Online Change Detection in Individual Web User Behaviour

Peter I. Hofgesang
hpi@few.vu.nl

Jan Peter Patist
jpp@few.vu.nl

VU University Amsterdam, Department of Computer Science
De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands

ABSTRACT

Based on our field studies and consultations with field experts, we identified three main problems that are of key importance to online web personalization and customer relationship management: 1) detecting changes in individual behaviour, 2) reporting on user actions that may need special care, and 3) detecting changes in visitation frequency.

We propose solutions to these problems and experiment on real-world data from an investment bank collected over 1.5 years of web traffic. These solutions can be applied on any domains where individuals tend to revisit the website and can be identified accurately.

Categories and Subject Descriptors

E.1 [**Data**]: Data Structures - Trees; H.2.8 [**Database Management**]: Data Mining

General Terms

Algorithms, Experimentation

Keywords

Incremental Web Mining, User Profiling, Change Detection

1. INTRODUCTION

Web personalization techniques allow companies to customize their web services according to their customer's needs. However, changing customer behaviour and frequent structural changes and content updates of websites are likely to impair the underlying models. Current personalization techniques require periodic updates and human interaction to cope with this problem and thus maintain up-to-date personalization. To automate this process, we need to identify points of change (see e.g. [1]) in user behaviour automatically as the data flows in. Identifying such changes - whether they are seasonal or temporal, or a long-term behavioural trend shift - and promptly taking appropriate action in response leverages business advantage.

During our analysis of numerous web usage datasets, including data from online retail shops and an online bank, we identified three main problems that we summarize in the following three goals:

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

Goal 1: Detect changes in individual behaviour A solution to this problem should include incremental maintenance of a compact user profile for each individual and the output of changes. Change signals can be used to update personalized content or for marketing actions while the base profiles can help selecting the proper personalized content.

Goal 2: Report "interesting" sessions From time to time online users may not be able to find some content or they may have technical difficulties. The idea behind the second problem is to identify such points, "interesting" sessions, in a stream of user actions. Based on such alerts, an assistant may offer instant online help (e.g. chat or voice call) or the system may take prompt automated action.

Goal 3: Detect changes in user activity Detecting changes in an individual's visitation frequency seems to relate to our first problem; however, it requires different treatment both technically and in the types of output alert. For example, early identification of a decrease in an individual's visitation frequency may support actions for customer retention. Based on the different types of change in visitation frequency, we may label changes as "runner up" or "slower down" based on the increase or decrease in frequency.

The contribution our work makes is the identification of three relevant problems in individual, on-the-fly web personalization and their proposed space and computationally efficient solutions.

2. INDIVIDUAL USER PROFILES

We choose to use a tree-like structure to represent user profiles. Each node may contain more than one page id, a frequency counter, a reference to its parent and a vector of references to its children.

Adding a session (we keep visited pages as sets, discarding order and cardinality information) to a profile tree can be broken down as a simpler recursive procedure. The procedure is called, in each step, by a given tree node n and the remaining pages to be inserted, P. As a first step, we select the children of n that have the highest overlap with P. If there are more nodes with equal, non-zero overlap we select the first one. If P has no overlap with any of the children nodes we insert P as a new node under n. Otherwise, we identify the type of overlap. In case the children node is equivalent to P or is fully part of P, we increase the counter of the children and invoke another recursive step on the children node and the non-overlapping remainder of P. In the other two cases, we need to split the children node by subtracting and raising the overlap directly above it as a new parent node, update the frequency counters and insert

the non-overlapping part, if there is any, as a new children node under the newly created node of common pages.

In our methods, we **maintain the profiles** over a sliding window. This can be easily done by maintaining references to leaf nodes in the tree in a "round" vector.

Here we define a **distance metric** over a profile tree and an incoming session (S). In fact, we define a similarity metric and we transform a distance by $dist = 1 - sim$. The similarity measure basically selects a branch with the highest overlap to the input session and calculates a score on it. Once again, we can define this metric recursively as each node returns the number of its overlapping pages with the subset of pages it received plus the total overlap on the remainder of pages returned from its children nodes. Each recursive step also returns a subscore and, among identical overlaps, we return the highest score as similarity. We calculate the scores at each node by taking the minimum of the normalised node frequency ($\frac{n.frequency}{norm_T}$) and the page probability ($norm_P$). Both norms are known upfront to the algorithm. $norm_T$ is simply the number of pages added to the tree and $norm_P = 1/|S|$.

3. CHANGE DETECTION FRAMEWORK

Goal 1: Detect changes in individual behaviour A user profile is maintained for each individual and compared, with a window of most recent sessions. First, the tree is initialized with n sessions. The score of an incoming session is calculated using the distance function defined in Section 2. If the session is not an outlier ($score < t_{outlier}$), it is added to a buffer (ref), excluding the last m scores, which are kept in a buffer W. A change is detected when the distance (d) between the most recent scores (W) and ref is larger than a threshold, T_{change}. The distance is defined as $d(ref, W) = \prod_{w \epsilon W} ecdf_{ref}(w)$, where $ecdf(x)$ is the empirical cumulative distribution function of ref. If we maintain the $ecdf_{ref}$ using a balanced tree algorithm, the complexity of the algorithm is $O(log(|ref|))$ and the distance function, $dist$, can be calculated efficiently as well.

Goal 2: Report "interesting" sessions The problem of finding "interesting" sessions is rather subjective and cannot be clearly formulated. We choose to rate pages as "highly interesting" when they are popular for a specific individual but rarely visited by others. Interesting sessions are those containing interesting pages. We consider only outlier sessions (calculated in Goal 1) as potential interesting sessions. We calculate a new score over the outlier session based on an automatically maintained popularity matrix. The components of this matrix are weights calculated by the $TF - IDF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$ weighting applied to our data, where i refers to a term (web pages) and j to documents (individual sessions) and n_{ij} is the number of occurrences of term i in document j. We label a session as interesting if its overall score is higher than a user-defined threshold.

Goal 3: Detect changes in user activity While the first two problems, Goals 1 and 2, can be included in a common framework, we need another layer of data abstraction and a different system for the third problem. The problem at hand is detecting change in a stream of binary data representing whether the user was active on a given day. Our algorithm is based on the CUSUM (cumulative sum) algorithm [3] and we model the probability density function (pdf) of the data by the poisson distribution. For a given user we incrementally estimate the probability $\hat{p}$ of visitation on a given day. The estimate $\hat{p}$ is compared to h alternatives p_{alt} via the $S_c = \sum_t^c log(\frac{p_{alt}}{pdf_{\hat{p}}})$ sums, where t is the last change point and c is the current time. Each S_c, for every h, is maintained incrementally. The h alternatives are fixed upfront. Given that $\hat{p}$ is more likely than the alternatives, S_c has a downward trend. An increasing S_c indicates a change in the trend and a change is detected if $S_c - \min_{t...c} S_c > T_{alarm}$. The complexity is linear in the number of alternatives, which is $O(|H_1|)$ per session.

4. EXPERIMENTAL RESULTS

Our experiment included results on server-side web access log data of an investment bank collected over a 1.5-year period. To evaluate our change detection methods, we randomly selected two sets each of 200 individuals with their sessions, and labelled them together with field experts. We looked at the individual sessions evolving over time and marked the points of change.

Goal 1: evaluation The accuracy of our method is 69.57% with 67 false alarms and the mean detection latency is 5.12 sessions.

Goal 2: evaluation While experimenting with the TF-IDF scoring, we found that the high popularity of a relatively few pages among all individuals resulted in extreme high term frequency ($TF_{i,j}$) components that overweighed the $IDF_{i,j}$ scores and, even though popular pages were present in most profiles, their final $TF - IDF_{i,j}$ scores were much higher for common pages than the much more interesting rare pages. To avoid the effect of the power law distribution of web pages, we ignored the $TF_{i,j}$ weight and used only the $IDF_{i,j}$ component in final scoring.

Goal 3: evaluation The accuracy of our method is 77.36% with 117 false alarms, and the mean detection latency is 6.37 days.

Figure 1 shows an example of score evolution and change detection both for Goal 1 and Goal 3. For more experimental results we refer the reader to [2].

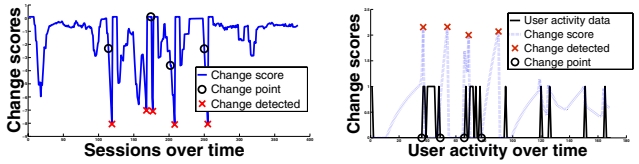

Figure 1: Example Goal 1 (left) and Goal2 (right): Score evolution and change detection for individuals

5. REFERENCES

[1] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall, Inc., 1993.

[2] P. I. Hofgesang and J. P. Patist. Online change detection in individual web user behaviour. In *http://www.few.vu.nl/~hpi/papers/IndivChangeDet.pdf*. Technical Report, Vrije Universiteit Amsterdam, 2007.

[3] E. Page. On problems in which a change in a parameter occurs at an unknown point. *Biometrika*, 44:248–252, 1957.