ABS: the Anti Bouncing Model for Usage Data Streams

Chongsheng Zhang*, Florent Masseglia
INRIA Sophia Antipolis-Méditerranée
AxIS Project-Team, 2004, route des Lucioles - BP 93
06902 Sophia Antipolis, France
{*chongsheng.zhang,florent.masseglia*}*@inria.fr*

Yves Lechevallier
INRIA Paris-Rocquencourt
AxIS Project-Team, Domaine de Voluceau - BP 105
78153 Le Chesnay, France
yves.lechevallier@inria.fr

***Abstract*—Usage data mining is an important research area with applications in various fields. However, usage data is usually considered streaming, due to its high volumes and rates. Because of these characteristics, we only have access, at any point in time, to a small fraction of the stream. When the data is observed through such a limited window, it is challenging to give a reliable description of the recent usage data. We study the important consequences of these constraints, through the "bounce rate" problem and the clustering of usage data streams. Then, we propose the ABS (Anti-Bouncing Stream) model which combines the advantages of previous models but discards their drawbacks. First, under the same resource constraints as existing models in the literature, ABS can better model the recent data. Second, owing to its simple but effective management approach, the data in ABS is available at any time for analysis. We demonstrate its superiority through a theoretical study and experiments on two real-world data sets.**

***Keywords*-data streams; usage; clustering; bounce rate;**

I. INTRODUCTION

The bounce rate (BR) of a website is the percentage of visitors (or users) who hit a given page and do not visit any other page on that website. It is defined as $BR = (\frac{T_o}{T_v})$ with T_o the total number of visits viewing only one page and T_v the total number of visits. According to Wikipedia *it essentially represents the percentage of initial visitors to a site who "bounce" away to a different site, rather than continue on to other pages within the same site* [12]. Bounce rate is very important for usage analysis and most commercial websites would like to lower it[1]. Actually, let us consider two websites A and C. When a user clicks through a paid advertising on website A and arrives on a landing page on C, she is expected to navigate through several pages on C. If this is not the real case, then the advertisements may be not well targeted. Usually, the aim of commercial websites is to drive their users through multiple pages since they are expected to click on paid advertisings. For such websites, bounce rate indicates how the pages succeed in encouraging the users to browse different pages. There are many reasons for a high bounce rate. We separate these reasons in two categories.

The first category is related to the content of the page, say, its relevance with regards to the users interests, the links to other pages, the bad ergonomics or the keywords which do not reflect its content.

The second category is related to the data model used for the usage analysis. This is particularly true for data streams. We claim that, in some cases, the observed bounce rate is higher than the real one, because of the data stream model. *To the best of our knowledge, this is the first paper providing a study for lowering the observed bounce rate in data streams.*

Let us introduce some definitions related to usage data streams. Because of their high volumes and rates, it is usually impossible to analyze such streams in real time. Sometimes, it is even impossible to solely store their whole content.

Definition 1: An **event** $e_i =< uid, time, page >$ is a tuple where i is the event identifier, *uid* is the user identifier, *time* is a timestamp and *page* is the page requested by user uid at that timestamp. An **event data stream** is a stream of events.

Definition 2: An **observation window** of size n is a set of n events from the stream.

Definition 3: The **navigation** of a user u_i at time t is the series of events $e_x =< u_j, t_k, p_l >, k \in [0, t]$ where $u_j = u_i$.

According to definition 1, an event data stream contains the users requests. Since the whole set of events from a data stream is too large to fit in main memory, the stream is usually processed through an observation window containing a subset of n events (C.f. definition 2). The navigation of a user, as given by definition 3, contains the set of pages that have been requested by that user up to the current time. A **model** represents the information and description of the stream. In our case, a model is a set of navigations, built on the events that have been selected from the stream. Obviously, the content of a model depends on the event selection strategy. Let k, be the maximum number of events that can be kept in main memory. A popular data stream

*This work was partially funded by ANR, grant number ANR-07-MDCO-008-01/MIDAS

[1]Meanwhile, some websites won't try to lower their bounce rate. A website might, for instance, want to provide its users with fast and accurate information (in that case, it does not want to keep the users as much as possible on its pages).

model is based on batches of events [5], [6] where the observation window is the chunk containing the last k events. For each batch, the events are processed while the next batch is being filled with the new events. Each batch is discarded when the next one is ready for processing. Example 1 illustrates this model and its principle on a toy dataset.

(1)

Event	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
User	u_1	u_2	u_3	u_4	u_1	u_5	u_3	u_6	u_5	u_7
Page	a	b	d	b	c	a	e	b	c	b

(2)

u_1	u_2	u_3	u_4	u_5	u_6	u_7
a	b	d	b	a	b	b
c		e		c		

(3)

u_1	u_3	u_5	u_6	u_7
c	e	a	b	b
		c		

Table I
STREAMING EVENTS (1), THEIR CORRESPONDING NAVIGATIONS (2) AND THE NAVIGATIONS IN B_n, THE BATCH THAT CONTAINS THE LAST SIX EVENTS (3).

Example 1: Let us consider the events given in Table I (1). Each event e_i associates an event Id i, a user (or visitor) and a page. The navigations on the whole dataset in this example are given in Table I (2). For instance, the navigation of u_1 contains two pages (a and c). For simplicity, we only keep the pages in the navigations (we don't show the timestamps). Let us consider B_n, which is the batch containing the last six events in this stream (*i.e.* $[T_5..T_{10}]$). The navigations of B_n are given in Table I (3). The main observation is that the navigations of users u_1 and u_3 are truncated. This has important consequences on the data analysiss since i) the observed bounce rate in B_n is much more important than the real bounce rate ($\frac{4}{5}$ in B_n vs. $\frac{4}{7}$ in the whole data) and ii) the clusters that would be obtained in B_n and the ones obtained in the total data are very different. Actually, a reasonable clustering obtained on the whole dataset would be $\{Clust_1 = (u_1, u_5); Clust_2 = (u_2, u_4, u_6, u_7); Clust_3 = (u_3)\}$ which would be described by the centers $\{a, c\}$, $\{b\}$ and $\{d, e\}$. Meanwhile, the clusters obtained for B_n would be $\{Clust_1 = (u_1); Clust_2 = (u_3); Clust_3 = (u_5); Clust_4 = (u_6, u_7)\}$ having centers $\{a, c\}$, $\{b\}$, $\{e\}$ and $\{c\}$.

Existing data stream models are based on removing obsolete events. Actually *they sort the events by timestamp and they maintain a maximum number of events in the model*. Therefore, the navigations of most users cannot be reliably retrieved. In Example 1 the navigations of users u_1 and u_3 are truncated.

Hence, the need for a better model that would not throw away such a precious information. We introduce the ABS (Anti-Bouncing Stream) model, a new model relying on a novel point of view.

II. RELATED WORKS

Bounce rate is a recent, though important, measure that is relatively unstudied in the literature. In [8], the authors proposed interesting techniques towards prediction of an advertisement bounce rate by analyzing its features. Though not related with data streams and observation window issues, this recent paper is one of the first studies on this subject.

Regarding **Data Streams Models**, [6] gives an interesting comparative study of batch and sliding windows (SW in the remaining of this paper and Fifo in [6]). The authors propose two approaches designed towards anytime algorithms and their exploitation in data streams. With SW, we need to maintain a list of current events in the model [9]. The main difference with batch lies in i) the frequent updates ii) the availability of the model at anytime. However, when a batch is complete and ready for analysis at time t, the SW model at time t contains the exact same events (and navigations). Let us also mention some other models such as the landmark windows [7], where the analysis is maintained for a window ranging from one fixed point in the past to the current time, and the decaying factor [3], [4] which aims to give higher weight to recent events in the analysis. Most papers on data stream mining have considered streams of feature vectors [1], [10], [4], [5], [11], [13] where there is no link between the records (*i.e.* each new record in the stream is the whole set of a user's requests, whereas in our context, the users' requests arrive in parallel and event after event). Essentially, papers in this context are dedicated either to clustering or to frequent itemset mining. For clustering, let us cite [1] where the authors introduced the concept of micro-clusters and the CluStream algorithm. For itemset mining, [5] exploits the FPGrowth algorithm in a batch environment and uses a decaying factor to manage the history of extracted patterns. In this paper, we consider the case of data streams where the events belong to global objects. In our case, these events are pages requested by users. Therefore, the data is streaming on two dimensions: the pages and the users. Despite the possible applications associated with this kind of data streams, they received little attention in the literature. The authors of [5] also consider a data stream of events, where the users sequences are built for each batch.

III. ABS: THE ANTI-BOUNCING STREAM MODEL

In this section, we propose a new model that allows for a seamless representation of the recent events, makes the data available for analysis at any time and has a fast and straightforward update principle.

The key idea of our model is to eliminate the idea of observation window on the event data stream. While the models based on batches, sliding windows or decaying factors maintain a list of events, our model only considers the new incoming events one by one. However, we cannot always add new events without regularly removing some data since we cannot afford the memory overhead. Essentially,

ABS provides a new management and pruning principle as follows:

1) Acquire new events and update the model on the fly.
2) Maintain a relation of order between the users.
3) Monitor the current number of events in the model and remove the last user from the model when the maximum available memory is reached.

A. Update principle

The main difference between ABS and the existing models in the literature (besides the absence of an observation window on the data stream) is that ABS doesn't sort the events. *Instead of a relation of order between the events, ABS proposes and exploits a relation of order between the users*. The most up to date user is at the head of the structure while the user at the other end of the structure is the one with the oldest update. Thus, ABS arranges the users in a sorted list that is updated after each event is read from the stream. When a new event occurs, the corresponding user is updated or created, and moved to the head of the list. This operation is not costly. Retrieving a user in order to add the new event to its navigation can be done in $O(1)$ time, thanks to a map (as it is the case in Batch and SW). Afterwards, moving the user to the head of the list is straight-forward and done in $O(1)$ time. *Thanks to this update principle, the users are always sorted from the less up to date to the most up to date.*

B. Pruning principle

The pruning principle of ABS allows for fast and relevant removal of users. We consider a maximum number of events allowed in the model. This maximum number can be, for instance, equal to the size (number of events) of a batch. When the current number of events is larger than the maximum, ABS removes the last user from the model and the number of events in the model is decreased accordingly (*i.e.* decreased by the number of events of the removed user). Consequently, the number of events in ABS and Batch or SW is approximately the same at any time (the number of pages contained in the user removed from ABS is negligible with regards to the number of events maintained in the model).Furthermore, the pruning step of ABS is faster than SW, since we don't need to retrieve any user for a removed event. We just remove the user at the end of the structure.

C. Algorithm

As can be seen in our pseudo-code of ABS (Figure 1), the data in our model is available for analysis at any time (like SW) while proposing a management that is as straightforward as Batch. Furthermore, our model allows a seamless representation of the recent data in the stream. Actually, the navigations in ABS are usually longer, compared to a Batch or a Sliding Window with the same memory size. We propose an analysis of these properties in section III-D.

Algorithm: ABS
Input: DS, an event data stream and M, the memory size (number of events).
Output: $Navigations$, the navigations contained in the model.

$nbEvents \leftarrow 0$
While not end of stream **Do**
1) nbEvents++
2) $e(u,t,p) \leftarrow$ ReadEvent(DS)
3) **If** $u \in Navigations$ **Then** AddEvent($Navigations[u]$,e)
4) **Else** CreateNavigation($Navigations$, e)
5) MoveToFirstPosition($Navigations[u]$)
6) **If** $nbEvents > M$ **Then**
   a) $nbEvents \leftarrow (nbEvents-$ number of events in the oldest/last navigation)
   b) RemoveLastNavigation($Navigations$)
7) **End If**
8) **If** Analysis requested **Then** Analysis($Navigations$)

Done
End ABS

Figure 1. Algorithm for the ABS model.

D. Does ABS Avoid Splitting Down the Navigations?

We consider that i) the occurrence of event $e_i = (u,t,p)$ is independent of $t-1$ (the time of the previous event) and ii) the probability distribution of a number of events occurring in a period $[0,T]$ is the Poisson distribution. If the expected number of occurrences in this interval is η then the probability that there are exactly n events is $f(n,\eta) = \frac{\eta^n.e^{-\eta}}{n!}$.

Let λ_i be the average rate of events for user u_i and N_i be the number of events associated with u_i during the period $[0,T]$. The distribution of N_i is a Poisson distribution given by:

$$P[N_i = k] = \frac{(\lambda_i T)^k.e^{-\lambda_i T}}{k!}$$

Let τ_i be the time interval between two events of u_i. The probability that the time between two events of u_i oversteps a value t is given by: $P[\tau_i > t] = e^{-\lambda_i t}$. Let us now consider two users a and b. At time $t=0$, the event $(a,0,p_0)$ occurs, associated with a, and at time $t_b \in [0,T]$, the event (b,t_b,p_{t_b}) occurs, associated with b. Let us analyze the probability that a remains in ABS and accumulates pages in his navigation. The probability P_a that an event of a occurs before T is the probability that the second event of a occurs before T, knowing that this time is larger than t_b. $P[\tau_a \leq T/\tau_a > t_b]$, this probability, is given by $1 - e^{-\lambda_a(T-t_b)}$ (according to the Bayes theorem). Let us note $P_{\neg b}$ the probability that no event associated with b occurs before T. It is given by the probability that the time

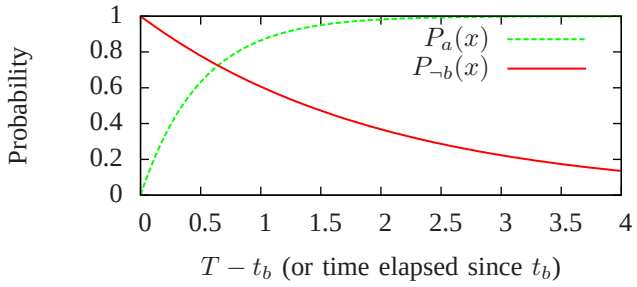

Figure 2. Probabilities that an event occurs for user a and does not for user b.

between two events associated with b is larger than $T - t_b$, *i.e.* $P[\tau_b > T - t_b] = e^{-\lambda_b(T-t_b)}$.

If P_a is high (close to 1) and $P_{\neg b}$ is low (close to 0), a will remain in ABS, along with the event $(a, 0, p_0)$ but b, along with the event (b, t_b, p_{t_b}) will be removed. Actually, P_a depends on the values τ_a and $T - t_b$. If the difference between λ_a and λ_b is significant, then the event $(a, 0, p_0)$ is highly likely to stay in the model. Let us consider, for instance, $\lambda_a = 2$ (the average rate of events associated with a is 2 per second) and $\lambda_b = 0.5$. Figure 2 gives the values of P_a and $P_{\neg b}$ for $T - t_b \in [0..4]$ in this case. At time $T - t_b = 2$ in Figure 2, we have $P_a \simeq 1$ and $P_{\neg b} \simeq 0.35$. Therefore, user a has a high probability to stay in the model since $P_{\neg b}$ is not small.

However, as can be seen in Figure 2, at time $T - t_b = 4$, $P_{\neg b} \simeq 0.15$ and the chance that new events associated to a and b occur are high. Actually, the analysis above is more complicated when the period $[t_b, T]$ is large. In this case, the probability that an event occurs becomes high for any user and it is not easy to evaluate the chance of $(a, 0, p_0)$ (the first event of a) to stay in the model. We need to consider that a new event $(a, t', p_{t'})$ occurs to evaluate the probability that $(a, 0, p_0)$ stays in the model.

Let $U(t') = \{u \in U / \exists (u, t, p_t)$ where $t \in [0, t']\}$ be a set of users where the last event (u, t, p_t) occurs in the period $[0, t']$. Therefore, for each new event $(a, t', p_{t'})$ of a, the probability that $u \in U(t')$ is not associated with any event during the period $[t', T]$ is given by $e^{-\lambda_u(T-t')}$. If this probability is small, then the event $(a, 0, p_0)$ has high probability to stay in the model. Actually, in this case the probability $P[\tau_a \leq T - t'] = 1 - e^{-\lambda_a(T-t')}$ must be compared to

$$min\{P[\tau_u \leq T - t_u / \tau_u > t' - t_u] = e^{-\lambda_u(T-t')} / u \in U(t')\}$$

where t_u is the time of the last event of u in the period $[0, t']$. Therefore, the probability that a stays in ABS depends on the minimum of values $\lambda_u, \forall u \in U(t')$. In other words, there are two important influence factors on the chances of a to remain in ABS. First, as the value λ_a represents the frequency of events associated with a during this period, a long navigation (where the number of pages is important) has a significant probability to be updated in ABS. Second, the probability that a stays in the model is larger when there exists one user u_{low} with a low frequency of events such that one event of u_{low} occurs in the period $[0, t']$.

The above reasoning is based on time intervals. However, our model is based on a given number of events. Meanwhile, it is possible to build a relationship between the time period $[0, T]$ and the number n of events in the model. It is given by the sum of Poisson distributions. Since the number of events of a in the period $[0, T]$ follows a Poisson distribution, the mathematical expectation or mean $E[N_a]$ is equal to $\lambda_a T$. Furthermore, since the random variables $(N_u / u \in U)$ are independent, we have $E[\sum_{u \in U} N_u] = \sum_{u \in U} \lambda_u T$. Consequently, the value T of the period $[0, T]$ can be estimated by:

$$T = \frac{\sum_{u \in U} \lambda_u}{n}$$

IV. Experiments

We evaluate our algorithms from three points of view: bounce rate, clustering results and time response. We have implemented and tested three models (Batch, SW and ABS) on two datasets. The first dataset comes from Orange Labs (a major mobile phone company) in the context of the ANR MIDAS project. It will be denoted as "Mobile usage" data in the rest of this section. It contains 3 months of requests from the suscribers to their mobile portal. For each month, the log file is approximately 7GB and contains 19 millions requests. The second dataset comes from the WWW access log file of Inria from February 2006 to May 2007. It is 14 GB and contains 20 millions requests (both log file formats are different, hence the different file sizes). For each dataset we want to know if our results are close to the ideal case where we could afford to analyze the stream with a much larger window. Therefore, we consider a "reference" model which is a batch of high capacity and the results on evaluated models will be compared to the results on the reference. Our reference is based on the principle of a batch that is 3 times larger than the evaluated model. During the stream processing, we randomly chose 200 random points for our measures. The points are selected prior to the experiment and are the same for each model. In the following experiments, "Window Size" is the size of the observation window (number of events allowed in memory). When a batch is ready for analysis, the data of SW and Batch at that point in the stream is the same. Therefore, our measures often compare ABS to one model called Batch/SW when it is clear from the context.

A. Bounce Rate and Average Length of Navigations

We first measured the bounce rate of each model (ABS and Batch/SW) under the same constraint of memory size. Our measures, given by figure 3 clearly show a lower bounce rate for ABS on the mobile data. For instance, with a

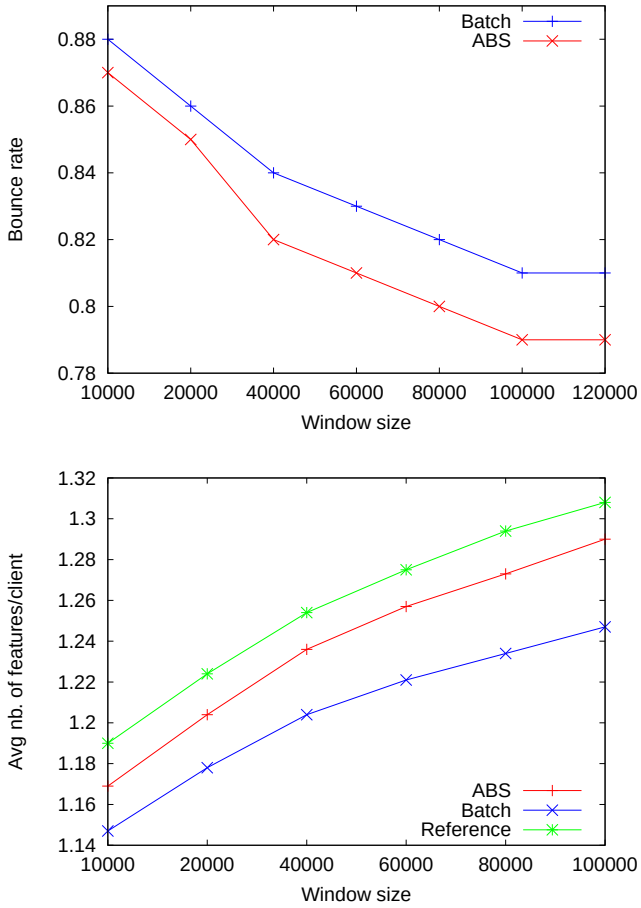

Figure 3. Average bounce rate (top) and average length of navigation per user (bottom) in ABS and Batch/SW on the Mobile usage data.

memory size of 120,000 events, the bounce rate on the Mobile usage data is 0.81 with Batch/SW and 0.79 with ABS. There are approximately 5 millions users in this file. The difference represents 100,000 users that were wrongly considered in the bouncing category. The average navigation length is represented by 3. We can observe that, unlike Batch/SW, ABS is very close to the reference in the Mobile usage data, whatever the memory size. Due to lack of space we don't show the comparison on the WWW usage data, but they are very similar. The reader should keep in mind that we don't give here the real value of bounce rate and average length of navigations in the Mobile usage data (these statistics are not publicly available). These numbers are obtained from observation windows on a biased sample.

B. Cluster Validation

We compared the clustering results on the data of ABS and Batch/SW with the reference. Our clusters come from an implementation of AP [14] applied at each random step. For a comparison, we only keep the users at the intersection between the evaluated model and the reference. We measure the purity and entropy values as described in [2].

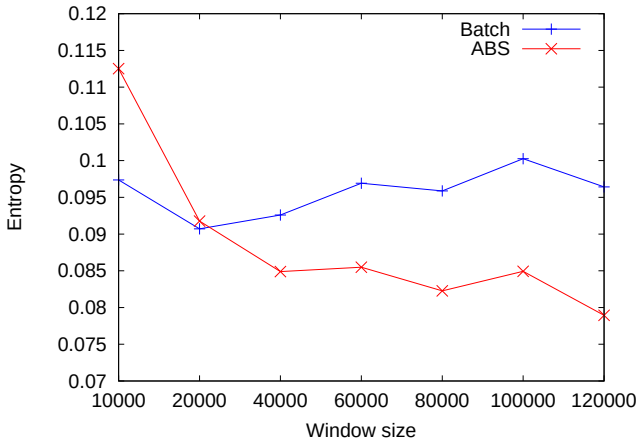

Figure 4. Average entropy on the Mobile usage data.

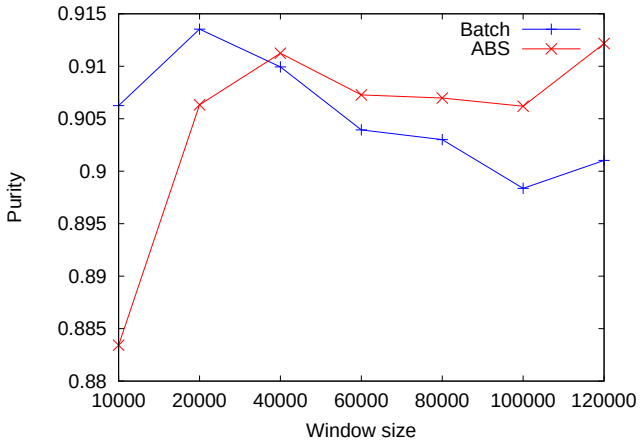

Figure 5. Average purity on the Mobile usage data.

Figure 4 gives the average purity of the clusters on ABS and Batch/SW compared to the reference with the Mobile usage data. We can observe that ABS has a better entropy than Batch/SW from 20,000 to 120,000 events in memory. The global trend of ABS' entropy is to decrease with the memory size, when Batch's entropy grows. Figure 5 gives the average purity of the clustering of ABS and Batch/SW with the Mobile usage data. Once again, the higher the memory size, the better the clustering of ABS. Due to lack of space we don't show the comparison on the WWW usage data, but they are very similar.

C. Time Response

The most time consuming part of our algorithms is the analysis (*i.e.*, in our case, the clustering). The complexity

WWW usage data

M	ABS	Batch	SW	Ref	ABS Vs. Best
10000	423	902	939	4439	46.9 %
20000	2591	3983	4200	17782	65 %
40000	4163	5645	5686	24472	73.4 %
60000	11241	15101	15168	73535	74.4 %
80000	9173	10587	10665	63163	86.6 %
100000	14991	23848	23751	days	62.9 %
120000	14535	16806	16835	days	86.5 %

Mobile usage data

M	ABS	Batch	SW	Ref	ABS Vs. Best
10000	164	159	294	505	103.1 %
20000	327	304	438	1254	107.5 %
40000	829	808	976	3806	102.6 %
60000	2090	1987	2231	9380	105.2 %
80000	1963	2947	3134	14380	66.6 %
100000	2250	2119	2269	days	106.1 %
120000	2857	2581	2737	days	110.7 %

Table II
RESPONSE TIME (SECONDS) WITH VARYING MEMORY SIZE ON THE WWW AND THE MOBILE USAGE DATA.

of that step depends on the number of objects and their average number of features. In Table II, we give the response times of the models we have implemented (ABS, Batch and SW). We also report the difference (percentage) between the time response of ABS and the best time response among the other models. For instance, on the WWW usage data, with a memory size of 10,000 transactions, we observe that the response time of ABS is 47% of Batch's response time (ABS is twice as fast). With the Mobile usage data, the response times of ABS and the best model are very close. With such a low difference in execution times, the criteria to chose a model should be the observed bounce rate and the clustering quality. From these points of view, ABS is better than the other models. Eventually, all the models have longer response times on the WWW usage data (compared to the Mobile usage data) since the number of features is 8,000 (versus 24 features in the Mobile usage data).

V. CONCLUSION

Lowering bounce rate is a critical issue for most Web sites. To that end, the best solution is obviously to understand the reasons for bounce rate and to enhance the site accordingly. However, the observed bounce rate might be higher than it really is in the original usage stream. As we have shown, this can be due to the model used for observing the stream. We have proposed ABS, a new model that allows to i) lower the observed bounce rate, ii) better represent the recent data in the stream and iii) avoid to break down the navigations represented in the model. Our experiments showed that our model allows a better representation of data streams while reducing the processing cost.

REFERENCES

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 81–92, 2003.

[2] Ramiz M. Aliguliyev. Performance evaluation of density-based clustering methods. *Inf. Sci.*, 179(20):3583–3602, 2009.

[3] Joong Hyuk Chang and Won Suk Lee. Finding recent frequent itemsets adaptively over online data streams. In *KDD '03: Proceedings of the ninth international conference on Knowledge discovery and data mining*, pages 487–492, 2003.

[4] Keke Chen and Ling Liu. He-tree: a framework for detecting changes in clustering structure for categorical data streams. *The VLDB Journal*, 18(6):1241–1260, 2009.

[5] C. Giannella, J. Han, J. Pei, X. Yan, and P.S. Yu. *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*. In H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha (eds.), Next Generation Data Mining. AAAI/MIT, 2003.

[6] Philipp Kranen and Thomas Seidl. Harnessing the strengths of anytime algorithms for constant data streams. *Data Min. Knowl. Discov.*, 19(2):245–260, 2009.

[7] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

[8] D. Sculley, Robert G. Malkin, Sugato Basu, and Roberto J. Bayardo. Predicting bounce rates in sponsored search advertisements. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1325–1334, 2009.

[9] Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu. A Regression-Based Temporal Pattern Mining Scheme for Data Streams. In *VLDB*, pages 93–104, 2003.

[10] Li Tu and Yixin Chen. Stream data clustering based on grid density and attraction. *ACM Trans. Knowl. Discov. Data*, 3(3):1–27, 2009.

[11] Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S. Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. *ACM Trans. Knowl. Discov. Data*, 3(3):1–28, 2009.

[12] Wikipedia. Bounce rate — wikipedia, the free encyclopedia, 2009. [Online].

[13] Raymond Chi-Wing Wong and Ada Wai-Chee Fu. Mining top-k frequent itemsets from data streams. *Data Min. Knowl. Discov.*, 13(2):193–217, 2006.

[14] Xiangliang Zhang, Cyril Furtlehner, and Michèle Sebag. Data streaming with affinity propagation. In *ECML/PKDD (2)*, pages 628–643, 2008.