

Two score vectors are created to analyse correctness of the prediction – one for the actual vector and one for the predicted category vector, accordingly to the following rule: 5 points for the first category, 4 points for second category, 3 for third, etc. (from fifth category on one point is given). Afterwards, minimums of corresponding elements of both vectors are determined. These numbers are summed up and give the final result score. It is illustrated in the following example:

THE ACTUAL CATEGORY VECTOR: R = [2 4 7 1 3 5 9] AND THE PREDICTED VECTOR: P = [1 4 5].

THE SCORE VECTOR **SCR(R)** FOR **R** AND SCORE VECTOR **SCR(P)** FOR **P** ARE:

Categories	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**…**
Scr(R)	2	5	1	4	1	0	3	0	1	0	0	0	0	0	0	…
Scr(P)	5	0	0	4	3	0	0	0	0	0	0	0	0	0	0	…

MINIMUMS FROM BOTH VECTORS **SCR(R)** AND **SCR(P)** IS AS FOLLOWS:

Categories	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**…**
Min	2	0	0	4	1	0	0	0	0	0	0	0	0	0	0	…

FINAL SCORE DETERMINED BY SUMMING UP ELEMENTS OF VECTOR MIN, I.E.
SCORE(P) = 7

Winner of problem 2: $\max\left(\sum_{predicted\ vector\ P} Score(P)\right)$

Problem 3:

Solution to this problem are 3 the most probable categories for a given user and range of number of visits in each category.

$$f_{predicted}\begin{pmatrix} user\_id \\ timestamp \end{pmatrix} = \begin{pmatrix} category\_id_1, range\_of\_pageviews_1 \\ \dots \\ category\_id_3, range\_of\_pageviews_3 \end{pmatrix} \quad (6)$$

Possible ranges and their assigned identifiers:

$$range\_of\_pageviews = \begin{pmatrix} 1 \\ 2-3 \\ 4-\infty \end{pmatrix} \equiv \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad (7)$$

Please output to file 'Gemius3.txt' as a result 3 predicted pairs of numbers: category identifier and range identifier (as in (7)) separated by comma. Pairs need to be separated by tab (\t) character.
Estimated vector is then compared with the actual vector defining visit path (1). The values of given categories are checked accordingly to equation (5), ranges of numbers of page views are checked:

$$number\_of\_pageviews_{actual\_i} \in range\_of\_pageviews_i \quad (8)$$

Two score vectors are created to analyse correctness of the prediction – one for the actual vector and one for the predicted category and range vector, accordingly to the following rule: 5 points for the first category, 4 points for second category, 3 for third, etc. (from fifth category on one point is given). In case of the actual vector to each value is added 1 point for range of page views. In case of the predicted vector 1 point is added for giving a correct range or 0 points for wrong range of page views. Afterwards, minimums of corresponding elements of both vectors are determined. These numbers are summed up and give the final result score. It is illustrated in the following example:

THE ACTUAL CATEGORY VECTOR: R = [2,1 4,3 7,3 1,2 3,3 5,3 9,2] AND
THE PREDICTED VECTOR: P = [1,2 4,2 5,3].

THE SCORE VECTOR **SCR(R)** FOR **R** AND SCORE VECTOR **SCR(P)** FOR **P** ARE:

Categories	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**…**
Scr(R)	2	5	1	4	1	0	3	0	1	0	0	0	0	0	0	…
Scr(P)	5	0	0	4	3	0	0	0	0	0	0	0	0	0	0	…

MINIMUMS FROM BOTH VECTORS **SCR(R)** AND **SCR(P)** IS AS FOLLOWS:

Categories	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**…**
Award	2+1	0	0	4+0	1+1	0	0	0	0	0	0	0	0	0	0	…

FINAL SCORE DETERMINED BY SUMMING UP ELEMENTS OF VECTOR MIN, I.E.
SCORE(P) = 9

Winner of problem 3: $\max\left(\sum_{predicted\ vector\ P} Score(P)\right)$

7. General remarks to problems:

1. The predicting algorithm may be iterative, where results for each iteration are subsequent elements of the predicted vector (problems 2 and 3) that are compared and verified with the actual vector. Every iteration ends with presenting a next element of the vector an the final result is the full vector (accordingly to a problem it is a vector of an appropriate type).

2. Goal of the above problems is to present a prognosis of behaviour of a given user, but it may be interesting to describe certain classes of users, defined by characteristics given in the user table in the appropriate text file (see paragraph 3). For instance a class may be defined as users that:
   - are from the same country, i.e. have the same country_id,
   - use the same browser, i.e. have the same browser_id and browser_ver_id,
   - use the same operating system, i.e. have the same system_id and system_sub_id.

   Results for such a class would mean describing behavioural patterns for whole groups of users.

Effective Prediction of Web User Behaviour with User-Level Models.

Krzysztof Dembczyński[1], Wojciech Kotłowski[1], Marcin Sydow[2]

[1] Institute of Computing Science, Poznań University of Technology, 60-965 Poznań, Poland
[2] Polish-Japanese Institute of Information Technology, 02-008 Warsaw, Poland

Abstract. The paper presents our solution to the ECML/PKDD 2007 Challenge Task that concerns prediction of Internet user behaviour by characterising the nature of their Web page visits. Our solution has low time and space complexity, scales well with large datasets and, at the same time, produces high-quality results. Comparison of the performance of our ultimate approach with a suit of other approaches that we examined exhibits its superiority and some hardness of the given problem.

1 Introduction

The contest objective [4] was to predict Web users' behaviour by characterising the nature of their visits. The visit is defined by categories of visited web pages and the number of page views in each category. The contest was organised into 3 tasks: predicting the number of Web page categories (1 or greater than 1), predicting the first 3 visited categories and predicting the number of pages seen in each of the first 3 categories.

The data was provided by Gemius – a leading Internet market research company in Poland – and divided into: training set (379485 records) and testing set (166299 records – twice less than in the training one). Each record contains `record number`, `user id`, `timestamp`. Additionally, in the training set the records contain the sequence of pairs, each of the form `category`, `#pages`. Each record corresponds to a user session started at timestamp and reflects the category and number of pages seen by that user in chronological order (from left to right). Example: `248 46 1167680792 12,1 8,7 12,3`.

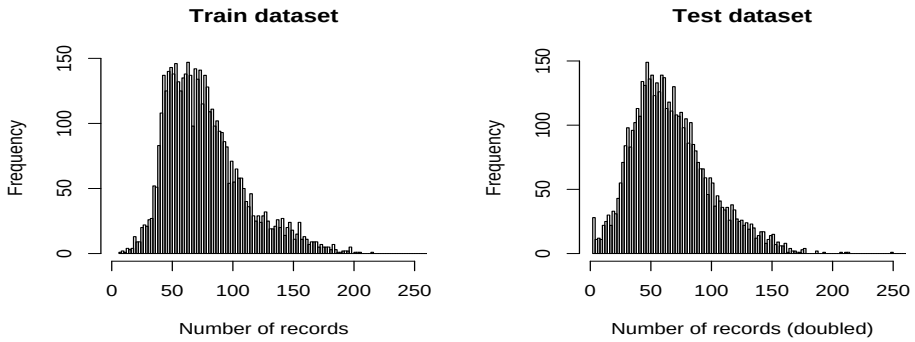
In addition, a third, auxiliary dataset was provided which contains 4882, user-related records consisting of the following attributes: `user id`, `country`, `region`, `city`, `system`, `sysVer`, `browser`, `browserVer`.

Our exploratory analysis phase is described in section 2. Section 3 describes our first attempts of building *global models*. During our work, we have observed that models constructed separately for each user (*user-models*) give better results than the global ones. The reason of this is a specific nature of data and the formulation of the challenge problems. In order to obtain stable predictions, our methods are strongly based on statistical decision [8, 1] and learning theory [3, 6], section 4 is devoted to this topic. Final approaches to the challenge problems are presented in sections 4.1-4.3. The last section contains a short conclusion.

2 Exploratory Data Analysis

Any serious data mining task concerning unknown real datasets should be preceded by an appropriate *exploratory data analysis* phase which is regarded as being crucial to obtain high quality results in the subsequent phases [2]. All the computations given in this section were performed using the R package [7].

The datasets concern 4882 different users and 20 different Web page categories. All the users were represented in the training as well as the testing dataset. The number of records per user in each dataset is summarised in Figure 1.



dataset	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	200+ records
train	7.00	54.00	72.00	77.73	94.00	497.00	7
test (doubled)	2.00	46.00	64.00	68.13	86.00	430.00	5

Fig. 1. Summary of the statistics for the number of records per user in the training and testing (doubled) datasets. The typical number of records per user lies between 50 and 90 and does not vary very much but is not high enough (given the number of different categories, and other attributes) to build detailed user-level probability model.

We discovered (fig. 2) a group of users (the highest id numbers) which were "new" to the recording system at the end of the training dataset. Interestingly, the id number growth in this group is both higher and almost constant what makes it homogeneous (fig. 2, top-left). Since we believe that the user numbers must be assigned in some natural (perhaps, chronological) order, this may mean that this group of users is distinct. No such a group was found in the testing set.

Having inspected the data, we henceforth assume that the timestamp attribute represents the number of seconds passed since the beginning of the era.

The timestamp range is the following. Training set: 31/12/2006 - 22/01/2007 (about 1 am), testing set: 22/01/2007 - 31/01/2007 (about 1 pm). The testing set is a chronological continuation of the training set (with less than 1 minute break in between). This chronological relationship was taken into account while selecting our prediction models. It also encouraged to use time-series approach.

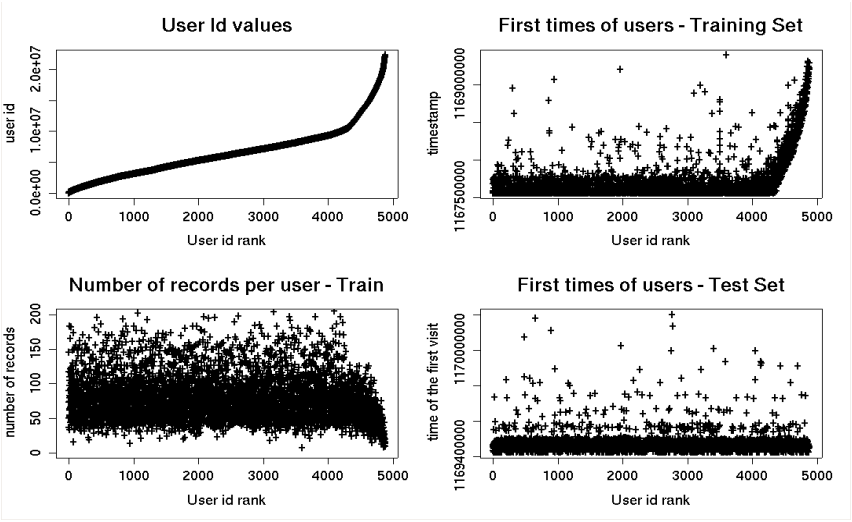


Fig. 2. The users with the highest id numbers are "newcomers" in the training set. The testing set does not contain users with analogous property.

We next transformed the timestamp attribute to the full date (i.e. `year`, `month`, `month-day`, `week-day`, `hour`, `minute`, `second`) to explore the week and 24-hour periodicity in users' behaviour, among others (fig. 3).

The difference of histograms (fig. 3, the middle column) of week-day-based activity is due only to more Mondays and Tuesdays in the testing set and abnormal activity on the Sunday, 31st December 2006, perhaps due to the New Years Eve greetings traffic, etc. After normalisation, both histograms would be almost flat. Thus, week-day is not a good discriminant at the global level. In contrast, the hour attribute seems to bring valuable discriminative information even on the global level (see fig. 3, left column and the top right histogram).

Table 1 summarises visit length (in the context of task 1). The following analysis was made in the context of tasks 2 and 3 of the challenge.

For each session in the training set we recorded the category on the first 3 positions of the visit path. Subsequently, the above data was aggregated over

Table 1. Statistics concerning the visit length – i.e. the number of consecutive page categories visited in each session. The distribution is extremely right-skewed. The number of recorded visited categories seems to be artificially cut at the value of 200 (it was clearly visible on one of our histograms not included here)

1 category	2 categories	3 cat.	3+ cat.	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
72.9%	11.5%	6.3%	9.25%	1	1	1	2.17	2	200

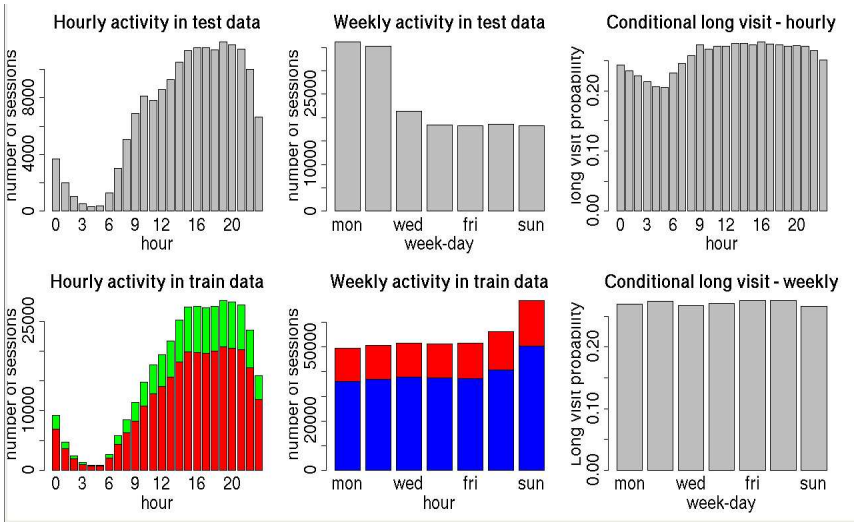


Fig. 3. Periodicity exploratory analysis. For the training set, the shorter bars represent the sessions in which more than 1 category of pages was visited, the longer bars - the other cases (task 1 of the challenge). One can observe (the histograms on the left, and top-right) that hour brings more information that week-day (almost flat bottom-right conditional histogram), on the global level.

separate users and, for each user, three 20-dimensional distribution vectors over categories were computed - for the 1st, 2nd and 3rd category on the visit path (e.g. if a particular user visited only category 12 and 8 on the first position of their session with equal frequency, the corresponding 1st-category distribution vector has entries of 0.5 on positions 8 and 12 and values of 0 elsewhere).

Subsequently, those probability distribution vectors served as the basis for computing entropy (left column on fig. 4), number of non-zero entries (the middle column) and the probability of the most likely category on the position 1, 2 or 3, for a given user (the right column on the figure).

All those measurements served to convince us that simple, user-level model for tasks 2 and 3 is a reasonable solution. Namely, the graphs on Figure 4 clearly show, that for most of users the categories on their 1st, 2nd and 3rd positions are quite easily predictable. In particular, low entropy, low number of categories encountered on the positions under examination and generally very high probability of the most likely category on a given position strongly influenced our decision of choosing very fast, yet simple prediction method for these tasks. One can easily observe from the fig. 4 that especially categories on the 1st and 3rd positions seem to be easily predictable. The phenomenon of the 2nd category being much harder to predict can be explained as follows. In 74.5% of sessions of over 2 categories the third category is the same as the 1st category.

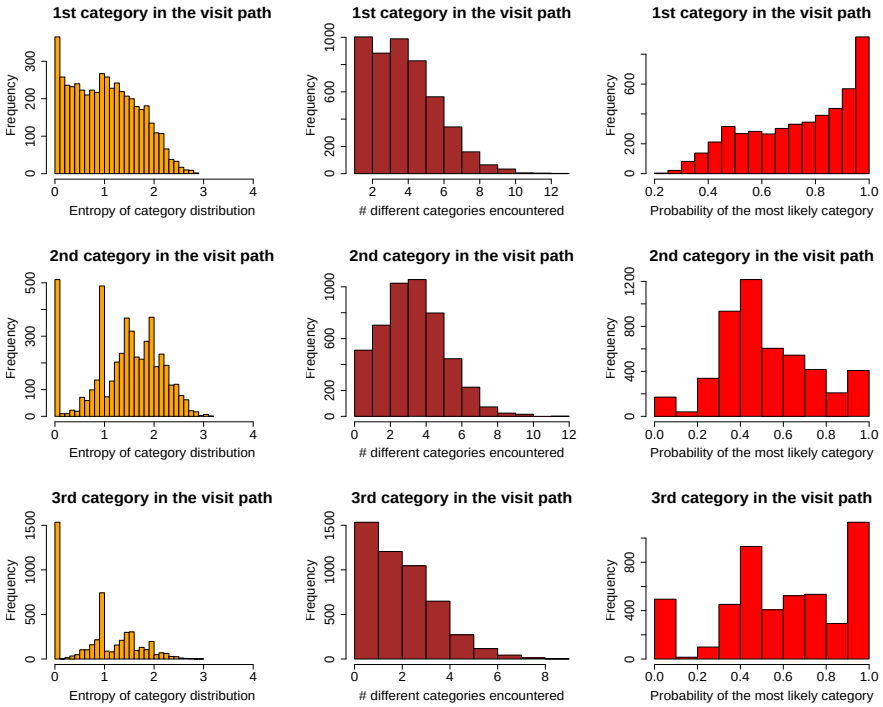


Fig. 4. Analysis focused on assessing feasibility of user-level simple modelling for the Task 2 and 3. Row number corresponds to the position on visit path. Left column: entropy of 20-dimensional probability distribution over categories – notice its relatively low values (maximum entropy for 20-dimensional distribution is 4.32193). Middle column: number of different categories on a given position (notice that it is close to 1). Right column: data-estimated probability of the most likely category on a given position (most of the mass is definitely above the value of 0.5, except 2nd category)

3 First Attempts on Global Models

The property which is apparent in the collected dataset is its granularity. The finer granule concerns a single visit (*visit's granule*), while the coarser granule corresponds to a single user (*user's granule*). On the one hand, using visit's granules permits the classifier to evaluate each visit separately, possibly giving different responses in each case, e.g. depending on the position of visit in chronological order, timestamp, etc. On the other hand, for user's granules we are able to do the averaging over all the visits for a given user, thus reducing the variance and giving more reliable responses.

Simple Global Model. Our first attempt was related to visit's granules. We considered the global model, in which we divided the original training set into 2 parts: training 89% and testing 11%. The division reflects the chronological re-

lationship between the original training and testing sets. The first 89% of the recorded sessions for each user constituted the training subset, and the last 11% the testing subset. To train the classifiers, we used features such as user data (`country`, `region`, `city`, `system`, etc.), week day, hour, part of the day, time from the last visit, number of visits during the day, number of visits in last 60, 120, etc. minutes, type of a last visit (whether it was a short or long visit), type of a second last visit, etc. However, obtained results were not satisfactory. The best result for task 1, which we have obtained using j48 (C4.5 implementation in Weka [9]), was 75.7% correctly classified visits.

Enhanced Global Model. Due to the poor quality of the results of the simple global model, we decided to estimate some additional values describing the behaviour of the users. In order to achieve it, we prepared an estimation set isolated from the training set. To reflect the chronological relationship present in data, the first 70% of the recorded sessions for each user constituted the estimation subset, the next 19% – the training subset, and the last 11% – the testing subset. The features calculated on the estimation set were:

- category-based (210 attributes in total): average number of pages seen in a session, average number of groups seen each session, number of different categories seen each session, majority category at position 1 through 3 on the path, average number of pages on each position (1st through 3rd), average number of pages seen in each category, average number of groups of each category, distribution of categories encountered on the 1st-3rd position of the path for each category, average number of pages seen in the 1st-3rd position for each category.
- visit length-based, obtained by considering only two types of visits (short or long) and estimating the probability of long visit for each day of week, for each hour of working day and hour of weekend day. Probability estimates were smoothed using the kernel estimation method (Gaussian kernel).

Notice that all those features represent some average characteristics of each user so that they are related to user's granularity level. In addition, user data (`country`, `region`, etc. - 7 attributes in total) were also included in the dataset.

We experimented with taking subsets of the above attributes. We also tried to take logarithms of some of the above attributes (those which were extremely right-skewed). In this setting, for the task 1, the best results have been obtained with j48 algorithm. The result was 76.7% correctly classified visits.

User Models. In the previous approach, the information about users was incorporated to the model by isolating the estimation set (including most of the observations) and calculating some coefficients for each user by averaging over their visits. In this approach, we decided to use *directly* user id number (attribute `user id`), without extracting any additional information about the users. This approach has been verified to be the most successful, therefore will be described in the next three sections (sections 4.1-4.3), separately for each task. Here we present common features of all the models.

Incorporating `user id` number as a condition attribute leads to the following problem: `user id` has nominal scale without any order between values, so that each of its values must be treated separately. It is possible to include such attribute in a general model, but for most of the classifiers, it will be binarised, i.e. changed into 4882 (number of users) binary attributes. Taking into account the size of the dataset, this is not a practical solution. Much more practical procedure, which can simulate conditioning on `user id` attribute, corresponds to building a separate model for each user. Such a procedure has been used in all of the models described later.

In each of the models `user id` number is used as one of the predictors (condition attributes). However, in none of the models any other attributes of the user (`country`, `region`, etc.) are included. This is due to the fact, that those attributes functionally depend on `user id` number, or in other words, `user id` number determines values on those attributes. Thus, they do not introduce any additional information, or in other words, they do not lead to finer granulation.

4 Final Solutions on User Models

All of the solutions described in this section are based on the statistical decision [8, 1] and learning theory [3, 6]. First, we briefly remind the basic concepts.

In the *prediction problem*, the aim is to predict the unknown value of an attribute y (called *decision attribute*, *output* or *dependent variable*) of an object using known joint values of other attributes (called *condition attributes*, *predictors*, or *independent variables*) $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The task is to find a function $f(\mathbf{x})$ that predicts value y as well as possible. To assess the goodness of prediction, the *loss function* $L(y, f(\mathbf{x}))$ is introduced for penalising the prediction error. Since $\mathbf{x}$ and y are random variables, the overall measure of the classifier $f(\mathbf{x})$ is the *expected loss* or *risk*, which is defined as a functional:

$$R(f) = E[L(y, f(\mathbf{x}))] = \int L(y, f(\mathbf{x}))\mathrm{d}P(y, \mathbf{x}) \tag{1}$$

for some probability measure $P(y, \mathbf{x})$. The optimal (risk-minimising) decision function is:

$$f^* = \arg\min_f R(f). \tag{2}$$

Since $P(y, \mathbf{x})$ is unknown in almost all the cases, one usually minimises the *empirical risk*, which is the value of risk taken from the set of training examples $\{y_i, \mathbf{x}_i\}_1^N$:

$$R_e(f) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)). \tag{3}$$

Function f is usually chosen from some restricted family of functions.

When solving the contest tasks, the problem was to find the best approximation of the optimal decision function.

4.1 Solution to the Task 1

The task is to predict whether a visit has page views of only one category (*short visit*), or more categories (*long visit*). We deal here with two classes, thus it is a simple binary classification problem for which the most common loss function is so called *0-1 loss*:

$$L_{0-1}(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}), \\ 1 & \text{if } y \neq f(\mathbf{x}). \end{cases} \tag{4}$$

Coding short visit by -1 and long visit by 1, the optimal decision function for a given $\mathbf{x}$ is:

$$f^*(\mathbf{x}) = \operatorname{sgn}\left(\Pr(y = 1|\mathbf{x}) - 0.5\right). \tag{5}$$

Of course, we have no information about the probabilities $\Pr(y = 1|\mathbf{x})$, so they must be estimated from training examples (alternatively, one can estimate whether the probability is higher or smaller than 0.5).

Shrinkage. Let $\hat{p}(\mathbf{x})$ be an estimator of the probability $\Pr(y = 1|\mathbf{x})$ (denoted as $p(\mathbf{x})$ from this moment on), which is calculated on the dataset. Our objective is to find the decision function defined as:

$$\hat{f}^*(\mathbf{x}) = \operatorname{sgn}\left(\hat{p}(\mathbf{x}) - \theta\right) \tag{6}$$

where, comparing with (5), we used the estimator $\hat{p}(\mathbf{x})$ instead of real unknown probability $p(\mathbf{x})$ and threshold θ instead of 0.5. The motivation for the latter is based on Bayesian inference. Suppose that we impose some prior distribution τ on parameter $p(\mathbf{x})$. It is easily seen from the data that $E_\tau p$, the expected value of p according to the prior distribution τ, is much less then 0.5, since in 73% of the cases the visit is short. It is a well known fact from Bayesian decision theory that the estimated parameters are shrunk towards the center of prior distribution. We impose such shrinkage by introducing regularised estimate of the probability defined as: $\tilde{p}(\mathbf{x}) = \alpha\hat{p}(\mathbf{x}) + (1 - \alpha)E_\tau p$, where α is chosen to be independent of $\mathbf{x}$ for simplicity. But the condition $\tilde{p}(\mathbf{x}) \geq 0.5 \Leftrightarrow \hat{p}(\mathbf{x}) \geq \theta$ where $\theta = \frac{1}{\alpha}0.5 + \frac{\alpha-1}{\alpha}E_\tau p$, and $\theta > 0.5$ as long as $E_\tau p < 0.5$. θ was chosen empirically to maximise the performance on the testing set.

The crucial thing in estimating $p(\mathbf{x})$ is the chosen vector of predictors (condition attributes) $\mathbf{x}$. Depending on the choice, three different models are considered, described below.

Model I: Simple Classification. The only predictor is `user id` number, so that $\mathbf{x} \equiv j$, where j is the number of user. For each user j, the fraction of the long visits was taken to be a probability estimator $\hat{p}(j)$. This estimator is constant in all observations for a given user. Therefore, it is characterised by a very small variance, but also a significant bias. The time complexity of the algorithm is linear with the number of visits. The memory complexity is linear with the number of users (not including the memory occupied by the dataset).

Model II: Trend prediction. Data for each user was regarded as a short time series with values 0 (short visit) or 1 (long visit). The abscissa values (predictors) were timestamps of the observations (normalised, in order to avoid some numerical difficulties due to large numbers). For each user, a polynomial trend was fitted to the time series and was used as a probability estimator. The fitting procedure was regularised least squares (ridge regression). The amount of regularisation was chosen empirically, to maximise the performance of the procedure on the testing set. It appears that models with very strong regularisation (more smoothing) are preferred due to their small variance.

For a given user, the time complexity of the method is dominated by the least squares fitting which is done by Cholesky decomposition and has complexity $O(m^3 + \frac{nm^2}{2})$, where n is the number of visits for a given user and m is the degree of the fitted polynomial. Since m is fixed, time complexity is linear in the number of visits as well as the memory complexity.

Model III: Autoregression. The autoregressive model was the most sophisticated one that we used. For each user a separate linear model is fitted to the user's time series, based on the following attributes: normalised timestamp, time from the last visit, length of the last visit, average length of the last 2, 4 and 8 visits. Since the predicted value (length of the current visit) depends on the values in previous moments, such algorithm resembles autoregressive models used in time series analysis [5], but with regularised least squares fitting procedure.

The classification procedure is more complicated here – all the objects must be classified chronologically, since the current value depends on the previous values. This causes the model to be less reliable with predicting the latest observations. That is why strongly regularised models (more smoothing, less variance) were preferred.

The complexity of the method, both in time and memory, is the same as for model II (linear in the number of visits), since least squares are also used as fitting procedure. However, training the autoregressive model takes more time due to the greater number of condition attributes.

Results and conclusions. For all our models, we present the 3 following estimates:

1. *training score* – value of the score on the training set. This estimate is thus over-optimistic, since the score is measured on the data which were used for fitting the classifier,
2. *validation score* – the training set was divided into 89% proper training set and 11% validation set. The classifier was learned on the proper training set and the score was calculated on the validation one. This estimate was used to choose the best classifier for the contest,
3. *solution score* – value of the score on the testing set. We were able to calculate this estimate using the proper solution sent by organisers after finishing the contest. The classifier was learnt on the whole training set.

The threshold θ was chosen to be equal to 0.55. This value was obtained by repeatedly fitting the classifier for values between 0.5 and 0.6 and choosing

Table 2. Values of the score (accuracy of classifier) for task 1.

Classifier	Score			time [sec.]
	training	validation	solution	
Majority vote	0.7292	0.7285	0.7331	0.021
Simple classification	0.7733	0.7661	0.7669	0.060
Trend prediction	0.7729	0.7696	0.7690	1.793
Autoregression	0.7781	0.7717	0.7687	7.842

the best results (validation score). The value of the score is (in case of task 1) the accuracy of the classifier, i.e. the fraction of correctly classified observations. The results are presented in Table 2. A "majority vote" classifier is also included which always assigns values from the larger class (short visit in this case – results of this classifier coincide of course with values presented in Table 1). We also present computational time for each classifier (calculated on the notebook with 512MB RAM and 2.13GHz Athlon processor), which includes training and classification of the testing set, but which does not include reading both files (training and testing) from disk into memory (it took additional 6.409 seconds).

Notice that although regression results were sent for the contest (the highest validation score), the best results on the testing set were achieved by the trend prediction approach (the highest solution score). All the models were relatively fast, especially majority vote and simple classification. Also all the models, apart from majority vote, have very similar results (almost no change in the score value). This suggests that any other condition attributes based on timestamp or previous visit length are hardly informative. This would also suggest choosing the simplest (parsimonious) model which is simple classification.

In comparison to the global models presented in section 3, the simple classification is only slightly worse than the best result obtained by enhanced global model, while trend prediction and autoregression seem to be better. The user models are simpler (less attributes taken into account), more stable, easier in parametrisation, and for these reasons, much more faster.

4.2 Solution to the Task 2

The second task concerns predicting a list of the 3 most probable categories (i.e. the 3 first page categories on the visit path) during a given visit of a given user. A specific score function was defined by the organisers in order to quantitatively measure the goodness of prediction. Assume that $y = (y_1, \ldots, y_m)$ is a sequence of m visited categories. Moreover let $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))$ be the sequence of the 3 most probable categories predicted by the classifier based on some predictor vector $\mathbf{x}$. In case of y as well as $f(\mathbf{x})$, according to the challenge rules, we assume that when a category appears more than once in the sequence, each time it is regarded as new, different category. The loss function is defined as

Table 3. Values of the score for task 2 and 3

Task	Score			time [sec.]
	training	validation	solution	
2	5.6830	5.6021	5.5606	30.124
3	6.5041	6.3747	6.3147	30.545

negative score and can be written in the following way:

$$L(y, f(\mathbf{x})) = -\sum_{j=1}^{m}\sum_{k=1}^{3} s(j,k) I(y_j = f_k(\mathbf{x})) \tag{7}$$

where

$$s(j,k) = \max\{1, \min\{6-j, 6-k\}\} \tag{8}$$

is the single score value and $I(x)$ is the indicator function equal to 1 if x is true, 0 otherwise. The risk of the classifier has the following form:

$$R(f) = \int \sum_{y} L(y, f(\mathbf{x})) P(y|\mathbf{x}) \mathrm{d}P(\mathbf{x}) \tag{9}$$

In order to find the optimal decision for a fixed predictor $\mathbf{x}$, we must minimise the risk point-wise, i.e. minimise $\sum_y L(y, f(\mathbf{x}))P(y|\mathbf{x})$. Since the probabilities $P(y|\mathbf{x})$ are unknown, we use empirical risk minimisation (3), so that we minimise the loss function (7) on the dataset. However, one can show, that estimating the probabilities $P(y|\mathbf{x})$ by frequencies would lead exactly to the same result.

Thus, the optimal decision function is obtained simply by choosing for each $\mathbf{x}$ value $f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x}))$, which minimises the empirical risk. However, one does not need to go through the whole dataset for each combination of values of $f(\mathbf{x})$. It is enough to calculate three aggregated coefficients for each category for a given $\mathbf{x}$ and then choosing $f(\mathbf{x})$ is independent of the number of visits.

There is still $\mathbf{x}$ to be chosen to make the model complete. Motivated by our exploratory analysis (see section 2) and the results for task 1, we decided to use parsimonious model taking into account only one predictor - `user id` number. Thus, the classifier is constant on every visit of the same user, so that it corresponds to user's granulation described in section 3. Such model has the advantage of being stable and having small variance. The results obtained using the model are presented in Table 3. They indicate that the model, despite its simplicity, has fairly good accuracy.

The time complexity of the method is linear with the number of visits. The memory complexity is (not including the dataset itself) linear with the number of users.

4.3 Solution to the Task 3

The third task was an extension of the second task. Apart from giving a list of the most probable categories in a visit, it concerns giving a range of number

of page views in each category. Assume that $y = ((y_1, t_1), \ldots, (y_m, t_m))$ is the sequence of m visited pairs (`category`, `#pages_range`) and

$$f(\mathbf{x}) = ((f_1^c(\mathbf{x}), f_1^t(\mathbf{x})), (f_2^c(\mathbf{x}), f_2^t(\mathbf{x})), (f_3^c(\mathbf{x}), f_3^t(\mathbf{x})))$$

is the sequence of 3 most probable pairs (`category`, `#pages_range`) predicted by the classifier based on some predictor vector $\mathbf{x}$. Then, the loss function is:

$$L(y, f(\mathbf{x})) = -\sum_{j=1}^{m}\sum_{k=1}^{3} s(j,k)I(y_j = f_k^c(\mathbf{x})) + I(t_j = f_k^t(\mathbf{x})) \tag{10}$$

Similarly as in the case of task 2, we minimised the empirical risk (3) to obtain the decision function. Again, only one predictor $\mathbf{x}$ was chosen – `user id` number. The results are shown in Table 3. The time and memory complexity of the method is the same as for the task 2.

5 Conclusion

After an intensive exploratory data analysis phase we examined a few approaches to the contest tasks and chose the solution that is simple, but effective and theoretically well-founded. We found this choice optimal in the context of the limited time. All of our algorithms scale well with large data and have linear time complexity, which is the smallest possible complexity for such problems, since reading the dataset is already linear in its size. The memory complexity never exceeds linear rate and grows linearly with the number of users, not visits.

We believe that there is still some improvement of the result possible and plan to explore it in a subsequent work. We plan to further experiment with our attributes and clustering users in order to obtain larger granules which in turn make it possible to compute estimates for models with richer structure.

References

1. Berger, J.: *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York (1993)
2. Dasu, T., Johnson, T.: *Exploratory Data Mining and Data Cleaning*. Wiley (2003)
3. Duda, R., Hart, P., Stork, D.: *Pattern Classification*. Wiley-Interscience (2000)
4. ECML/PKDD'2007 Discovery Challenge: User's behaviour prediction `http://www.ecmlpkdd2007.org/challenge/` (2007)
5. Hamilton, J. D.: *Time Series Analysis*. Princeton University Press (1994).
6. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. Springer (2003)
7. R Development Core Team: *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, `http://www.R-project.org` (2005)
8. Wald, A.: *Statistical decision functions*. Wiley (1950)
9. Witten, I., Frank, E.: *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann (2005)

Bayesian Inference for Web Surfer Behavior Prediction

Malik Tahir Hassan, Khurum Nazir Junejo, and Asim Karim

Dept. of Computer Science, Lahore University of Management Sciences
Lahore, Pakistan
{mhassan, junejo, akarim}@lums.edu.pk

Abstract. The accurate prediction of user behavior on the Web has immense commercial value as the Web evolves into a primary medium for marketing and sales for many businesses. This broad and complex problem can be broken down into three more understandable problems: predicting (1) short and long visit sessions, (2) first three most probable categories of pages visited in a session, and (3) number of page views per category in a visit session. We present Bayesian solutions to these problems. The focus in our solutions is accuracy and computational efficiency rather than modeling the complex Web surfer behavior. We evaluate our solutions on four weeks of surfer data made available by the ECML/PKDD Discovery Challenge. Probabilities are estimated from the first three weeks of data and the resulting Bayesian models tested on last week's data. The results confirm the high accuracy and good efficiency of our solutions.

1 Introduction

Web users definitely exhibit patterns of surfing behavior. Discovering such patterns have immense commercial value as the Web evolves into a primary medium for marketing and sales for many businesses. Web-based businesses seek useful users' patterns to help identify promising events, potential risks, and make strategic decisions. Web surfer behavior modeling and prediction has been a popular research topic. Over the years numerous approaches have been proposed for solving various aspects of the problem with varying degrees of success. In general, the problem involves the prediction of a user's sequence of page views based on previous history of the user. Oftentimes, to simplify the problem somewhat, Web pages are abstracted and grouped into categories and the problem is reduced to the prediction of a user's sequence of categories visited. Nonetheless, this is a complex machine learning problem that requires careful consideration from the technical and practical points of view.

Among the various approaches used for the modeling and prediction of Web surfer behavior, probabilistic approaches have been very common [1-5]. Borges and Levene [1] propose the use of N-gram probabilistic models which assume that the probability of a page visit depends only on the last N pages browsed. Similarly, Manavoglu et al. [2] present probabilistic user behavior models by applying maximum entropy and Markov mixture models. For prediction for known users, they propose a Markov

model. Another probabilistic solution is presented by Deshpande and Karypis [3]. They try to reduce the state complexity resulting from all kth-order Markov models by pruning many of the non-affecting states. Eirinaki et al. [4] present a hybrid probabilistic predictive model by extending the properties of Markov models with link-based methods such as PageRank. Such an approach is applicable only when structural link information of the pages is known. Lu et al. [5] group or cluster clickstream data using a pair-wise alignment algorithm. Then, a first-order Markov model is built for each cluster of sessions.

The majority of the approaches try to tackle the general Web surfer behavior modeling problem rather than specific prediction problems. This often makes the solutions complex and difficult to interpret. The Web surfer behavior prediction problem can be broken down into three sub-problems: (1) predicting short and long visit sessions by users, (2) predicting first three most probable categories of pages visited by users, and (3) predicting range of page views per category made by users. These sub-problems capture key Web surfer behaviors of practical value. Moreover, they represent simpler problems in comparison to the general Web surfer behavior prediction problem.

In this paper, we present Bayesian solutions to these problems. In particular, we develop Bayes classifiers for each sub-problem, invoking the naïve Bayes assumption of conditional independence of the input given the class. We model the sequence of page categories visited as a Markov chain. The naïve Bayes assumption and the first-order Markov property are made to improve space and time efficiency of the solutions. The performance of our solutions is evaluated on four weeks of data made available by the ECML/PKDD Discovery Challenge [6]. The results show high prediction performance identical to those produced by a support vector machine (for problem 1). Moreover, our solutions are time and space efficient.

The rest of the paper is organized as follows. We formally describe the Web surfer prediction problems in Section 2. Our solutions to the problems are described in Section 3. Experimental evaluation of the solutions, including their complexity analysis, is presented in Section 4. We conclude in Section 5.

2 Problem Description and Notation

Let variable $X = \{U, T\}$ identify a visit session, where variables U and T denote the user ID and the starting timestamp of the visit session, respectively. A visit session or path is described by a sequence of page categories visited during that session. Variable C_i identifies the category visited in position i of the sequence, and a visit session has one or more positions in the sequence. A particular visit session can have the same page category visited at different positions; however, two consecutive page categories must be different. Individual Web pages are abstracted and grouped into a finite number of page categories. The range of the number of page views made for a given page category is captured by the variable R_i, where i denote the ith position in the sequence. All variables have discrete and finite sets of possible values. The variable T is discretized into time slots. The historical training data available to the learning system contains unique visit sessions represented by instantiations of the

variables X, C_i's, and R_i's. The test data contain different instantiations of the variable X only.

The Web surfer behavior prediction problem is divided into three sub-problems. Problem 1 is to learn to predict whether a visit session X is short or long. A visit session is said to be short if it contains one sequence position only; otherwise, it is said to be long. Problem 2 is to learn to predict the first three page categories for a given visit session X. Problem 3 is to learn to predict the range of page views for each page category in positions 1, 2, and 3 for a visit session X. All three problems are classification problems. The objective in each is to predict the output as accurately as possible.

3 Our Solution

We present Bayesian solutions to the three problems described in the previous section. The Bayesian approach has been adopted for the following reasons: (1) it is simple and intuitive, providing insight into the problem and its solution, (2) it is adaptable to concept drift, and (3) it is computationally efficient and acceptably accurate. In particular, we use a Bayesian classifier for each of the three problems, as described in the following sub-sections.

3.1 Problem 1

This is a two-class classification problem. We present a naïve Bayes classifier for its solution. Given a visit session X, the most probable class $z = Z \in \{long, shart\}$ is given by

$$z = \underset{z \in \{long, short\}}{\arg\max} \; P(Z = z)P(X \mid Z = z) \tag{1}$$

where $P(.)$ denotes the probability of the enclosed event. If we assume that the user ID U and the timestamp T are conditionally independent of each other given class Z, we get the naïve Bayes classification:

$$z = \underset{z \in \{long, short\}}{\arg\max} \; P(Z = z)P(U \mid Z = z)P(T \mid Z = z) \tag{2}$$

This represents the most probable class under the naïve Bayes assumption. If we do not consider the timestamp T of a visit session, the last probability in Equation (2) drops out further simplifying the solution.

3.2 Problem 2

This problem involves the prediction of the first three page categories of a visit session. To solve this problem, we model the sequence of page categories visited as a Markov chain. The chain start state (the first page category) is determined by a Bayes

classifier. Subsequent states are determined by combining the posterior probability estimates given by the Markov chain with that of the Bayes classifier for that particular position. The reason for selecting the first-order Markov model over the kth-order model is two fold: (1) The problem involves the prediction of only the first three states for which a first-order Markov model is sufficient. and (2) The first-order Markov model is computationally efficient. Moreover, we believe that a kth-order model is more realistic for modeling page view transitions rather than page category transitions.

According to the Bayes rule, the posterior probability of page category C_i visited in position i (i = 1, 2, or 3) of a visit session X is given by

$$P^B(C_i \mid X) = P(C_i)P(X \mid C_i) / P(X) \tag{3}$$

The most probable page category visited at the start of the sequence ($c_1 = C_1$) is then given by

$$c_1 = \arg\max_{c_1} P(C_1 = c_1)P(X \mid C_1 = c_1) \tag{4}$$

This fixes the start state of the Markov chain. The subsequent states can be found by combining the predictions of the Bayes classifier (Equation 3) and the Markov model. According to the Markovian property, for a given visit session X the posterior probability of page category C_i visited in position i (i = 2, 3) depends only on C_{i-1} and can be expressed as

$$P^M(C_i \mid C_{i-1}, X) = P(C_i \mid C_{i-1})P(X \mid C_i, C_{i-1}) / P(C_{i-1}, X) \tag{5}$$

The page category visited at position i (i = 2, 3) is then given by

$$c_i = \arg\max_{c_i} P^B(C_i = c_i \mid X)P^M(C_i = c_i \mid C_{i-1}, X) \tag{6}$$

Notice that in evaluating Equation (6), we do not need to estimate the unconditional probabilities in the denominator of Equations (3) and (5).

If a visit session X is described by user ID U and timestamp T, the naïve Bayes assumption can be invoked to simplify the expressions above, as shown for problem 1.

3.3 Problem 3

This problem involves the prediction of the range of the number of page views for the first three page categories visited in a visit session. The page categories c_i (i = 1, 2, 3) visited have been determined in problem 2. We use a Bayes classifier to predict the range $r_i = R_i$ of page views made at position i (i = 1, 2, 3) in visit session X as

$$r_i = \arg\max_{r_i} P(R_i = r_i \mid C_i = c_i)P(X \mid R_i = r_i, C_i = c_i) \tag{7}$$

The page category c_i is the one predicted in problem 2.

3.4 Estimating the Probabilities

The various probabilities used in our solution are estimated from the historical training data by maximum likelihood estimation. Since all variables are observed in the training data, maximum likelihood estimates are equivalent to the frequencies in the data. Specifically, the probability estimate of $P(X = x|Y = y)$ is given by

$$P(X = x \mid Y = y) \approx \frac{\text{no. of examples with } X = x, Y = y}{\text{no. of examples with } Y = y} \tag{8}$$

For an unconditional probability, the denominator will be the total number of examples in the training data. To estimate the transition probabilities in Equation (5), we count an example if it contains the given transition at any position of the sequence.

4 Evaluation

We carry out a number of experiments to demonstrate the efficiency and effectiveness of our solution to the Web surfer behavior prediction problem. The evaluations are performed on a desktop PC with an Intel 2.4 GHz Pentium 4 processor and 512 MB of memory.

4.1 Data and its Characteristics

We use the data provided by the 2007 ECML/PKDD Discovery Challenge [6]. The data were collected by Gemius SA, an Internet market research agency in Central and Eastern Europe, over a period of 4 weeks through use of scripts placed in code of the monitored Web pages. Web users were identified using cookies technology. The first 3 weeks of data are used for training while the last week of data are reserved for testing.

The data records individual visit sessions described by the fields: path_id, user_id, timestamp, {category_id, pageviews_number},…. An example visit session is shown below:

path_id	user_id	timestamp	path (category_id, pageviews_number) …, … …				
27	1	1169814548	7,1	16,2	17,9	16,1	…

The timestamp field records the time at which a visit session starts and the category ID field identifies a group of Web pages with similar theme such as entertainment, technology, or news. There are 20 page categories in the data. The entire data contain 545,784 visit sessions from which 379,485 visit sessions are used for training and the remaining 166,299 visit sessions are used for testing. There are 4,882 distinct users in the data.

An analysis of the training and test data reveals non-uniform data distribution. The minimum and maximum number of visits by a user in the training data is 7 and 497, respectively, with an average of 77.7 visits per user. The minimum and maximum

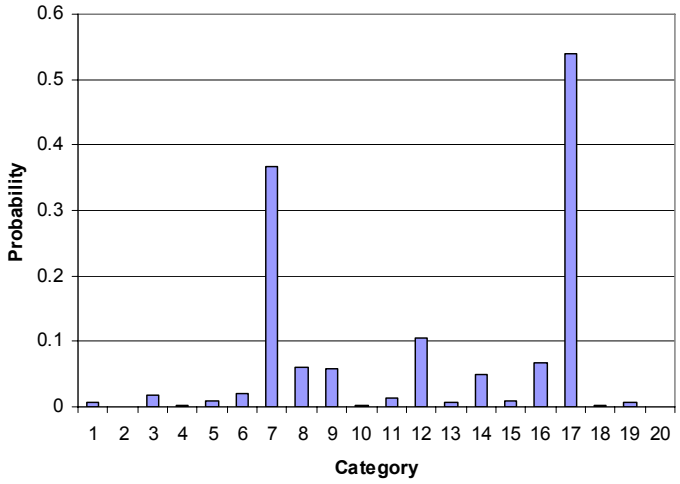


Fig. 1. Probability of categories in training data

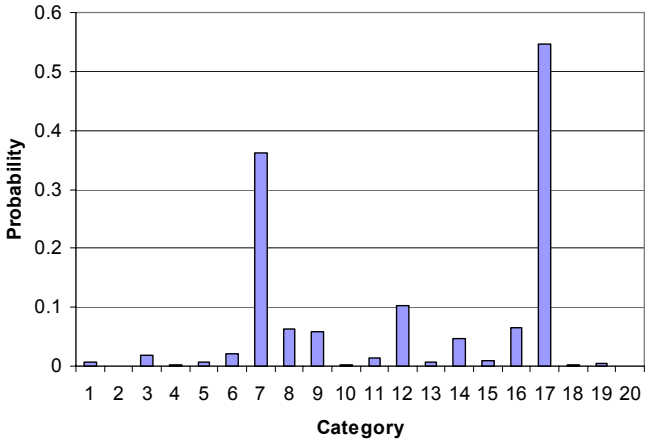


Fig. 2. Probability of categories in test data

number of visits by a user in the test data is 1 and 215, respectively. Similarly, the distribution of page categories is uneven. Some categories are being visited more frequently than others. This is evident from Figures 1 and 2 which show the probability of the categories in the training and test data, respectively. About 73% of the visit sessions in the training and data data are short, i.e., a visit where only one category is surfed. These statistics confirm that the data distributions of the test and training sets are similar.

4.2 Evaluation Criteria

Problem 1 is a two-class classification problem. The classification score, defined as the number of correct classifications, is used to evaluate this problem. Problem 2 is evaluated by computing a score. This score is the sum of scores of each prediction, where each prediction score is defined as follows: The prediction score is the sum of weights assigned to the 3 predicted categories. If the first, second, and third categories are predicted correctly, then assign weights 5, 4, and 3, respectively, to these positions. If a prediction is incorrect, then it is assigned a weight of 4 if that category occurs in the second position, 3 if it occurs in the third position, 2 if it occurs in the fourth position, 1 if it occurs in position five and beyond, and zero if it does not occur. The weight assigned cannot be greater than the maximum possible for that position (e.g. the weight assigned to position 2 cannot be greater than 4). Problem 3 is also evaluated by computing a score. This score computation is identical to that for problem 2 except that the weights are incremented by one if the predicted range is correct; otherwise they are not incremented. For all problems, higher scores signify better performance. The maximum possible score for each problem is also presented in our results

We present time and space complexity results in Sections 4.3 and 4.4.

4.3 Results

We present results for problems 1, 2 and 3 under two settings. In the first setting, we consider only the user ID as input while in the second setting we consider both the user ID and timestamp as input. We discretize the timestamp field into four values: weekday-day, weekday-night, weekend-day, and weekend-night. Daytime starts from 8AM and ends at 6PM. We tried several discretizations for timestamp but present results for the above defined discretization only. Problem 1 is also solved using a support vector machine (SVM) through SVM^{Light} [7]. The default parameters' settings of SVM^{Light} are used for this result.

The results for problems 1, 2, and 3 without and with timestamps are given in Tables 1 and 2, respectively. The accuracy of our solution without considering timestamps for problem 1 is 76.64%. The SVM also produces an accuracy of 76.64% for the same setting. Our achievement here is in terms of computational efficiency. For our hardware setup, our solution takes less than 1 minute to learn from the training data and classify the test data. In contrast, the SVM takes several hours to learn. When both user ID and timestamp are considered, the prediction performance of our solution drops slightly to 76.60% while that of SVM increases slightly to 76.68%. Including the timestamp field, however, decreases the time and space efficiency of the solutions.

A similar pattern of results is seen for the two settings of problems 2 and 3. For problem 2, the percentage score drops slightly from 83.2% to 83.17% when timestamp is considered together with user ID. On our hardware setup, it takes about 6 minutes without timestamps and about 15 minutes with timestamps to solve this problem (learning plus testing). For problem 3, the percentage score drops slightly from 72.53% to 72.42% when both timestamp and user ID are considered. Similarly,

Table 1. Prediction performance results for our solution without considering timestamp

	Problem 1	Problem 2	Problem 3
Score	127457	903145	958643
Max. possible score	166299	1085494	1321706
Percentage score	76.64%	83.20%	72.53%

Table 2. Prediction performance results for our solution when considering timestamp

	Problem 1	Problem 2	Problem 3
Score	127383	902849	957235
Max. possible score	166299	1085494	1321706
Percentage score	76.6%	83.17%	72.42%

the running time increases from about 1 minute to about 1.5 minutes when both timestamp and user ID are considered.

The interesting result is that considering timestamp decreases prediction performance (very) slightly (this drop may not be statistically significant). This is probably due to the greater chance of a probability estimate in our solution turning out to be zero adversely affecting the prediction. The SVM for problem 1 did show a slight increase in prediction performance. However, our solution is orders of magnitude more efficient than SVM.

4.4 Complexity Analysis

In this section, we discuss the computational complexity of our solution and demonstrate its efficiency.

The time complexity of our solution for all three problems is $O(N)$ where N is the total number of visit sessions in the data. The model is learned in $O(N)$ time and constant time is required to classify every test example as all the probabilities have been pre-computed.

The space complexity of our solution is defined by the number of probability estimates required. For problem 1, we require $2 + (4882 \times 2)$ estimates when timestamp is not considered and $2 + (4882 \times 2) + (4 \times 2)$ when timestamp is considered. In these expressions, 2 is the number of classes, 4882 is the number of distinct users, and 4 is the number of distinct timestamps. For problem 2 when timestamp is not considered, the number of probability estimates required is $(20 \times 3) + (4882 \times 20 \times 3) + (20 \times 20) + (4882 \times 20 \times 20)$. The first two terms correspond to the probabilities in Equation (3) and the last two terms correspond to the probabilities in Equation (5). When timestamp is considered an additional $(4 \times 20 \times 3) + (4 \times 20 \times 20)$ estimates are required. In these expressions, 20 is the number of categories and 3 is the number of positions.

For problem 3 without timestamp, the number of probability estimates required is $(3 \times 20) + (4882 \times 3 \times 20)$, where 3 is the number of page view ranges. When timestamp is considered, an additional $(4 \times 3 \times 20)$ estimates are required.

From the above results we see that space complexity is $O(N)$. As discussed earlier, the data is sparse and many probability estimates are zero. Hence smart selection of data structures can reduce the space requirements further. In our implementations, we use hash maps instead of matrices to store the non-zero probability values only.

5 Conclusion

In this paper, we present our solution to the 2007 ECML/PKDD Discovery Challenge on Web surfer behavior prediction. We adopt Bayesian approaches for all three problems of the challenge. For problems 1 and 3, which are standard classification problems, we use Bayes classifiers for their solution. For problem 2, which requires predicting the sequence of page categories visited, we combine Bayesian classification with Markov chain prediction. The solutions are evaluated on four weeks of data collected from Polish websites. The results show that our solutions are accurate and efficient. In particular, our solution to problem 1 has the same prediction accuracy as SVM but is orders of magnitude faster. We also find that incorporating the start time of visit sessions does not have any practical impact on prediction accuracy.

The problem of Web surfer behavior prediction is of immense commercial value. We believe that a direct solution to the problem is more practical than those involving complex Web surfer behavior modeling. As part of future work, we will explore other probability estimating approaches suitable for limited data and ways of boosting prediction performance.

References

1. J. Borges and M. Levene: Data mining of user navigation patterns. In *Proc. Of International Workshop on Web Usage Analysis and User Profiling* (1999)
2. E. Manavoglu, D. Pavlov, and C.L. Giles: Probabilistic user behavior models. In *Proc. Of International Conference on Data Mining (ICDM)* (2003)
3. M. Deshpande and G. Karypis: Selective Markov models for predicting Web page accesses. *ACM Transactions on Internet Technology* 4(2) (2004) 163-184
4. M. Eirinaki, M. Vazirgiannis, and D. Kapogiannis: Web path recommendations based on page ranking and Markov models. In *Proc. of the 7th Annual ACM international Workshop on Web Information and Data Management (WIDM)* (2005)
5. L. Lu, M. Dunham, and Y. Meng: Discovery of significant usage patterns from clusters of clickstream data. In *Proc. of WebKDD* (2005)
6. ECML/PKDD: Discovery challenge. http://www.ecmlpkdd2007.org/challenge (2007)
7. T. Joachims: Making large-scale SVM learning practical. In *Advances in Kernel Methods – Support Vector Learning*. MIT Press (1999)

Predicting User's Behavior by the Frequent Items

Tung-Ying Lee[1]

[1] Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan
tylee@cs.nthu.edu.tw

Abstract. Frequent items, frequent itemsets, frequent sequences, and graphical models have been used in predicting user's behavior. However, we found that frequent items are robust, time-efficient, and meaningful in the view of statistics. Our method is based on frequent items. The experiments are performed on a real dataset which is provided in ECML/PKDD Discovery Challenge. For most situations, we found that predicting from frequent items can obtain more convincing results than predicting from multiple Markov chains.

Keywords: Frequent Items, Markov Chains

1 Introduction

Due to the development of electronic commerce and Web services, web usage has a rapid increase in interest. Significant patterns derived from web usage have many applications. These patterns are used not only in load balance but also in personalizing product messages for individuals by system designers or vendors [2] [3]. Traditionally, web mining can be classified into following classes: content mining, structure mining, and web usage mining [2]. Web usage mining is concerned with the analysis of web usage.

In ECML/PKDD 2007 Discovery Challenge 1, usage data of three weeks and user profiles are provided and the goal is to predict user's behavior in the following week. Collecting user usage and user profiles is practical. Hence, we analyze these data fields for formulating our problems as learning or mining problems.

The rest of the paper is organized as follows. Section 2 describes two kinds of formulation. Some important observations and experiments are shown in Section 3. Our method and time/space analysis are presented in Section 4 and 5. The accuracy of prediction is calculated in Section 6. Section 7 concludes this paper.

2 Formulation

In ECML/PKDD 2007 Discovery Challenge 1, the data consist of two tables, Users table and Visit Paths table. The Users table provides the user profiles, including three types of information, geo-locations, user's operating system, and user's browser. Each tuple in other table is a path, which is a sequence of categories and corresponding page views during one visit of a user. The ordering in a path is sorted by the order of viewing. These data fields are listed below.

Table 1. Data fields in ECML/PKDD 2007 Discovery Challenge 1.

Table	Types	Field Format
Users Table	Geo-location	Country, Region, City
Users Table	Operating System	OS, OS Version
Users Table	Browser	Browser, Browser Version
Visit Paths Table	Time	Timestamp
Visit Paths Table	Path	(categories, pageviews), …

The goal of this challenge is to predict the first three categories and corresponding page views in the start of a new visit of a user. We formulate this problem as a mining problem and learning problem.

2.1 On-Line Analytical Processing

On-Line Analytical Processing (OLAP) has been studied for several years. Many tools provide OLAP functionality. We can transform our problem into an OLAP query. In this data set, Users table and Visit Paths table is a multidimensional data set with hierarchical structure. The dimension, geo-location, has three levels, country, region, and city, and the other two dimensions have two levels respectively. Multidimensional data are typical input for an OLAP system, which usually supports many operations for aggregating measure attribute like rollup (increasing the level of aggregation), drill-down (decreasing the level of aggregation), and slice_and_dice (selection and projection) [2]. The measure attribute in this data set is path. A prediction can be made by issuing an OLAP query to the OLAP system. A simplest prediction strategy may take the most popular category and the mean of page views for predicted values. The predictor in this strategy only request answers from the OLAP system.

2.2 Graphical Model Formulation

A graphical model can be used in problem formulation. For each user, a graphical model is trained from historical data. The graphical model is used to predict the next categories. Here, a directed graphical model (usually called Bayesian Network) is selected as the model. Each category in a visit path is a node in a Bayesian network. We do not put the page views into the graphical model; the prediction for page views

can be decided by statistical data, such as mean page views of a category for a user. Because only first three categories in a visit path are considered, the network consists of the first three categories of visit paths. Fig. 1 shows two kinds of graphical models. The first is Markov chain. The model is based on the assumption that the current category depends on the previous one category. However, the second is more complex and assume that the previous visit path has influence on the first categories in current visit.

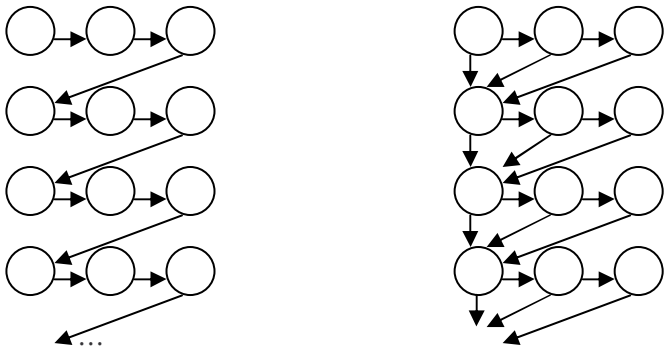


Fig. 1. A Markov chain (left) and a Bayesian network (right)

3 Observations

Although the problem is transformed into a mining or learning problem (See Section 2), the main difficulty is the design of queries in OLAP formulation or the selection of correct graphical model. Before designing our methods, we performed some small experiments:

A short visit is a visit with page views of only one category; a long visit has views of pages belonging to more than two categories. We define the major visit type of a user. A user is a long visit major if more than 50 percents of his visit paths are long visits; otherwise, he is a short visit major. The detailed is listed in Table. 2.

Observation 1. Most of the users who are short visit majors have higher ratio of short visits to all visit paths.

Table 2. The First Experiment: Long-visit-major User and Short-visit-major User.

Ratio /Major Visit Type	Long Visit	Short Visit	Total
0.50 ~	113	218	331
0.55 ~	100	235	335
0.60 ~	71	291	362
0.65 ~	66	373	439
0.70 ~	54	474	528
0.75 ~	58	532	590
0.80 ~	37	548	585
0.85 ~	39	580	619
0.90 ~	17	584	601
0.95 ~	14	478	492
Total	(569)	(4313)	(4882)

According to Observation 1, we know that most users of long visit major are with low ratio of long visits to total visit paths, i.e. it is harder to predict long-visit-major user behavior.

The second experiment is performed on 569 long-visit-major users. For each user, the first categories in his visit paths are collected, and the frequency of the most frequent item (denoted as $First_1$) is recorded. In the similar way, the most frequent item of the second categories and the third categories ($Second_1$ and $Third_1$) are calculated. The distribution of the frequency of these three terms is shown in Table 3.

Observation 2. The prediction of the first category in a long visit path is simpler than that of the second one or that of the third one.

Table 3. The Second Experiment: The distribution of the frequency of $First_1$, $Second_1$, and $Third_1$

Frequency /the ith categories	$First_1$	$Second_1$	$Third_1$
0.0 ~	12	37	38
0.1 ~	25	83	128
0.2 ~	74	112	122
0.3 ~	90	104	87
0.4 ~	82	65	66
0.5 ~	69	58	51
0.6 ~	67	41	38
0.7 ~	69	33	21
0.8 ~	41	25	10
0.9 ~	40	11	8
Median	(0.6325)	(0.4318)	(0.2750)

The second experiments show the correlation of the second category and $Second_1$ may be low, because most of the frequency of $Second_1$ is less than 0.5. (The median of the frequencies of $Second_1$ and $Third_1$ are 0.4318 and 0.2750.)

The previous experiments are simple queries for historical data. The following experiments belong to the learning approach.

We selected a model consisting of multiple Markov chains like Fig. 3. This model includes three types of probability function, the initial probability in the first category, transition probability from the first category in the previous visit to the current first category (called inter-transition probability), and transition probability from the previous category to the current category in the same visit (called intra-transition probability).

Some problems happen in this formulation. The first category *c* in the previous visit may have no transition to other categories, because this category c only appear in the last visit of historical visit paths of a user. For resolving this problem, we use $First_1$ serving as the prediction for the first category in the current visit. The computation for inter-transition probability is to collect all pairs $\{category\_id_1\} \rightarrow \{category\_id_2\}$. We add a new category "0" which means that users finish their visits. The last known visit paths is used to predict the first unknown visit path and this path is finished if a "0" category is predicted.

For testing this model, we perform prediction on the test data. The test data includes 166299 visits. If a predicted path is longer than 1, this visit is predicted as long visit. In experiment 1, all users are divided into long-visit-major and short-visit-major users. Assuming the long-visit-major (short-visit-major) users only generate long visits (short visits), we get the result R_1 from the statistics in experiment 1. Comparing this result derived from multiple Markov chains with R_1, the mismatch rate is 42.83%. It seems to be that multiple Markov chains do not capture the user behavior well. Thus, we have the following observation.

Observation 3. The multiple Markov chains model does not perform well.

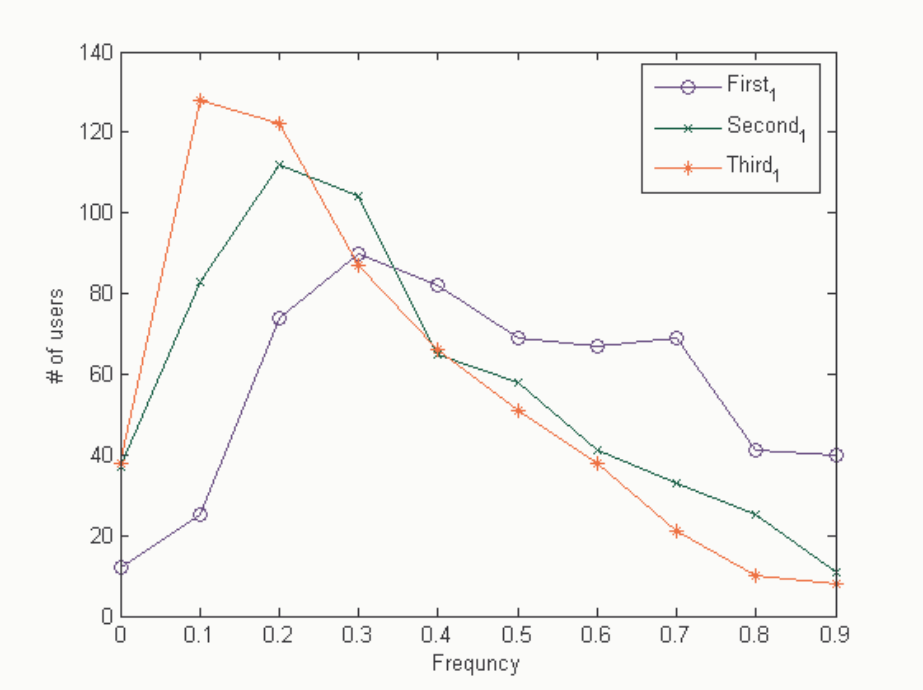


Fig. 2. The Line Graph for Table 3.

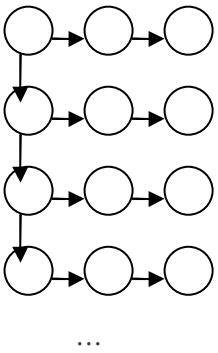


Fig. 3. Multiple Markov Chains

The last observation can be found from the historical data. For example, user 46 have three visit patterns, 12 → 8 → 12, 12 → 8, and 8 → 12.

Observation 4. If a pattern a → b → … is frequent; the pattern b → … is usually frequent.

Other fields in the Users table are considered in the last observation. Most of users (4739 out of 4882) are come from country 42. For analyzing the relationship between country and $First_1$, $Seocnd_1$, and $Third_1$, we use the entropy as the measure. We think that their entropies are too high; hence, we do not use country information.

Observation 5. If users are classified by their countries, user behavior is still hard to predict. It is because the variables of first three categories have entropies of values greater than 1 (nat).

Table 4. The Relationship between Country and $First_1$, $Seocnd_1$, and $Third_1$

Country_id	# of Users	$First_1$ (entropy) (nat)	$Seocnd_1$ (entropy) (nat)	$Third_1$ (entropy) (nat)
42	4739	1.5637	1.7325	1.868
170	47	1.5671	1.6283	1.7567
84	30	1.1295	1.85	1.9305
39	18	1.4571	1.6095	1.956
56	6	1.3297	1.0114	1.0114
14	5	0.95027	0.95027	0.67301
165	4	1.3863	1.0397	1.3863
18	4	1.0397	0.56234	1.0397
148	3	0.63651	1.0986	0.63651
115	3	1.0986	1.0986	1.0986
60	3	1.0986	0	0.63651
179	2	0.69315	0.69315	0
131	2	0.69315	0.69315	0
128	2	0.69315	0.69315	0.69315

4 Description of our Methods

After some experiments and observations, we adapt three frequent items to our prediction methods.

4.1 Method for Problem I

The first problem is to predict if the current visit is long or short. For each user, the method returns short or long depending on the number of short visits and the number of long visits. If this user is a short (long) visit major, the method returns short (long).

4.2 Method for Problem II/Problem III

The first three categories can be predicted by using the following steps:

(1) For each user, select all the first one categories in visit paths into a list C_1. The list C_1 may be {12, 8, 12, …}. The most frequent category I_1 in C_1 and the corresponding number of page views $n(I_1)$. The number $n(I_1)$ is the mean page views of I_1, and only page views of the first category are computed.
(2) For each user, select all the second categories in visit paths into a list C_2. The list C_2 may be {8, 8, 8, …}. The most frequent category I_2 in C_2 and the corresponding number of page views $n(I_2)$ are computed.
(3) For each user, select all the third categories in visit paths into a list C_3. The list C_3 may be empty. The most frequent category I_3 in C_3 and the corresponding number of page views $n(I_3)$ are computed.
(4) If C_i is empty, I_i and $n(I_i)$ are replaced by I_{i-1} and $n(I_{i-1})$.

5 Time/Space/Scalability Analysis

Our method is very efficient in time and space. If the whole data set *D* can be stored in memory, the frequent items in the first, second, and third categories ($First_1$, $Second_1$ and $Third_1$) can be calculated after one pass over data set. The implementation needs some counters for keeping the numbers of appearances of items. If the types of categories are fixed, additional space has the same order for the original data set, i.e. O(|D|). These counters can be randomly accessed if we use an array to implement these counters. Hence, the time complexity is also linear, i.e. O(|D|).

In scalability issue, the frequent item method is satisfactory. Because the most frequent item is kept, the size of the rule of prediction is the same to O(|users|) where the number |users| is the number of users. The calculation of frequencies of items may cost extra space O(|categories|) at most where the number |categories| is the number of categories. If the assumption that the whole data set is less than memory size is not hold, the increase of the size of data set has little influence on the execution time of frequent item method.

Our programs are implemented by using a script language Ruby. The training and prediction process cost a few seconds (~ 5 seconds) on the computer with AMD 1.8GHz Athlon 64 Processor and 1G memory.

6 Experimental Results

Our experiments are performed on the test data set provided by ECML/PKDD 2007 Discovery Challenge. The measure of most concern is the accuracy defined by |the_correct_items|/|items|. The accuracy for three problems is listed in Table 5. The accuracy for predicting page views must be less than the accuracy for category because the category must be correct. Hence, it is more difficult to predict page views

of a category. There are 20 kinds of categories. Although our prediction accuracies for the second and third categories are about 50%, this accuracy is about ten times higher than that from random guess.

Table 5. The Accuracy of Our Prediction Method.

Problem	Accuracy
Problem 1	76.64% (127457/166299)
Problem 2: the first category	74.37% (123669/166299)
Problem 2: the second category	49.70% (22059/44380)
Problem 2: the third category	57.52% (14670/25503)
Problem 3: page views of the first category (the predicted category must be correct)	44.62% (74201/166299)
Problem 3: page views of the second category (the predicted category must be correct)	25.00% (11093/44380)
Problem 3: page views of the third category (the predicted category must be correct)	32.52% (8294/25503)

7 Conclusion

For predicting user behavior of web usage, we formulated this problem in different views, and two methods are proposed. One is based on multiple Markov chains, and the other is based on frequent items. After dissecting the web usage data, we decide to use the latter. We analyze the prediction method based on frequent items; both its time and space complexity are linear. Its accuracy is tested on a data set provided by ECML/PKDD 2007 Discovery Challenge. The experimental results show the effectiveness of the proposed method.

References

1. Chaudhuri, S. and Dayal, U.: An overview of data warehousing and OLAP technology. ACM SIGMOD Record, Vol. 26, Iss. 1. ACM Press New York, NY, USA (1997) 65–74
2. Pierrakos, D., Paliouras, G., Papatheodorou,C., and Spyropoulos, C. D.: Web Usage Mining as a Tool for Personalization: A Survey. User Modeling and User-Adapted Interaction, Vol. 13, No. 4. (2003) 311–372
3. Srivastava, J., Cooley R., Deshpande, M., and Tan, P.-N.: Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. ACM SIGKDD Explorations Newsletter, Vol. 1, Iss. 2. (2000) 12–23

Stacking Heterogeneous Data Resources for addressing the ECML-PKDD 2007 Discovery Challenge 1

Dimitrios Mavroeidis[1], Charis Brisagotis[1], Dimitris Drosos[1] and Michalis Vazirgiannis[1,2]

[1] Department of Informatics, Athens University of Economics and Business, Greece
[2] GEMO Team, INRIA/FUTURS, France

Abstract. In this paper we analyze and describe the approach undertaken by the DB-NET research group for addressing the ECML/PKDD Discovery Challenge 1. The Discovery Challenge was concerned with the construction of predictive models regarding several features of a user's browsing behavior. The training data consisted of attributes that described the users' browsing behavior (such as timestamp of visit and category-view paths) as well as information concerning their social and demographic profile (such as country, city, browser and operating system). The challenge that we faced in addressing this task, was the fact that the data resources, that could be employed for deriving the user behavior models, were heterogeneous. More precisely, for each user one could build predictive models based solely upon his personal browsing behavior, or incorporate features that described his social and demographic profile. In order to address this issue in a principled manner we have employed Stacking. Stacking was based on seven level-0 classifiers built upon seven heterogeneous datasets (constructed by the available training data). Consequently, for producing the user-specific models, we have compared the cross-validated accuracy of each level-0 and the level-1 classifiers (for each user), and finally selected the one that exhibited the highest accuracy estimate. The level-0 and level-1 classifiers were chosen after thorough experimentation to be C4 and Logistic Regression respectively. Apart from a detailed description and a discussion of the submitted model (which was solely for task 1 of the challenge), this paper contains our preliminary experimentations and experimental results for task 2.

1 Introduction

The ECML/PKDD discovery challenge (1) was concerned with the construction of predictive models regarding several features of a user's browsing behavior. The business potentials of such models are very broad as they can be used for building interactive marketing strategies and for content personalization (i.e. pointing to the content most desired by the users). In the context of challenge 1, the data were provided by Gemius (*www.gemius.pl*), an internet market research company, that collected its data, using cookies technology and special scripts placed on the monitored web-pages. The information that was accumulated consisted of the timestamp of a user's visit to the web-site, along with his category-view path (i.e. the pages of the web-site were pre-classified to a number of categories). Moreover, for each user, several social and demographic data

were collected that included his IP-based location (country, city, region), his browser and the operating system used. The data provided for the challenge were collected after one month of monitoring Polish web-sites. The training data file consisted of the first three weeks of the month, while the test data (where predictions should be made), consisted of the last week of the month.

A simple inspection of the training data reveals that there exist several users, whose browsing behavior is easy to model. These users consistently visit only one category and generate the same number of pageviews (for example *user ID 33*). On the other hand there exist several users that generate long complex visit paths, that cannot be easily modeled using solely the single user data (for example *user ID 5901007*). It is evident that for these users, more complex models should be employed that incorporate social and demographic features, such as the country of origin, operating system, etc. In order to address the aforementioned issues and to combine the supplied information in a principled manner, we have employed stacking, a machine learning approach for combining models that are derived from different resources. The stacking approach was initially proposed in [1], and has since attracted a significant amount of research in the field of machine learning (such as [2, 3, 1]. It has to be noted that in our approach we differentiate from the classical stacking approach, since we combine the output of a single classifier trained on different datasets instead of combining the outputs of different classifiers trained on the same dataset. Although, the issue of whether stacking classifiers improves on classification accuracy remains controversial [3], our results can be considered as further experimental evidence of the effectiveness of the stacking approach.

In our solution for the task 1 of challenge 1 we have constructed seven training dataset from the available data. These datasets were: single user data, users grouped according to their country, users grouped according to their operating system, users grouped according to their browsers, users grouped according to their country and operating system, users grouped according to their browser and operating system and users grouped according to their country and browser. Consequently, we have built seven level-0 classifiers using the C4 classifier, and then stacked their outputs using Logistic Regression. For producing the browsing model for each user, we have compared the cross-validation accuracy of each level-0 classifier and the level-1 classifier (for each user), and finally selected the one that exhibited the highest accuracy estimate.

The rest of the paper is organized as follows. Section 2 provides a short description of the Discovery Challenge. Section 3 presents the Data preprocessing strategies we have employed. Section 4 describes the learning algorithms used and presents the experimental results. Section 5 discusses the results and contains the concluding remarks.

2 Discovery Challenge Description

2.1 Task Description

The Discovery Challenge 1 consisted of 3 tasks. The first task aimed in constructing models that predict the length of a user's visit. The target attribute for task 1 could take two values: *short* and *long*, where *short* visits generated only one page-view, while

long visits generated more than one page-view. Such prediction models would allow a web-site to focus on users that generate longer sessions, thus saving resources. Task 2 of the challenge was concerned with the prediction of the sequence of categories a user is interested in. This would allow a web-site to devise targeted marketing strategies and personalize its content accordingly. The third task extended the second in that it required the prediction of the number of pages a user will generate for each category. In the context of the Discovery Challenge we have submitted a solution only for task 1. However, in this paper we extend our approach and present experimental results for task 2 as well.

The data provided for the challenge were collected after one month of monitoring Polish web-sites. The data contained information about 4882 users that generated circa 500000 session with the web-site. The training data file consisted of the first three weeks of the month (circa. 380000 session paths), while the test data (where predictions should be made), consisted of the last week of the month (circa. 140000 session paths). The final scores where calculated according to accuracy, in task 1 and according to certain score functions in task 2 and 3, that quantified the collectedness of the sequence of predicted categories.

2.2 Data Description

The training data given for the challenge described the browsing behavior of single users, as well as several of their social and demographic features. In order to illustrate the structure of the training data we provide an example of a single user's browsing data:

path id	*user id*	*timestamp*	*list of (category id, pageview number) pairs*			
27	1	1169814548	7,1	16,2	17,9	...

The features in this table are the *path id*, that is a unique number assigned to the cookie session generated by user with *user id=1*. The category-view path designates the sequence of categories the user has visited along with the number of pages he has viewed. In the case described on the table the user has visited 1 page belonging in category 7, 2 pages belonging in category 16, etc. Moreover, each cookie session is assigned with the respective timestamp, that designates the time when the session was generated. The rest of the features are related to the social and demographic characteristics of the users.

user id	*country id*	*region id*	*city id*	*system id*	*system sub id*	*browser id*	*browser ver. id*
10	42	11	44	3	9	1	517

From the structure of the data it can be observed that the Discovery Challenge 1, does not correspond to a typical data mining problem, where the data are provided in a standard vector based format. Thus, if one wishes to apply standard data mining approach, such as Support Vector Machines [4] and Decision Trees [5], it is necessary to perform certain data transformations. In the subsequent section we describe the transformations and their underlying assumptions, that we have employed for addressing this challenge. It has to be noted that there exist other approaches that do not require that the

data are transformed into a vector space format. These approaches are based on Relational Data Mining [6], Inductive Logic Programming [7], or Bayesian Networks [5]. It would be interesting to compare our results, to the results provided by such approaches.

3 Data Preprocessing

It can be easily observed that the main feature that should be employed for modeling the users' browsing behavior is the timestamp. The timestamp, also referred to as Unix time, is defined as the number of seconds that have elapsed since midnight UTC of January 1, 1970. In order to detect browsing patterns one needs to transform this variable into more meaningful features. In the proposed approach we have transformed the timestamp variable to a feature that described the day of the visit and a feature that contained the time of the visit (only the hours, not the minutes). It has to be noted that we have experimented with several other alternatives for deriving features from the timestamp. More precisely, we have attempted to introduce a boolean feature that described whether the time of the visit was a weekend or not. Moreover, concerning the time of the visit, we have experimented with several unsupervised (such as equal frequency) and supervised (such as maximization of Information Gain) interval generating approaches. Extensive experimental results have demonstrated that such transformations did not improve on the cross-validated accuracy.

Apart form the proposed solution for task 1, in this paper we present experimental results for task 2. In order to transform the training data into vector format for this task, we have considered only the top-three visited categories, as features. On the test data, the predictions of the categories of interest were based on the prediction of the superseding categories. It has to be noted that this presents a "Naive approach" that makes several simplifying assumptions concerning the data. It would be interesting to compare the results of this approach to more sophisticated Baysian Network or ILP approaches. Unfortunately, at the time of the writing of this paper, the results of the other contestants as well as the true labels for the test data were not known, so we can not compare our approach to the other contestants.

It is evident that in order to achieve optimal classification results, one has to employ accordingly the social and demographic features. In order to achieve this goal, we have created seven training datasets, that are based on the following combination of the available data.

1. single user data
2. users grouped according to their country
3. users grouped according to their operating system
4. users grouped according to their browsers
5. users grouped according to their country and operating system
6. users grouped according to their browser and operating system
7. users grouped according to their country and browser

The features that were contained in each of these datasets are the respective timestamp and category-view path. Each one of these datasets could be used to derive predictive

models for a user's browsing behavior. The baseline approach would be to build a separate model for each datasets and then select, at user level, the one that exhibits the highest accuracy estimate. In our proposed approach, we have taken one step further, and we have employed stacking in order to combine the outputs of the level-0 classifiers. The details of the learning algorithms employed are presented on the subsequent section.

4 Learning Algorithms

In this section we report the accuracy estimates of the proposed approach, as well as several other experiments we have attempted. This section is divided with respect to the type of classifier: Level-0 classifiers, refer to the classifiers applied directly on the seven training datasets and Level-1 classifiers refer to the classifiers that have been applied to the output of the Level-0 classifiers. In all results we report the average accuracy of the 10-fold cross validation, as well as the standard deviation observed through all the users (i.e. the standard deviation of the cross validation averages for each user).

4.1 Level-0 classifier

Concerning the Level-0 classifiers we have conducted extensive experiments using a Decision Tree classifier (C4) and the Naive Bayes classifier. In order to perform our experiments, we have used the WEKA Knowledge Explorer [8].

The subsequent table (table 1.) presents the averages and standard deviations of the 10-fold cross validation runs for Naive Bayes and C4. It has to be clarified that the standard deviation values, corresponds to the deviation of the average cross-validation accuracy over all the users. It can be observed that apart from the system model, that refers to the dataset produced by grouping the users according to the operating system, all the other models exhibit high values of standard deviation. Moreover, it can be observed that the Decision Tree results, though not statistically significant, are better that the Naive Bayes results. Based on these results we have selected C4 algorithm as the Level-0 classifier, in our submitted model.

Table 1. 10-fold cross validation results for Level-0 classifier for Task 1

Model	**Naive Bayes Accuracy (Std. Deviation)**	**C4 Accuracy (Std. Deviation)**
User Model	74.13% (15.65%)	75.91% (14.48%)
Country Model	81.22% (14.42%)	82.23% (13.72%)
System Model	69.92% (3.59%)	70.3% (4.12%)
Browser Model	75.66% (8.09%)	76.61% (8.34%)
Country-System Model	79.75% (14.81%)	80.71% (14.21%)
Country-Browser Model	79.82% (15.16%)	81.05% (14.04%)
System-Browser Model	76.16% (12.12%)	76.89% (12.13%)

Moreover, we have observed that out of the 4882 users, in the training data, 241 exhibited cross-validated accuracy higher that 98% for at least one of the aforementioned models, with the majority having operating system "1" (which presumably is Windows as it used by the vast majority of the users). Furthermore, 1391 users had at least 90% accuracy for at least one of the aforementioned models, with the majority coming from country 42 (which is presumably Poland, as the majority of users comes from this country).

Concerning task 2, where objective is to predict the top three categories a user is interested in, we present the classification accuracy for C4 and Naive Bayes. The results are reported for each category separately (tables 2,3, and 4). It to be noted that initial category predictions (first and second) are used in order to predict subsequent categories. The following tables summarize the results for all categories.

Table 2. 10-fold cross validation results for Level-0 classifier for Task 2, category 1

Model	**Naive Bayes Accuracy (Std. Deviation)**	**C4 Accuracy (Std. Deviation)**
User Model	70.72% (21.74%)	72.54% (20.67%)
Country Model	75.63% (21.53%)	76.81% (20.78%)
System Model	58.73% (13.23%)	60.27% (12.91%)
Browser Model	66% (21.8%)	67.56% (21.06%)
Country-System Model	75.56% (20.58%)	76.73% (19.94%)
System-Browser Model	65.89% (21.66%)	67.69% (20.6%)

Table 3. 10-fold cross validation results for Level-0 classifier for Task 2, category 2

Model	**Naive Bayes Accuracy (Std. Deviation)**	**C4 Accuracy (Std. Deviation)**
User Model	51.33% (30.11%)	50.4% (28.6%)
Country Model	60.54% (32.59%)	59.92% (30.86%)
System Model	60.53% (11.38%)	64.54% (12.38%)
Browser Model	55.16% (22.86%)	57.5% (23.48%)
Country-System Model	62.74% (30.95%)	61.07% (28.87%)
Country-Browser Model	56.99% (33.45%)	56.67% (32.09%)
System-Browser Model	58.98% (23.52%)	62.04% (22.63%)

4.2 Level-1 classifier

Concerning the Level-1 classifiers, we have experimented with Logistic Regression, Naive Bayes and Decision Tree learners. As input to our Level-1 classifiers, we have used the posterior probabilities of class-assignment that are derived by the Level-0

Table 4. 10-fold cross validation results for Level-0 classifier for Task 2, category 3

Model	**Naive Bayes Accuracy (Std. Deviation)**	**C4 Accuracy (Std. Deviation)**
User Model	62.22% (35.16%)	56.81% (32.38%)
Country Model	59.14% (37.43%)	58.35% (36.79%)
System Model	76.63% (11.07%)	77.61% (12%)
Browser Model	66.58% (27.48%)	68.17% (27.32%)
Country-System Model	62.28% (35.71%)	60.26% (34.64%)
Country-Browser Model	60.23% (37.97%)	60.36% (37.47%)
System-Browser Model	58% (33.77%)	59.48% (34.05%)

classifiers. Concerning task 1, the Naive Bayes classifier exhibited average cross validated accuracy of 79.76% and standard deviation 10.19%. On the other hand, for the same task, Logistic Regression, exhibited 83.87% average cross-validated accuracy with 8.74% standard deviation. Thus in our submitted results, we have selected the Logistic regression classifier.

Concerning task 2, due to time limitations we have experimented solely with the C4 algorithm, as a level-1 classifier. Its results concerning the first category were: 50.80% average cross validated accuracy and 37.74% standard deviation. For the second category, C4 exhibited 52.44% average cross validated accuracy with 31.14% standard deviation and for the third category, 73.27% average cross validated accuracy with 21% standard deviation

Consequently, for producing the user-specific models, we have compared the cross-validated accuracy of each level-0 and the level-1 classifiers (for each user), and finally selected the one that exhibited the highest accuracy estimate.

5 Conclusions

In conclusion, we should note that the Discovery Challenge 1, was a well formulated and organized challenge, that allowed Data Mining researchers and practitioners to apply their algorithms in a real world challenging problem. Unfortunately, at the time of the writing of this paper, the results of the contests were not made available, thus we cannot assess the competitive performance of our approach. Based on the accuracy estimates, our results serve as further experimental evidence, that stacking can improve on the effectiveness of learning algorithms. Given further time, we would attempt to utilize ILP and Relational Data Mining approaches that do not require that the data are transformed into a vector format.

References

1. Wolpert, D.: Stacked generalization. Neural Networks **5**(2) (1992) 241=260
2. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research **10** (1999) 271–289

3. Džeroski, S., Ženko, B.: Is combining classifiers with stacking better than selecting the best one? Mach. Learn. **54**(3) (2004) 255–273
4. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
5. Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press (2001)
6. Dzeroski, S., Lavrac, N., eds.: Relational Data Mining. Springer (2001)
7. Horváth, T., Yamamoto, A.: Special ilp mega-issue, foreword. Machine Learning **64**(1-3) (2006) 3–4
8. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)

Web Analyzing Traffic Challenge: Description and Results

Chedy Raïssi[1,2], Johan Brissaud[3],
Gérard Dray[2], Pascal Poncelet[2], Mathieu Roche[1], and Maguelonne Teisseire[1]

[1] LIRMM - UMR 5506, CNRS, Univ. Montpellier 2, France
[2] LGI2P, EMA Site EERIE Parc Scientifique Georges Besse, Nîmes, France
[3] BEE WARE Company, 210, Avenue Frederic Joliot, 13852 Aix-en-Provence, France
raissi@lirmm.fr,jbrissaud@bee-ware.net
{gerard.dray,pascal.poncelet}@ema.fr,{mroche,teisseire}@lirmm.fr

Abstract. This paper describes the Web Analyzing Traffic Challenge (Discovery Challenge of ECML/PKDD'07) and the results. Using the data from query logs it is possible to recognize an attack and define its class. Then the aim of this challenge is the filtering of attacks in Web traffic.

Key words: attack detection, classification.

1 Introduction

The number of computer attacks carried out grows in tandem with the web. According to the National Institute of Standards and Technology[4], American companies as early as 2004 suffered losses of up to 59.6 billion dollars following IT attacks. Considering the number of IT systems now deployed, intrusion detection is a significant research area for the purpose of assessing and forecasting system attacks as early as possible.

The OSI Model (*Open Systems Interconnection Basic Reference Model*) is usually represented by a diagram showing a column composed of stacked rectangular shapes, each one symbolizing a layer of the model. However, in reality the seventh layer is much wider and more diverse than the layers below it. This application layer is definitely the biggest, widest, and most complex of all. It contains more than just protocols and parameters, and is made up of languages, scripts, libraries and human concepts, etc. As a consequence, the OSI Model observed from a security perspective makes the diagram take on a reversed pyramid shape. So the higher the layers, the richer and more diverse is their content, which means they are also more complex to secure.

Trying to filter application traffic as diverse and dynamic as Web traffic can quickly bring awareness of the existence of several strong constraints and the necessity to fulfil specific requirements such as:

[4] http://www.nist.gov.

– **Unknown attack detection**: A major consequence of application diversity is that the potential for vulnerabilities is infinite. Experience has already revealed that the vast majority of application attacks consist in the unknown variety.
– **False positives**: Considering the richness and diversity of this domain, and seeing that the threshold of acceptance is user-dependent, avoiding and eliminating False Positives are critical issues when analyzing the application layer.
– **Ambiguous queries**: When looking at existing applications it becomes quickly obvious that they harbour weaknesses or vulnerabilities. Traffic addressing these resources will then appear to carry weaknesses, but cannot be blocked without stopping the application.
– **Abnormal behaviour detection**: Attacks are not the only danger prevalent. Securing Web traffic is a more complex task than mere intrusion prevention. There are various other types of requests that require supervision.

2 Main objectives

The issue being addressed by this challenge is the filtering of application attacks in Web traffic. This is a complex matter because of diversity in attack purposes and means, the quantity of data involved and technological shifts. Application attacks can be multi-class and undergo constant change. They do however maintain some distinguishing features (escaping, bypassing, keywords matching external entities, etc.).

To achieve this aim data sources available from HTTP query logs are used. Using this data we can not only recognize an attack but also define which class it belongs to. Participants would have to start with an HTTP query in context and deduce which class it belongs to and what is its level of relevance.

To efficiently address this issue, we divided the challenge in the two following tasks:

1. **Task 1: Multi-class and contextual classification**
   We have to be able to classify queries that may belong to different classes, and we have to do so according to context. A query in attack form that is not dangerous because made in the wrong context has to be properly labelled. The amount of data to process being considerable due to traffic density, any real-world classification application should be able to process the queries extremely quickly. Participants are judged on the classification performance but also on the time performance of their algorithm implementations.

2. **Task 2: Isolation of the attack pattern**
   We should be able to pinpoint in an attack query the shortest chain that conveys the attack[5].

[5] In this paper, this task will not be developed.

3 Dataset composition

The attacks of the dataset look like real attacks but have no chance to succeed because they are constructed blindly and do not target the correct entities. One sample can eventually target several classes (SQL injection, Command execution, etc.). Each example is totally independent of the others.

The data set are defined in XML (portable and standard format). Each sample is identified by a unique id, and contains the three following major parts: *Context* (stands for the environment in which the query is run), *Class* (describes how this sample is classified by an expert) and the *description* of the query itself.

- **Context:** It contains the following attributes:
  1. Operating system running on the Web Server (UNIX, WINDOWS, UNKNOWN).
  2. HTTP Server targeted by the request (APACHE, MIIS, UNKNOWN).
  3. Is the XPATH technology understood by the server? (TRUE, FALSE, UNKNOWN)
  4. Is there an LDAP database on the Web Server? (TRUE, FALSE, UNKNOWN)
  5. Is there an SQL database on the Web Server? (TRUE, FALSE, UNKNOWN)

- **Class:** It lists the different subdivision levels of HTTP query categorization (and how they are represented in the context part of the dataset).
  The "type" element indicates which class this request belongs to:
  1. Normal Query (Valid)
  2. Cross-Site Scripting (XSS)
  3. SQL Injection (SqlInjection)
  4. LDAP Injection (LdapInjection)
  5. XPATH Injection (XPathInjection)
  6. Path Traversal (PathTransversal)
  7. Command Execution (OsCommanding)
  8. SSI Attacks (SSI)

  Moreover, a flag is added explaining whether a query is within the assigned context or not (element "inContext" taking two values: TRUE or FALSE).

  Another element ("attackIntervall") indicates where the attack is located on the query description. This element begins with the name of the element where the attack is located (uri, query, body, header) followed by ":". Thereafter the interval considered as an attack is specified. For headers, we also indicate the header name where the attack is located. The interval begins from the beginning of the considered header value.

– **Query:** It is described with its different components:
  1. Method
  2. Protocol
  3. Uri
  4. Query
  5. Headers
  6. Body

4 Evaluation Criterion

For this challenge, precision and recall (see formulae 1 and 2) are the basic measures used in evaluating search strategies.

$$Precision = \frac{\text{number of relevant attacks detected}}{\text{number of attacks detected}} \tag{1}$$

$$Recall = \frac{\text{number of relevant attacks detected}}{\text{number of relevant attacks}} \tag{2}$$

$Fmeasure$ combines recall and precision in a single efficiency measure (see formula 3).

$$Fmeasure(\beta) = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall} \tag{3}$$

For the challenge, $Fmeasure$ is calculated with $\beta = 1$, meaning that the same weight is given to precision and recall.

5 Results

Only two challengers sent a submission for this challenge. A discussion to explain this number of participants will be developed in section 6.

The evaluation of the challengers is based on the "*test*" dataset available at the end of June 2007. The "*learning*" dataset was available on April 15^{th}, 2007. The description of the different datasets is given in the table 1.

The table 2 presents if the attacks are correctly detected. The challenger 1 provides the best result based on the $Fmeasure$[6]. For the two challengers the precision is better than the recall. Let us note that the $Fmeasure$ obtained

[6] A new submission of the challenger 1 has be sent when the deadline was passed. This new submission (non-official submission) gives again a better result with a $Fmeasure$ at 0.9345.

	Test dataset	Learning dataset
Number of examples	70,000	50,000
Ratio of attacks	60%	70%
Ratio of attacks in context	75%	85%
Type of Attacks		
post	15%	12%
get	58%	70%
cookie en post	7%	4%
header en post	5%	3%
cookie en get	8%	6%
header en get	7%	5%

Table 1. Description of the two datasets used for the challenge

without context is low for the submission of the two challengers: 0.4826 for the challenger 1[7] and 0.1728 for the challenger 2.

The table 3 presents if the classes of attacks are correctly detected by the challengers. This table shows that the challenger 1 obtained the highest values of $Fmeasure$ for all the classes of attacks. The results show that some types of attacks are quite easy to detect (*e.g.* XSS, XPathInjection, LdapInjection), while other ones are very difficult to find out (*e.g.* OsCommanding, SSI)[8]. We have to note that, for example, an instance of the SSI class is difficult to detect since this type of attack is usually a multi-class attack.

The 'Valid' class (*i.e.* normal queries) was easy to detect for the two challengers: $Fmeasure$ at 0.8793 for the challenger 1 and 0.6900 for the challenger 2.

Finally, we can observe that multi-class attacks were not considered by all the challengers.

	$Precision$	$Recall$	$Fmeasure$
Challenger 1: LIFO Orléans, France	0.8229	0.7807	0.8012
Challenger 2: Department of Informatics Athens, Greece	0.4976	0.4721	0.4845

Table 2. The $Fmeasure$ of the challengers

[7] However the value of the $Fmeasure$ with the non-official submission of the challenger 1 is excellent: 0.9824.

[8] These conclusions are based on the most significant results of the challenger 1.

Group	Challenger 1	Challenger 2
Valid	0.8793	0.6900
OsCommanding	0.4093	0.0598
SSI	0.4216	0.0307
SqlInjection	0.6205	0.0358
PathTransversal	0.6819	0.0409
XSS	0.7597	0.0394
XPathInjection	0.7405	0.0487
LdapInjection	0.8811	0.0281

Table 3. The *Fmeasure* of the classes of attacks

6 Conclusion

As we have said in the introduction section, the problem of detecting intrusion in Web traffic is far away from trivial. This contest aims at providing different approaches for extracting such intrusions.

The contest is now finished and we have two observations. At the very beginning of the contest, 25 researchers from different countries such as China, Finland, Indonesia, Korea, Australia, Pakistan, Italy would like to apply their techniques (usually classification techniques) for learning intrusions. At the end, only two challengers sent their results and we would like to acknowledge them. We have asked the other challengers to know why they did not submit their results. The response was mainly they did not have a lot of knowledge about detection intrusion and when they tried to apply "traditional approaches" in order to characterize intrusion, their results were not good enough. We believe that this observation is very important in a data mining context since more and more we must integrate the expert earlier in the knowledge discovery process. The second observation is related to the problem of detection intrusion itself. When we proposed the contest, we knew that this problem was a very difficult topic but thanks to the results of the challengers it is clear now that detection intrusion becomes more and more a new hot topic since we have to deal not only with supervised, unsupervised classification but also with real time data mining.

ECML/PKDD Challenge: Analyzing Web Traffic A Boundaries Signature Approach

Matthieu EXBRAYAT

Laboratoire d'Informatique Fondamentale d'Orléans
Université d'Orléans, B.P. 6759
45067 ORLEANS Cedex 2, France
Matthieu.Exbrayat@univ-orleans.fr

Abstract. To detect HTTP attacks (and the corresponding interval) we propose a signature detection method which concentrates on both ends of attack patterns. We also use a textual word / symbolic word model to describe attack patterns.

1 Introduction

1.1 A brief overview of the "Analyzing web traffic" challenge

The ECML-PKDD 2007's "Analyzing web traffic" challenge consists in determining whether a given HTTP request contains attack(s) and whether these attacks would succeed depending on the server's context (OS, ...). On a second step, the attack interval must be accurately determined (± n characters, n=3). The learning dataset (xml) consists of samples that are structured as follows:

```
<sample id="37306">
  <reqContext>
    <os>WINDOWS</os>
    <webserver>UNKNOWN</webserver>
    <runningLdap>TRUE</runningLdap>
    <runningSqlDb>TRUE</runningSqlDb>
    <runningXpath>FALSE</runningXpath>
  </reqContext>
  <class>
    <type>LdapInjection</type>
    <inContext>TRUE</inContext>
    <attackIntervall>headers:User-Agent:1-30</attackIntervall>
  </class>
  <request>
    <method>GET</method>
    ...
    <headers><![CDATA[Host: www.iEPL.ch:80
    ...
    User-Agent: mtn%29%28+%7C++++%28Ahk%3D*%29
```

```
      ...
      ]]></headers>
    </request>
  </sample>
```

Each sample is identified by a unique id.
The *reqContext* element describes the execution context of the destination server. This context consists of five elements, each of these element taking one of three possible values: two values correspond to known values (e.g. os="WINDOWS" or "UNIX"); the third one is "UNKNOWN" (e.g. *Webserver* in this sample).
The *class* element gives the type of attack (*type*), indicates where the attack is located in the request (*attackIntervall*), and if it will succeed depending on the execution context (*inContext*).
The content of the request (*request*, which has been simplified here) constitutes the third part of the sample. In this sample, the attack is located in the field *User-Agent* of the *headers* element. We can also see that data is URL-encoded [1].

1.2 Intrusion detection and machine learning

Two main attack detection approaches [2] have been developed and implemented in intrusion detection systems (IDS) during the past two decades. The first kind of approach, called signature detection, consists in looking for some (sequences of) bytes that are typical for a given attack. The second kind of approach, called anomaly detection, consists in determining whether a given request is "similar" or not to valid ones, this similarity being computed in an adequate feature space. These two approaches are of general use and can be applied to a wide range of attacks, including HTTP attacks. Signature detection can be implemented in a very effective manner, thus limiting the detection overhead. Meanwhile, a new kind of attack, whose signature is not known by the system, will not be detected. Anomaly detection systems will be more likely to detect such new attacks, but as a counterpart will also raise false alarms (and might also be more time consuming that signature detection).

Intrusion detection can be hard-coded or can rely on learning machine methods. Concerning hard-coded (commercial) IDSs, we can cite several well known tools, such as the general purpose network IDS SNORT [3] and the Apache module mod_security [4]. These tools are based on signature detection and use hand-crafted rules (the length or which can vary).

Both signature and anomaly detection have been implemented with learning machine approaches. It is quite straightforward that signature detection is related to supervised learning techniques (e.g. neural networks [5], svms [6]) while anomaly detection corresponds to unsupervised methods (i.e. clustering [7]).

1.3 Proposed approach

Considering the "analyzing web traffic" challenge, we are clearly in the context of signature detection and thus of supervised learning. Nevertheless, current

machine-learning signature detection methods use to determine minimal sets of bytes rather than complete attack patterns, the main reason being that detecting an attack (and its *global* location) is at first glance more important than determining precisely the boundaries of the attack pattern, this last point remaining an issue. We thus propose a different signature detection method, which concentrates on attack pattern boundaries.

Our approach focuses exclusively on both ends of the attack interval. The basic assumption is that, if these ends can be accurately determined, then they should be of a rather generic form for a given type of attack. From another point of view, attacks consist of some kind of code (scripts, path, etc.). They thus respect a given structure.

We first define what an HTTP attack pattern consists of. We can easily see that such a pattern consists of a sequence of terms, some of them being words (alphanumerical sequences), some of them being symbols (parentheses, dots,...). Some words appear to be script keywords, some other clearly represents variables. As a consequence, we propose to define an attack pattern as a sequence of these three kinds of elements (keywords, variables, symbols). We thus have to learn attack rules from such sequences. We propose a rule model which concentrates on the few first and last elements of an attack sequence. A group of sequences that share the same l first and last terms (l being small) are represented by a unique rule. A candidate request which contains a sequence beginning and ending like a given rule is considered as holding an attack, and the attack interval is directly identified. While relatively naive, this approach appears to be efficient through cross validation on the learning dataset.

In section 2 we while define more formally the underlying concepts. In section 3 we will present the learning and classification algorithms. In section 4 we will present various tests conducted on the learning dataset. We will finally sum up and propose some perspectives in section 5.

2 Definitions

2.1 Language model

Two kinds of language models as currently used by IDSs to describe the content (the general term being *payload*) of incoming data: n-grams and word models. N-grams [8] check whether the payload contains some sequences of n bytes (n being fixed) that constitute an attack signature. They can be used both with ASCII and binary payloads. Nevertheless, they do not fit our approach, as we concentrate on successive terms of variable width. Word models define words as sequences of symbols bounded by specific symbols (delimiters). Formally, with a given alphabet Σ, a word w is such that $w \in L = (\Sigma \setminus D)^*$, where D is the set of delimiters.

Our model is derived from the word model. We use two kinds of words which alternate in the sequence rather than a single kind of words separated by delimiters. In other words, we consider that both $(\Sigma \setminus D)$ and D produce words. We

will further call the first kind of words *textual words* (tw) and the second one *symbolic words* (sw). Let Σ the alphabet, Θ the set of alphanumerical characters, and Δ the set of remaining symbols.

$$\Theta = \{'a'..'z'\} \cup \{'A'..'Z'\} \cup \{'0'..'9'\} \Delta = (\Sigma \setminus \Theta) tw \in \Theta^* sw \in \Delta^*$$

2.2 Subtyping and refining the language model

From tw and sw we derive three types of words that will be used in our method.
Textual words are split into two groups : keywords ($kw \in K$) and variables ($vw \in V$), with $K \cup V = \Theta^*$. Whether a text word is a keyword or a variable is defined automatically, depending on the global frequency of the word within attack patterns. Let τ a frequency threshold, let f the global frequency (within attack patterns) of $tw \in \Theta^*$,

$$tw \in \begin{cases} K \text{ if } f \geq \tau \\ V \text{ otherwise} \end{cases}$$

Determining keywords according to their frequency is relevant: as learning requests are *a priori* not correlated, variable names are random; their frequency is thus low. In addition to frequency, we also consider that a keyword must consist of at least two characters (which is a relatively standard case).
Symbolic words are not subtyped, but their content is refined: some characters (sub-delimiters) are discarded. Practically speaking, we discard the space character (or "+", the equivalent character in url-encoded strings), which, as a string delimiter, is meaningless within sw. For the sake of simplicity, we will keep sw as the name for simplified symbolic words.
In the remaining, w denotes a word belonging to any of these three categories.

2.3 Words similarity

Two words w_1 and w_2 of type keyword or symbol are considered similar if they have the same value. Two words w_1 and w_2 of type variable are considered similar whatever their values are:

$$w_1 \sim w_2 \Leftrightarrow \begin{array}{r} (w_1 \in K \ \wedge w_2 \in K \ \wedge w_1 = w_2) \\ \vee \ (w_1 \in \Theta^* \wedge w_2 \in \Theta^* \wedge w_1 = w_2) \\ \vee \ (w_1 \in V \ \wedge w_2 \in V) \end{array}$$

2.4 Sequences l-similarity

Two sequences of words $S_1 = \{E_{1_1}, E_{1_2}, \ldots E_{1_m}\}$ and $S_2 = \{E_{2_1}, E_{2_2}, \ldots E_{2_n}\}$ are considered l-similar iif their l first and l last elements are similar. l is called their "similarity level".

$$S_1 \sim_l S_2 \Leftrightarrow \forall i \in \{1..l\}(E_{1_i} \sim E_{2_i}) \wedge (E_{1_{m-i+1}} \sim E_{2_{n-i+1}})$$

2.5 Raw attack patterns

A raw attack pattern RAP describes the attack pattern found in a given request:

$$RAP_i = \{S_i, A_i^*, As_i, Cx_i, inCx_i\}$$

- S_i is a sequence as described above, which corresponds to the attack interval;
- A_i^* denotes a set of attack types (a request can contain several attacks);
- Ai_i denotes the attack site (i.e. the attack interval, except than the indices of both ends are not indicated) Some requests (ab. 5 %) contain several (two) attack sites. Our current implementation does only handle the first one;
- Cx_i denotes the execution context of the request;
- $inCx_i$ indicates whether the attack is in context or not.

For instance, the sample in section 1.1 would generate the following RAP:

- S={"mtn" ; ")(—(" ; "Ahk" ; "=*)"}
- A^*={"LdapInjection"}
- As={"headers:User-Agent"}
- Cx={"Windows" ; "Unknown" ; "True" ; "True" ; "False"}
- $inCx$=True

The url-encoded attack pattern "mtn%29%28+%7C+++%28Ahk%3D*%29" is decoded as: "mtn)(— (Ahk=*)", which produces the above attack sequence. We can see that symbolic words are refined (spaces are discarded). In this example we will suppose that mtn is a keyword and Ahk a variable.

2.6 Raw Attack Pattern l-similarity

Two raw patterns RAP_1 and RAP_2 are l-similar iif their attack sequences are l-similar and their type(s) of attack are alike:

$$RAP_1 \sim_l RAP_2 \Leftrightarrow (S_1 \sim_l S_2) \wedge (A_1^* = A_2^*)$$

2.7 Rules

Our learning algorithm generates rules, each rule representing a group of l-similar raw attack patterns (which we will call the *underlying RAPs*). These rules are then used to classify candidate requests. A rule R_i consists of:
$R_i = \{S_i, l_i, A_i^*, As_i^*, SCx_i, K_i^*, f_i\}$

- S_i is a sequence of length $2 \times l$, composed of the l first and last elements of the attack sequences of the underlying RAPs (which are similar in the underlying sequences, according to the l-similarity definition). The value of "variable" words is meaningless.
- l_i indicates the level of similarity between the underlying RAPs.
- A_i^* contains the attack types of the underlying patterns (which are identical according to the definition of RAP l-similarity).

- The set of attack sites As_i^* corresponds to the list of sites (e.g. "query") where the underlying RAPs occur.
- The synthetic execution context SCx_i is built according to the execution contexts of the *valid* underlying RAPs (see section 2.8).
- We also set a list K_i^* of significant keywords (i.e. keywords which can be found within all the underlying RAPs).
- f_i is the frequency of R_i, i.e. the number of underlying RAPs.

For instance, the RAP generated by our sample might group with twenty other patterns, with a level of similarity l=2, to form the following rule:

- S={"mtn" ; ")(—(" ; "rfjf" ; "=*)"} ("Ahk" is a variable)
- l=2
- A^*={"LdapInjection"}
- As={"headers" ; "query" ; "uri"}
- SCx={"Windows" ; "Unconcerned" ; "True" ; "Unconcerned" ; "False"}
- K^*={"mtn"}
- f=20

2.8 Synthetic Execution Context

A Synthetic execution context SCx generalizes the execution contexts of the underlying *in-context* RAPs of a given rule, in order to determine which elements of this context are relevant w.r.t. the underlying attack. Let SCx a synthetic context representing n contexts $\{Cx_1, Cx_2 \dots Cx_n\}$; let SCx_j being the j^{th} element of SCx and Cx_{i_j} being the j^{th} element of Cx_i, $Cx_{i_j} \in \{V_1, V_2, V_{unknown}\}$.

$$SCx_j = \begin{cases} V_1 & \text{if } \forall i \in \{1..n\} Cx_{i_j} \in \{V_1, V_{unknown}\} \\ V_2 & \text{if } \forall i \in \{1..n\} Cx_{i_j} \in \{V_2, V_{unknown}\} \\ V_{unconcerned} & \text{otherwise} \end{cases}$$

For instance, if on some contexts, OS is set to "Windows", and on some others it is set to "Unix", then OS is set to "unconcerned" in the synthetic context. If OS is either "Unix" or "Unknown", but is never "Windows", then OS is set to "Unix" in the synthetic context. We can notice that this synthetic context can only be considered as meaningful if $n > 1$ (or even $n >> 1$).

3 Learning and classification algorithms

3.1 Building rules

As a pre-treatment, raw attack patterns are extracted from the learning dataset as exposed in section 2.5. Rules are then produced as follows: we start with the lowest value of l, and group all l-similar patterns into candidate rules. We then check whether these rules are also present in valid requests: a candidate rule must be more frequent in attacks than in valid requests; if not, it is rejected; otherwise, the candidate rule is kept and its underlying RAPs are removed from the list of patterns. We stop when no more rule can be produced.

```
Function Learn(in: src, out: rules)
  // src = the list of raw attack patterns
  // rules = the list of rules generated
  Begin
    rules <- empty
    l=1;
    Repeat
      l <- l+1                                                    (1)
      // Generate candidate rules for level l
      candidate_rules <-empty
      Foreach raw pattern ra in src
        If (a rule ru in candidate_rule is l_similar to ra) Then
           increment the frequency of the coresponding rule ru
        Else
          generate a new rule ru' according to ra
            (set its frequency to 1)
          candidate_rules <- candidate_rules U ru'
        Endif
      Endforeach
      // Check if candidate rules are relevant
      Foreach rule ru in candidate_rules
        If (ru is more frequent in attacks                        (2)
               than in valid requests) Then
          rules <- rules U ru
          Foreach raw pattern ra represented by ru
            src <- src \ ra
          Endforeach
        Endif
      Endforeach
    Until (MaxLength (src) < l*2) or (src is empty)               (3)
  End.
```

We can see that this algorithm starts with l=2 `(1)`. Using l=1 would produce rules that do not discriminate enough (tests confirm poor scores for l=1).
MaxLength `(3)` indicates the length (in term of elements) of the longest raw pattern remaining in src. This value is used to stop the main loop, when l becomes so big that taking the l first and last elements of any remaining pattern amounts to consider this whole pattern. If such patterns did not produce a rule until there, neither will they in further loops.

3.2 Request classification

For each candidate request, we check if it matches one or more rules, among which we choose the right one according to a score function.

```
Function Classify(in: req, in: rules, out: res)
```

```
// req : the request to check
// rules : the attack rules to look for
// res : the classification result(s)
Begin
  cand_rules <- empty
  Foreach field of req
    Foreach rule ru in rules
      If field matches ru Then                              (1)
        cand_rules <- cand_rules U ru
      Endif
    Endforeach
  Endforeach
  If cand_rules is empty Then  // req is classified as valid
    response <- '<req.id,1,1>'
    res <- res U response
  Else // attack detected
    bestrule <- Best(cand_rules)                            (2)
    incontext <- InCtx(req,bestrule)                        (3)
    interval <- Inter(req,bestrule)                         (4)
    Foreach attacktype in bestrule                          (5)
      response <- '<req.id,attacktype,incontext,interval>'
      res <- res U response
    Endforeach
  Endif
End.
```

As a pre-treatment of `(1)`, each field of *req* (uri, headers, etc.) is decomposed in a candidate sequence according to our language model, and attacks are search in each field iteratively. An attack is found (rule matching) if a sub-sequence of the candidate sequence is l-similar to the attack sequence of the rule. Best `(2)` determines the best rule amongst candidate ones (currently: the one having the highest frequency). InCtx `(3)` determines whether the attack is in context or not (see 2.8). Inter `(4)` appears at this point of the descriptive algorithm, but is in fact implemented during the checking phase `(1)`. Finally if a rule corresponds to several types of attack, a response is generated for each type `(5)`.

Technically speaking, rule matching `(1)` is not executed in a completely iterative way: rules are ordered according to their (decreasing) frequency. For each field of the candidate request, the search stops at the first matching rule (i.e. the one with the best score for this field). The number of candidate rules is thus bounded by the number of fields. If a rule contains significant keywords (see 2.7), we first search for them (in fact only the first one): the rule is considered not applicable if this word is not present in the field checked.

With N candidate requests, this algorithm's complexity is O(N * number of fields (or field's attributes) * number of rules). Meanwhile, we must notice that i) we choose to focus mainly on the language and rule models; ii) this upper bound is rarely achieved; iii) we propose several ways to drop it down in section 5.

4 Results

4.1 Reminder of the score functions

Two types of scores are used. The first type, which we call *classic*, is based on the standard precision and recall measures:

$$precision = \frac{number\ of\ relevant\ attacks\ detected}{number\ of\ attacks\ detected}$$

$$recall = \frac{number\ of\ relevant\ attacks\ detected}{number\ of\ relevant\ attacks}$$

$$Fmeasure = \frac{2 \times precision \times recall}{precision + recall}$$

The second type, called *fuzzy*, is of the same shape, except that an attack is considered relevant if the interval is determined $\pm$ 3 characters.
We consider a detected attack to be relevant if it is detected of the good type. In this section we will distinguish between in-context and out-of-context attacks.

4.2 Influence of τ

This test has been conducted over a sample of the learning dataset (50 %), from which we randomly generated five distributions of 70 % learn dataset / 30 % test dataset. The results correspond to an average value of the five runs. We ran tests with $0.1\% \leq \tau \leq 10\%$.

Fig. 1 corresponds to the classic *scores (precision, recall, and Fmeasure) for in-context attacks*. Scores are raising with τ until a peak (Fmeasure $\approx$ 0.96, precision $\approx$ 98%, recall $\approx$ 93%) for $\tau \approx 0.6\%$. Fmeasure then decreases for $1\% \leq \tau \leq 2\%$, is then stable until $\tau = 6\%$, and finally decreases until 0.84. It thus remain quite stable for higher values of τ (further tests show a little gap for $\tau \approx 35\%$, where Fmeasure reaches 0.80). Precision is always better than recall for in-context attacks. This is mainly due to our approach, which, just like other signature detection methods, favours formerly well identified attack patterns.

We can see that low values of τ have a greater influence on recall than on precision. This and the global shape of the score can be explained by the influence of τ on the number of rules produced (see fig. 2), which decreases rapidly until $\tau \approx 0.6\%$. This is due to the fact that the number of keywords decreases as τ growths. With small values of τ, many variables are considered as keywords, and many rules are produced that only differ on false keywords, This results in some overfitting from which recall suffers, while precision is not affected. On the contrary, with high values of τ, many keywords are considered like variables, more attack sequences are thus considered as l-similar, and the number of rules decreases. As rules become more general, their ability to discriminate diminishes, and both precision and recall decrease, precision being the most affected: the number of attacks detected increases slightly while the number of *relevant* attacks detected decreases. This last point is due to the fact that an attack is

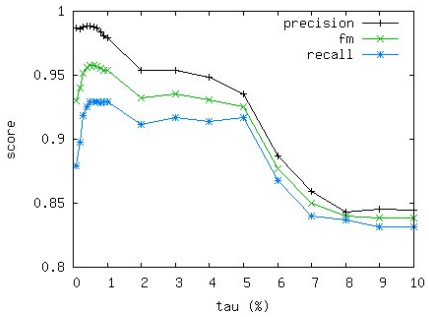


Fig. 1. Fmeasure score for in-context attacks

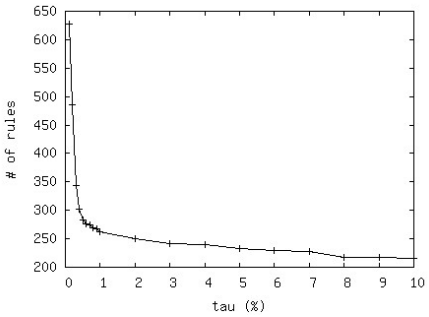


Fig. 2. Number of rules produced

considered relevant if it is detected of the good type: with less accurate rules, the classification algorithm is more likely to find several types of attack for a given request and to choose the wrong one.
The inflection point of the number of rules corresponds to the highest scores observed in the previous figure. Each learning subset contains about 5300 attack requests. With $\tau = 0.6\%$, the learning phase produces 277 rules (average value), which means that a rule represents, on the average, 19 attack requests. This reduction factor is still limited, and several ways to raise it will be proposed in section 5. We can notice that further tests have shown that the number of rules becomes stable (212 on average) for $\tau \geq 35\%$.

Concerning *fuzzy scores for in-context attacks*, fig. 3 shows that curves have the same shape as classic scores, except that high values of τ have a higher influence on the score (dropping down from 0.81 with $\tau = 0.6\%$ to 0.62 for $\tau = 10\%$: the lack of accuracy of rules strongly affects the determination of exact boundaries. We must notice that our fuzzy scores are slightly over-evaluated: our evaluation algorithm does only control the first attack interval of each request, while some of them (ab. 5%) do contain two attack intervalls.

Fig. 4 presents the time elapsed during classification (implemented in Java), depending on τ. Once again, as the complexity of the current classification algorithm is related to the number of rules, we can see an inflection for $\tau \approx 0.6\%$. With greater values of τ, this time remains stable around 18 seconds.

Fig. 5 and 6 respectively correspond to the *classic and fuzzy scores for out-of-context attacks*. The global shape of these curves is similar to the ones for in-context attacks, presenting a peak for $\tau \approx 0.6\%$. We can anyway notice that this peak is much lower than for in-context attacks. We can also notice that recall is higher than precision. We think that this mainly comes from the way the "in-context" property is determined. For rules that represent a small number of attacks, the synthetic context might be unreliable, and some out-of-context attacks may be classified as in-context ones. As a consequence the number of attacks detected (and by the way of relevant attacks detected) is affected. This explains both the lower scores and the fact that recall is higher than precision.

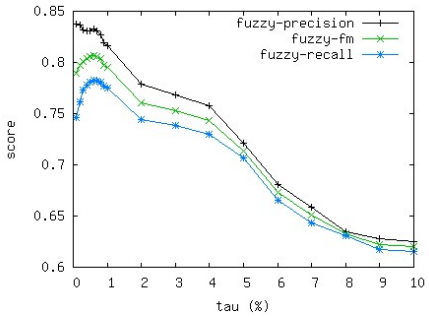


Fig. 3. Fuzzy-Fmeasure score for in-context attacks

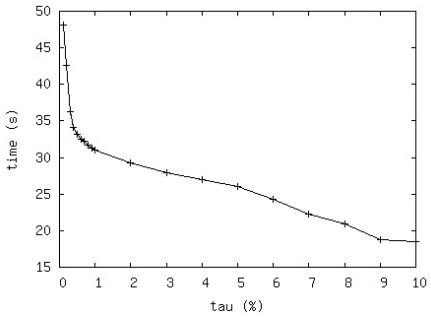


Fig. 4. Time elapsed (test phase)

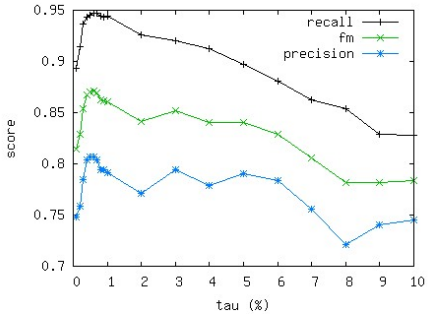


Fig. 5. Fmeasure score for out-of-context attacks

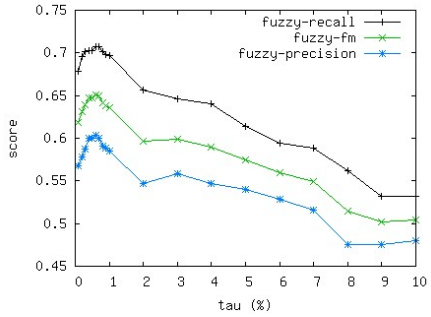


Fig. 6. Fuzzy-Fmeasure score for out-of-context attacks

4.3 Cross validation

We finally ran a 90/10 cross validation on the whole learning dataset. As we could expect, results (see table 1) are similar to the ones observed with the 70/30 runs. Learning with 90 % of the test dataset, we obtain an average of *309 rules* for 13400 attacks, having thus a reduction factor of 43, which is much better than in section 4.1. This confirms that the rules produced are relevant and representative for a large enough learning dataset.

5 Summary

We have proposed a signature detection method which indicates the boundaries of the attack interval within an HTTP request. Our method is based on a language model using three kinds of elements : keywords, variable names and symbolic words. We induce rules that describe short sequences of elements constituting the beginning and ends of a given attack pattern. The java implementation of this approach achieves satisfying performance on the learning dataset.

	precision	recall	Fmeasure
Classic, in-context attacks	98,8%	97,0%	0.979
Fuzzy, in-context attacks	83,4%	81,9%	0.826
Classic, out-of-context attacks	94,2%	96,0%	0.951
Fuzzy, out-of-context attacks	72,5%	74,0%	0.732

Table 1. Cross validation (90/10) with $\tau = 0.6\%$

Several ways could be explored to raise scores and to speed up execution.
First, we could drop down time complexity: instead of checking rules iteratively, we could index rules according to their symbolic words and keywords, and then slide along the request only once, checking whether each incoming word belongs to this index. We could even adapt the Aho-Corasick algorithm [9].
Second, several kinds of attacks (e.g. path traversal) contain repetitive sub patterns (e.g. "../../.."), the length of which is varying among attacks. We could detect such repetitions and adapt the syntax of refined symbolic words in order to handle them properly. While less rules would be produced, their accuracy would also raise and more attacks would be detected.
Last, some meta-level (from the point of view of our language model) attacks are not handled properly, such as multiple unicode encoding [10]. We could specifically search (but may be aside from our main approach) for such anomalies.

References

1. Berners-Lee, T.: RFC 3986 : Uniform Resource Identifier (URI): Generic Syntax. Internet Society (2005)
2. Axelsson, S.: Research in intrusion detection systems: A survey. Technical Report 98-17, Chalmers University of Technology (1999)
3. Beale, J.: Snort 2.1 Intrusion Detection, Second Edition. Syngress (2004)
4. Ristic, I.: Apache Security. O'Reilly (2005)
5. Ghosh, A., Schwartzbard, A.: A study in using neural networks for anomaly and misuse detection. In: Proc. of the 8th USENIX Security Symposium, Washington D.C. (1999)
6. Laskov, P., Schäfer, C., Kotenko, I.V.: Intrusion detection in unlabeled data with quarter-sphere support vector machines. In: DIMVA. (2004) 71–82
7. Portnoy, L., Eskin, E., Stolfo, S.: Intrusion detection with unlabeled data using clustering. In: Proc. of ACM CSS Workshop on Data Mining Applied to Security. (2001)
8. Marceau, C.: Characterizing the behavior of a program using multiple-length n-grams. In: Proc. of New Security Paradigms Workshop. (2000) 101–110
9. Aho, A.V., Corasick, M.J.: Efficient string matching: An aid to bibliographic search. Communications of the ACM **18**(6) (1975) 333–340
10. Microsoft: Ms00-78 - web server folder traversal vulnerability. Microsoft Sec. Bull. (2000)

Feature Extraction from Web Traffic data for the Application of Data Mining Algorithms in Attack Identification

Konstantinos Pachopoulos[1], Dialekti Valsamou[1], Dimitrios Mavroeidis[1] and Michalis Vazirgiannis[1,2]

[1] Department of Informatics, Athens University of Economics and Business, Greece
[2] GEMO Team, INRIA/FUTURS, France

Abstract. In this paper we present and discuss the approach undertaken by the DB-NET research group for addressing the ECML/PKDD Discovery Challenge 2. The challenge was concerned with the analysis of web traffic data with the aim of constructing predictive models that can identify possible future attacks. The training data provided for the challenge consisted of a collection of pre-classified traffic data into 8 categories; one containing the valid (non-malicious communications), while the other 7 contained several types of web attacks. The attack-types were: Cross-Site Scripting, SQL Injection, LDAP Injection, XPATH Injection, Path traversal, Command execution and SSI. A challenge that we faced stemmed from the fact that the training data were provided in a preliminary HTTP protocol format, containing string representations of the HTTP packet fields (such as method, protocol, and uri). This information could not be directly incorporated in standard data mining algorithms, and significant preprocessing should be performed. In order to address this challenge we have identified several string patterns that could signify a malicious communication, and transformed the unstructured information to feature-vector format. This transformation allowed us to employ C4, a decision tree algorithm that exhibited an estimated accuracy of 77%.

1 Introduction

The ECML/PKDD Discovery Challenge 2, was concerned with the identification of 7 types of common attacks in Web Traffic Data. The motivation for conducting data mining research in the specific application area, stems from the increasing importance of web-security. This is illustrated in the National Institute of Standards and Technology report (*http://www.nist.gov/*), that estimates that American companies in 2004 suffered losses of up to 59.6 million dollars as a result of IT attacks. In the context of the data mining challenge, a dataset from HTTP query logs was provided that consisted of 50000 pre-classifier instances in 7 attack-categories and a valid (normal communication) category. Challenge 2 was composed by two tasks. The first required that a predictive model was constructed with the purpose of identifying future attacks. The second task required

the isolation of the attack pattern, i.e. the identification of the exact string-area, in the HTTP protocol data, that conveys the attack. Due to time limitations we have addressed solely the first task.

The major challenge that we have faced, stemmed from the fact that the Traffic data were provided in raw HTTP protocol format. More precisely, the training data consisted of several fields of the HTTP packet, namely: *method*, *protocol*,*uri*, *query*, *headers* and *body*. Thus, a significant amount of prepossessing was required, prior to applying data mining algorithms for solving this task. In order to transform the data into feature-vector format, we have identified several useful string patterns that could signify that a communication is malicious. These features were derived using several web resources such as the Snort rules *www.snort.org*. The derived features are analytically described in section 2. It has to be noted that the related work on Data Mining for Attack Identification (such as [1–4]), was not directly relevant, as it does not focus on the construction of the feature-vectors representations.

Having transformed the data in feature-vector format, we have employed C4, a decision tree algorithm. In our experiments we have used the implementation of the C4 algorithm provided by WEKA knowledge Explorer [5]. The cross-validated estimate of accuracy was 77%, which can be deemed as satisfactory in the context of an 8-class problem. In the context of the discovery challenge workshop we expect to get the chance to interact with domain experts that will be able to evaluate the derived decision tree rules beyond their predictive ability. We have also experimented with incorporating the data into the classifier in a more preliminary format (without much pre-processing). This was attempted using a Support Vector Machine (SVM) [6] with the String Kernels proposed in [7] (again using the implementation provided by WEKA [5]). The String Kernel computed directly the similarities (kernels) between the string fields of the HTTP packet, thus allowing for their direct incorporation in an SVM classifier. Unfortunately, albeit our efforts, we were not successful in tuning the String Kernel and the String representations appropriately, and the performance of the string-based classifier was significantly lower as compared to the decision tree accuracy estimates.

The rest of the paper is organized as follows. Section 2 provides an analytic description of the Feature Extraction process. Section 3 describes the learning algorithms used and presents the experimental results. Section 4 discusses the results and contains the concluding remarks.

2 Feature Extraction

In order to provide a better illustration of the training data format, we present the instance with id=35023. This instance presents an example of an SQL Injection and Command Execution type of attack. The attack is contained in the *Accept-Language* component of the *headers* field. A closer inspection of the field reveals that the attacker attempts to make a bulk insert of file *pwdump.exe*. The training data provides us also with information as to whether the attack is in

context or not. In the specific example the attack is out of context as the Server is not running an SQL Database Server.

```
 <sample id="35023">
    <reqContext>
      <os>WINDOWS</os>
      <webserver>UNKNOWN</webserver>
      <runningLdap>TRUE</runningLdap>
      <runningSqlDb>FALSE</runningSqlDb>
      <runningXpath>TRUE</runningXpath>
    </reqContext>
    <class>
      <type>SqlInjection</type>
      <type>OsCommanding</type>
      <inContext>FALSE</inContext>
      <attackIntervall>headers:Accept-Language:1-86</attackIntervall>
    </class>
    <request>
      <method>GET</method>
      <protocol>HTTP/1.0</protocol>
      <uri><![CDATA[/cShcktp/WGRj_l3T4VCIAM/t3Ww_4V69aXiVXc6jx/
      iAdgqUz6mMT.gif]]></uri>
      <query><![CDATA[6D1c=%5BiOt%2FqSl&loh=5nkf&rooYbher9mNpne8
      =764148341&lL72ps=voteto%2FEtl9&12ztsEa2a=9015055&apscallceAbnI=
      dHo4z&8otb5Ni=aL1]]></query>
      <headers><![CDATA[Host: www.1Bcrehl.de
Connection: keep-alive
Accept: image/gif;q=0.7, video/mpeg;q=0.2, video/mpeg
Accept-Language: bulk+++insert++aearfip+++from+%27pwdump.exe%27++++with
++++%28codepage%3D%27RAW%27++%29
Client-ip: 125.169.143.8
Date: Fri, 23 Apr 10 18:12:08 UTC
If-Unmodified-Since: Tue, 14 Oct 08 19:12:53 CET
Authorization: Digest opaque="beyen"
Range: 8612-,-16,911862-7625
Referer: http://www.5A0ba.biz/ice7ne7/uo2rlh.zip
TE: trailers,deflate
Trailer: Upgrade
User-Agent: Mozilla/6.9 (Machintosh; U; PPC 6.1; or-Ob; rv:3.6.1)
Gecko/27251021
Via: FTP/9.5 165.159.141.135:514, 6.2 219.57.195.36, FTP/0.7
170.223.228.212:4305
Transfer-Encoding: gzip
]]></headers>
    </request>
  </sample>
```

For each attack type contained in the training data, we have identified certain string patterns that could signify an attack. These string patters were used in order to create a feature-vector representation of the raw HTTP packet data, where the value of each feature was set to *TRUE* if the string pattern was contained in the HTTP packet fields and *FALSE* otherwise. In the subsequent sections, we make a high level presentation (due to their large number) of the features used for each type of attack. All the details as well as the source code can be provided through personal communication.

2.1 SQL Injection

- searches in HTTP packet fields for "or" followed by "="
- searches in HTTP packet fields for "exe" or "vbs"
- searches in HTTP packet fields for "shell"
- searches in HTTP packet fields for "bulk insert"
- searches in HTTP packet fields for odd number of ""
- searches in HTTP packet fields for "'–"
- searches in HTTP packet fields for "password" or "pswd" or "account"
- searches in HTTP packet fields for "insert"
- searches in HTTP packet fields for "; junk –" or ";–"
- searches in HTTP packet fields for "update"
- searches in HTTP packet fields for "sp_"
- searches in HTTP packet fields for "xp_"
- searches in HTTP packet fields for "x=x–
- searches in HTTP packet fields for "select" followed by "from" and optionally "where""

2.2 LDAP Injection

- searches in HTTP packet fields for "user=anything) or (or | or & or %26"
- searches in HTTP packet fields for "user=*"
- searches in HTTP packet fields for "user=anything)(|(x=*)", where x: cn|uid|uidNumber|gidNumber|homeDirectory

2.3 XPATH Injection

- searches in HTTP packet fields for " ' OR y=y OR "=' ""
- searches in HTTP packet fields for "A or B or C and D"
- searches in HTTP packet fields for " " "
- searches in HTTP packet fields for " ']|/*|/foo[bar=' "

2.4 Command Execution

- searches for the ps command
- searches for existence of bin dirs
- searches for the ps command
- searches for the gcc command
- searches for user management commands
- searches for shell commands
- searches for dir commands
- searches for network commands
- searches for Windows specific commands

2.5 Path traversal, Cross-Site Scripting and SSI Attacks

- searches for PathTraversal commands and tags
- searches for SSI commands and tags
- searches for XSS commands and tags

3 Experiments

Having transformed the original training data in feature vector format we can move on and employ standard data mining algorithms to address the challenge. In our submitted model we have employed a C4 decision tree algorithm. Moreover, in order to maximize the predictive performance, we have selected a subset of the original features using the Information Gain criterion. The feature selection process was performed through greedy search, with the aim of maximizing the cross-validated predictive performance. This feature selection process resulted in the selection of 216 features from the set of original features. The expected accuracy of the submitted Decision Tree Classifier was estimated through 10-fold cross-validation to be 77%. Apart from the predictive power of our decision tree, it would be interesting to present the results to domain experts that will be able to evaluate the quality of the induced set of rules.

We have also attempted to use the training data without much preprocessing. In order to accomplish that, we have employed a Support Vector Machine classifier [6] and more precisely the Sequential Minimal Optimization SMO [8], along with the String Kernel proposed in [7]. The string kernel would handle directly the string representations of the HTTP fields without significant preprocessing. The success of this approach relied crucially on whether the String Kernel would be able to capture the "semantic distances" between the respective instances, i.e. the String Kernel should deduce that instances belonging to the same class are similar, while instances belonging to different classes are dissimilar. Unfortunately, albeit our efforts, we were not able to configure the String Kernel parameters (and preprocess the string representation appropriately) such that the String Kernel computes the "semantic distances" correctly. This resulted in very low accuracy estimates for the string-based approach, that were significantly lower as compared to the decision tree accuracy estimates.

4 Discussion - Conclusions

In conclusion, Discovery Challenge 2 presented us with the opportunity of applying data mining methodologies to address the task of identifying malicious Web Traffic. The related literature on Data Mining for Attack Identification focuses on the Data Mining component and mostly ignores the feature extraction part. It is evident that data mining researchers should focus more on the problem of feature extraction as this is essential for building successful models. The discovery challenge workshop will provide us with an excellent opportunity to interact with domain experts, assess the data mining results, and possibly open new directions for the applications of data mining in the Web Traffic analysis domain.

References

1. Lee, W., Stolfo, S.J., Mok, K.W.: Adaptive intrusion detection: A data mining approach. Artif. Intell. Rev. **14**(6) (2000) 533–567
2. Stolfo, S.J., Lee, W., Chan, P.K., Fan, W., Eskin, E.: Data mining-based intrusion detectors: an overview of the columbia ids project. SIGMOD Rec. **30**(4) (2001) 5–14
3. Barbará, D.: Special issue on data mining for intrusion detection and threat analysis. SIGMOD Rec. **30**(4) (2001) 4–4
4. Margineantu, D., Bay, S., Chan, P., Lane, T.: Data mining methods for anomaly detection kdd-2005 workshop report. SIGKDD Explor. Newsl. **7**(2) (2005) 132–136
5. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)
6. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
7. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.J.C.H.: Text classification using string kernels. Journal of Machine Learning Research (2002)
8. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Scholkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning, MIT Press (1998)

Water transport in Sumer in the kingdom of the III Dynasty of Ur

Marek Stępień, Jerzy Tyszkiewicz, Wojciech Jaworski

Institute of Computer Science, Warsaw University
Banacha 2, 02-097 Warsaw, Poland

1 Introduction

The challenge is related to a database of ca. 28'000 administrative documents from the kingdom of the III Dynasty of Ur, which existed in the 21st century b.C. in Mesopotamia, present day southern Iraq. The documents will be made available to anybody interested in the text format. An example is located at the end of this document.

The documents were originally written in Sumerian on clay tablets, using cuneiform script. The database contains transliterations of these texts, where each cuneiform sign from the tablets is transliterated into an ASCII syllable. The transliteration is more or less phonetical, but it is immaterial for the challenge. In general, each of the cuneiform signs could be read in several ways, and therefore is nowadays transliterated in several ways. Conversely, each phonetical syllable could be in principle written using more than one sign – this is why there are e.g. transliterations `lu, lu2, lu3,` etc. Conventionally, `lu` can be understood as `lu1`. This numbering has no semantical meaning, however, almost all Sumerian words have a default way of writing, and consequently, e.g., `lu` and `lu2` mean something different. Syllables written in capital letters are names of signs, used when the translator decided not to sugest any translation. Hyphens are used to compose words form syllables in transliterations, despite the fact that boundaries of words are not indicated on the original tablets. Some of the words have a post- or pre-determinative, a syllable written in transliteration in `{}` brackets. The determinatives indicate the semantical category of the word. E.g., predeterminative `dingir`, traditionally written as `{d}`, indicates a deity name, while postdeterminative `{ki}` indicates a town name and `{gesz}` is a predeterminatve for wooden objects.

On a typical tablet, the text can be written on both sides of the tablet, each of which can have several columns, and each column is divided into lines, separated by horizontal marks drawn on the tablets. Sometimes the line was too long and the last few signs were drawn below the rest of this line, but still above the separator. This is marked by / in the transliteation and has generally no semantical meaning. Empty lines are also met on tablets and often do have semantical meaning – e.g., they can separate the "grand total" from individual items in a list of transferred goods.

The principal source for Sumerian and its transliteration are The Pennsylvania Sumerian Dictionary `http://psd.museum.upenn.edu/epsd/index.html`

and the Cuneiform Digital Library Initiative `http://cdli.ucla.edu`, and references therein.

Dates, if present on the tablets, are typically located at their end and consist in the fullest form of three parts: `u4` *n*`-kam`, meaning the *n*-th day, `iti` *MN*, where *MN* is a month name (month names and their order in the year can differ depending on the province of the kingdom), and `mu` *YN*, where *YN* is a year name, referring to an important event which happend in that year (year names were uniform in the whole kingdom). The calendars, consisting of month names and year names, can be found on the Web site `http://cdli.ucla.edu/downloads.html`. Year names are not unique, i.e., in the period of III Dynasty of Ur there were several pairs of years whose names were identical.

Sometimes the tablets bear a seal impression – see example below.

The number system is rather complicated. Its description is a part of the special ATF file format documentation, see

`http://enlil.museum.upenn.edu/cdl/doc/ATF/`).

2 Difficulties

Texts on the tablets are sometimes damaged, which is indicated by '[' and ']' signs. '[' means "the following sign is destroyed on its left edge", while ']' means "the following sign is destroyed on its right edge". Consequently, e.g., `[lu2]` means a sign read as `lu2`, which is destroyed on both its edges. `[x]` means "an unreadable sign, destroyed on both edges" and `[...]` means "several destroyed signs". Many other destruction indications are met, typically written in the native tongue of the translator.

It can not be assumed that the corpus is a classical statistical sample of the corpus of all texts ever written in the kingdom of the III Dynasty of Ur. There are several reasons for it, mainly that some of the ancient towns were excavated into a great detail, yielding high numbers of tablets, while some other ones were subject to only preliminary investigation. Besides that, a large fraction of tablets comes from illegal excavations, and their origin is unknown. Furthermore, only about a half of the tablets in museum collections are translated, and the choices which ones to translate depended on many factors, including private research interests of the translators.

3 Water transport

Water transport was quite developed in the times of the III Dynasty od Ur. Barges and boats were transporting goods and people along rivers and artificially built canals. Quite large number of documents records those activities. The documents referring to water transport can be often identified by occurences of words `ma2, {gesz}ma2` meaning ship or boat, `ma2-gur8, {gesz}ma2-gur8` meaning a barge, `id2, id3, id6, id7, id5` meaning a river or a canal.

To the best of our knowledge, there were no significant attmept to describe the system of water transport in Sumer.

4 The Challenge

We expect the participants to provide answers to some of the following items.

- Identifying and dating documents related to water transport.
- Identifying the main elements of the waterway system and their interconnections.
- Estimating the quantities of cargo transported over the main routes, and their evolution over time, in the spirit of the table `http://www.allcountries.org/uscensus/1087_freight_carried_on_major_u_s.html`

The solutions will be evaluated on the basis of

- Applying well-chosen methods of data mining.
- Providing significant and statistically valid data about the water transportation system.

Authors of the best papers might be subsequently asked to co-author historical publications based on the data they have mined.

5 Example

Below we present a single text taken out of the data. The file with data is a text file composed of a sequence of documents.

```
&P100026 = AAS 038
@tablet
@obverse
1. 3(disz) gurusz u4 5(disz)-sze3
#lem: n; jurusz[male]; ud[sun]; n

2. ma2 udu niga nibru{ki}-sze3 gid2-da
#lem: ma[ship]; udu[sheep]; niga[fattened]; GN; gid[long]

3. giri3 ensi2-ka
#lem: jiri[foot]; ensik[ruler]

4. iti nesag2
#lem: itud[moon]; nesaj[offering]

@reverse
$ (seal)
@date
1. mu us2-sa ur-bi2-lum{ki} ba-hul
#lem: mu[year]; us[follow]; GN; hulu[bad]
```

```
@seal
$ (no data)
@column 1
1. {d}szul-gi
#lem: RN

2. nita kal-ga
#lem: nita[male]; kalag[strong]

3. lugal uri5{ki}-ma
#lem: lugal[king]; GN

4. lugal an ub-da limmu2-ba
#lem: lugal[king]; an[sky]; anubda[quarter]; limmu[four]

@column 2
1. ur#-[{d}li9-si4]
#lem: PN

2. ensi2#
#lem: ensik[ruler]

3. umma{ki}#
#lem: GN

4. ARAD2-zu#
#lem: PN
```

`& P100026` is the ID of the text. `AAS 038` is the identification of the first publication, irrelevant for the challenge.

One can see the mentioned text destruction signs `[` and `]`. `$seal` indicates position of a seal impression, and the text from the seal is given below the `@seal` marker.

Lines starting with `#` give the lemmatization, identifying the root of each word in the preceding line of Sumerian text, and then in the [] brackets its English translation. Shorthands' meanings are as follows: `DN` - deity name, `GN` - geographical name, `PN` - personal name, `n` - numeral. However, the lemmatization has been created automatically and contains some errors, as line 4 of the second column of seal text demonstrates: `ARAD2-zu` means "servant", but has been recognized as a personal name.

Syllables after numerals in the Sumerian text identify the sign used to write down that very numeral.

The full manual of the ATF format, used to encode the texts, is available from `http://enlil.museum.upenn.edu/cdl/doc/ATF/`

The translation is as follows:

```
1) 3 workers for 5 days
2) towing a boat with fattened sheep to Nippur
3) responsible ensi (=governor)
4) month nesag
-
(seal)
5) year after the year: Urbilum was destroyed
(=46th year of reign of Shulgi)

(i)
1) Shulgi (deified, as dingir indicates)
2) strong man
3) king of Ur
4) king of the four quarters
(ii)
1) Ur-Lisi
2) ensi (=governor)
3) (of) Umma
4) his servant
```

Given this tablet, we can argue as follows: month name `nesag2` was used only in Umma, and the text was sealed by the governor of Umma (or another official acting on his behalf and using his seal), so without any doubt the ship, referred to by `ma2`, transported fattened sheep from Umma to Nippur in the fourth month of the 46th year of Shulgi. We cannot deduce anything about the number of transported sheep, unless we find another document from that month and year recording a supply of fattened sheep from the governor of Umma for Nippur. Note however, that such a receipt would be likely written by a Nippur scribe and bear the Nippur name `szu-numun` of the fourth month. This creates a difficulty which must be overcome, because `szu-numun` is also the name of the sixth month in Umma.

The travel took 5 days (we do not know whether one-way or with return), but the former seems more likely, given the distance form Umma to Nippur and the fact the ship sailed upstream.

6 Contact with authors

The authors will maintain a public forum on the Discovery Challenge website `http://www.ecmlpkdd2007.org/challenge`, where all questions related to the challenge can be posted. We will try to answer them as soon as possible. We have chosen this *broadcast* mode of contact with the participants to ensure that everybody gets precisely the same amount of information from us. Therefore we will not answer to questions sent by e-mail.

Using Semi-supervised Learning for Mining Sumerian Administrative Documents in the Kingdom of the III Dynasty of Ur

Dimitrios Mavroeidis[1], Dimitris Diamantis[1] and Michalis Vazirgiannis[1,2]

[1] Department of Informatics, Athens University of Economics and Business, Greece
[2] GEMO Team, INRIA/FUTURS, France

Abstract. In this paper we present and discuss the approach undertaken by the DB-NET research group for addressing the ECML/PKDD Discovery Challenge 3. The challenge was concerned with the analysis of a collection of administrative documents from the kingdom of the III Dynasty of Ur, which existed in the 21st century b.C. in Mesopotamia, located in present day southern Iraq. The water transport system of the era was very developed, undertaking a substantial part in the economic growth and development of the region. The task required that data mining techniques were applied on a collection of administrative documents, with the purpose of identifying and the dating the documents related to the water transport system. In the proposed solution, we have formulated the identification problem of the water transport-related documents as a semi-supervised clustering task. Our methodological approach was motivated by the fact that we could easily identify several documents that belonged in the water transport cluster, using simple keyword matching rules. These documents were consequently used as prior knowledge in the context of a 2-way semi-supervised clustering algorithm, where one cluster was defined as containing the water transport-related documents, while the other contained the rest of the documents. Concerning the document dating process, we have observed that the exact dating (defined through kingdom eras) could be extracted from the majority of the documents in the collection. This allowed us to train a Support Vector Machine and derive a document dating model that was consequently employed for dating the rest of the documents. In order to identify the main elements of the waterway transport system we have analyzed statistically the variables of the instances belonging in the waterway cluster and its centroid. Moreover, we have used Information Gain in order to identify the variables that can can be used for separating the two clusters.

1 Introduction

The ECML/PKDD 2007 Discovery Challenge 3 was concerned with the analysis of a set of administrative documents that belonged to the kingdom of the III Dynasty of Ur. The original data were written in Sumerian, on clay tablets in cuneiform script. The data for the challenge were provided in ASCII format,

where each ASCII syllable corresponded to a cuneiform sign (through transliteration). Moreover, for several transliterations the English translation (lemmatization) was provided. The main task of the challenge was to identify the documents that were related to the water transport system. In its basic formulation the task is related to unsupervised learning, as no class labels are initially provided. However a closer inspection of the documents can reveal that using some basic keywords matching rules (i.e. documents containing the words *ship*, *boat*, *barge*, *haul* and *river*) several documents that are related to the water transport system can be identified. This class-label information allowed us to employ semi-supervised learning techniques for addressing the challenge. As opposed to the identification of the water transport documents, the document dating task could be easily formulated as a supervised learning problem, as most of the document contained exact dating information.

Semi supervised learning [1] presents a recent development in machine learning and data mining research. In semi-supervised clustering the algorithms are provided with limited information concerning the grouping of the data, usually in the form of must and cannot-link constraints. The constraints are typically incorporated into the clustering objective function, which is subsequently optimized using EM-type algorithms. Discovery Challenge 3, presents an excellent opportunity for evaluating these algorithms empirically in real world problems.

In our approach for addressing the water transport identification problem, we have employed the semi-supervised k-means algorithm proposed in [2]. The algorithm's input-constraints were defined by using several simple keyword matching rules. More precisely, we considered that the documents containing the words: *ship*, *boat*, *barge*, *haul* and *river* were related to the water transport system. Subsequently, we have derived a set of must-link constraints between all the pairs of the water transport related documents. The number of cluster was set to $k = 2$, and the resulting water transport cluster was labeled as the one containing the majority of the must-link documents. In order to identify the main elements of the waterway transport system, we have statistically analyzed the derived clusters. Moreover, using the Information Gain, we have identified the variables that can be used to separate the two clusters. Concerning the document dating task, we have employed traditional supervised learning techniques and more precisely the Support Vector Machine [3] implementation provided by WEKA [4]. The utilization of supervised learning was possible, as the dating information in the form of kingdom era, was contained in the majority of the documents in the document collection.

The rest of the paper is organized as follows. Section 2 provides a short description of the Discovery Challenge. Section 3 describes the learning algorithms used and presents the experimental results. Section 4 discusses the results and contains the concluding remarks.

2 Discovery Challenge Description

2.1 Task Description

The ECML/PKDD Discovery Challenge 3 was concerned with the analysis of a historical document collection (dated in the 21^{st} century B.C.) retrieved form the area of Messopotamia. The document collection corresponded to Sumerian cuniforms signs that were transliterated for the challenge to ASCII syllables. Moreover, in many cases English translations (lemmatizations) were provided. The discovery challenge 3 consisted of the following tasks:

1. Identification of documents that are related to the water transport system.
2. Dating of the documents that are related to the water transport system.
3. Identification of the main elements of the waterway system and their interconnections.
4. Estimation of transported cargo over the main routes and its evolution in time.

In our submitted solution we have addressed the first three tasks of the challenge. Concerning the fourth task, we report the difficulties that prevented us from addressing this task.

It has to be noted that we have addressed this challenge from the data-mining perspective, without interacting with domain experts. In the data mining process, domain experts are essential (as highlighted in the CRISP-DM process *www.crisp-dm.org*), for formulating the appropriate data mining problems (business understanding) and for assessing the quality and usefulness of the derived data mining models. A domain expert could enhance and improve our approach in the following manners:

1. Build better representations for the documents.
2. Define distance measures that capture the semantic relatedness between the documents.
3. Evaluate the appropriateness of the semi-supervised clustering results.
4. Provide the necessary feedback for tuning the algorithms' parameters (i.e. the number of clusters).

3 Experiments

3.1 Document Representation

The data representation scheme was based on the Vector Space Model (VSM) [5]. The VSM is a standard approach used for representing text, which allows for the direct employment of standard supervised and semi-supervised learning approaches. In our vector representations, we have considered the index of all the transliterations, and represented the documents as a boolean vector based on the words (transliterations) they contained. The resulting index contained totally ca. 51000 features.

3.2 Water Transport

In its basic formulation the identification of the water transport documents can be addressed by unsupervised learning algorithms. Data Mining experts along with domain experts can employ clustering algorithms, and based on the characteristics of the resulting clusters, identify the cluster that is related to the water transport system. However, we have observed that there exists a number of documents that can be easily identified as related to the water transport system. These documents contained the words: *ship*, *boat*, *barge*, *haul*, *river*. This observation allowed us to guide the clustering process, through utilizing semi-supervised approaches.

In the submitted solution, we have employed a recent development in semi-supervised k-means clustering [2], and its implementation provided in (*http://www.cs.utexas.edu/users/ml/risc/code/*). In the experiments we have set $k = 2$, where one cluster was considered as related to the water transport system, while the other contained the rest of the documents. In the submitted results, the cluster that contained the majority of the initially identified water-transport documents was labeled as the water transport cluster. It should be noted that the configuration of the k parameter could be suboptimal, however due to the absence of domain experts that could evaluate the characteristics and the meaningfulness of the results, we report the cluster analysis (section 3.4) based solely on $k = 2$.

3.3 Dating of Documents

In order to address the document dating problem, we have observed that we could derive the dating of the majority of the documents, in the form of kingdom era. This allowed us to use standard supervised learning approaching for constructing dating models that were consequently be used for dating the rest of the documents. In the results we have submitted for the challenge, we have used a Support Vector Machine (SVM) classifier and more precisely the SMO [6] implementation of WEKA [4]. with a Linear Kernel and complexity parameter $C = 1$.

3.4 Analysis of Water Transport Cluster

In order to gain a better understanding of the characteristics of the water transport related documents, as identified by the clustering process, we have analyzed them statistically. Initially, we have used Information Gain, in order to identify the top-10 variables that separate the two clusters. These variables are reported in Table 1.

Ranking	Variables
1.	EraOfText
2.	mu
3.	en
4.	{d}nanna
5.	unu6
6.	{d}inanna
7.	ba-hun
8.	us2-sa
9.	kar-zi-da
10.	ba-hul

Table 1. Top-10 discriminative variables as derived by Information Gain

In table 1, it can be observed that the top discriminative variable is *EraOfText*, that contains the kingdom era of each document. This observation allows us to conclude that the majority of the water-transport related documents, as derived by the clustering procedure, belong to certain kingdom eras. This information can prove to be valuable to archeologists, as it would allow them to concentrate on a subset of the initial document collection, when trying to analyze the water transport system. The rest of the variables, reported in Table 1 are transliterations.

Moreover, we have analyzed the variable values of the instances belonging to the water transport cluster. Our aim was to detect the variables whose mean values are statistically significantly different in the water transport cluster as compared to the whole collection of instances. In Table 2, we report the variables, whose mean values are maximally different in this respect. It has to be noted that the differences are not statistically significant at the 95% confidence level.

In Table 2, we report the lemmatization of the transliterations as well (enclosed in brackets). Moreover the symbol *PN* stands for Personal Name, symbol *GN* stands for Geographical Name, *WN* stands for Waterway Name, n represents a number and u means unlemmatizable.

Furthermore, we have analyzed the values of the centroid of the waterway cluster, in order to detect the characteristics of this cluster. In Table 3, we report the Top-10 Variables (with the highest values) of the centroid of the waterway cluster.

Ranking	Variables
1.	ensi2-ka ensik[ruler]
2.	nita nita[male]
3.	sirara6-he2-gal2 PN
4.	e2-kikken-ta ekinkin[mill]
5.	hul-a hulu[narrow]‖*hulu*[*ruination*]
6.	ak ak[do]
7.	si-i3-tum situm[balance]
8.	uri5ki-ma GN
9.	kal-ga kalag[strong]
10.	sag10 sag[rare]
11.	limmu2-ba limmu[four]
12.	zu-zu-ma-ISZ PN
13.	pa4 pap[relation]
14.	6(gesz2) n
15.	geme2-ri-im-i3-li2 PN
16.	lu2-dx PN
17.	sze n
18.	1(barig) n
19.	ur-diszkur PN
20.	KU u

Table 2. Top-20 variables with maximal mean differences

Ranking	Variables	Values
1.	si-i3-tum situm[balance]	0.79
2.	sze n	0.64
3.	la2-ia3 la'u[arrears]	0.52
4.	IL2 u	0.42
5.	a2-bi u	0.36
6.	ki u	0.36
7.	2(gesz2) n	0.36
8.	a-ab-ba PN	0.32
9.	szesz-kal-la u	0.32
10.	8(gesz2) u	0.26

Table 3. Top-10 variables values for water transport cluster centroid

3.5 Estimation of transported cargo over the main routes

In addressing the task of estimating the transported cargo over the main routes, we have encountered the following problems that prevented us from submitting a solution for this task:

1. To water transport related documents that contained river names (i.e. the symbol *WN* that stands for Waterway Name) where very few as compared to the total number of cluster elements.

2. Concerning the documents that contained the word "total" (that implies that a certain cargo was transported), we could not identify the route of the transportation as these documents referred in many cases either to only one city, or in more than two cities.

4 Discussion - Conclusions

In conclusion, Discovery Challenge 3 presented us with the opportunity of applying data mining methodologies and approaches to address the original task of analyzing historical documents, dated in the 21^{st} century B.C. The discovery challenge workshop, provides an excellent opportunity to interact with domain experts, assess the data mining results, and possibly open new directions for the applications of data mining in the historical document analysis domain.

References

1. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-Supervised Learning. MIT Press, Cambridge (2006)
2. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML. (2004)
3. Vapnik, V.: Statistical Learning Theory. Wiley (1998)
4. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)
5. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11) (1975) 613–620
6. Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Scholkopf, B., Burges, C., Smola, A., eds.: Advances in Kernel Methods - Support Vector Learning, MIT Press (1998)

Author Index