ARTICLE IN PRESS

Information Sciences xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

The anti-bouncing data stream model for web usage streams with intralinkings

Chongsheng Zhang [a,*], Florent Masseglia [b], Yves Lechevallier [c]

[a] Henan University, 475001 Kaifeng, China
[b] INRIA and LIRMM, 34392 Montpellier Cedex, France
[c] INRIA Rocquencourt, 78153 Le Chesnay cedex, France

ARTICLE INFO

Article history:
Received 4 August 2012
Received in revised form 9 March 2014
Accepted 15 March 2014
Available online xxxx

Keywords:
Web usage streams
Intralinking records
Data stream models
Bounce rate

ABSTRACT

Web usage mining is a significant research area with applications in various fields. However, Web 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 show that data intralinkings, i.e. a usage record (event) may be associated with other records (events) in the same dataset, are common for Web usage streams. Therefore, in order to have a more authentic grasp of Web usage behaviors, the corresponding data stream models for Web usage streams should be able to process such intralinkings. We study the important consequences of the constraints and intralinkings, through the "bounce rate" problem and the clustering of usage streams. Then we propose the user-centric **ABS** (the **A**nti-**B**ouncing **S**tream) model which combines the advantages of previous models but avoids their drawbacks. First, ABS is the first data stream model that is able to seize the intralinkings between the Web usage records. It is also the first user-centric data stream model that can associate the usage records for the users in the Web usage streams. Second, owing to its simple but effective management principle, the data in ABS is available at any time for analysis. Under the same resource constraints as existing models in the literature, ABS can better model the recent data. Third, ABS can better measure the bounce rates for Web usage streams. We demonstrate its superiority through a theoretical study and experiments on two real-world data sets.

1. 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 $BR = \left(\frac{T_o}{T_v}\right)$ 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 onto other pages within the same site* [46]. Bounce rate is very important for usage

* Corresponding author.
E-mail addresses: Chongsheng.Zhang@yahoo.com (C. Zhang), Florent.Masseglia@inria.fr (F. Masseglia), Yves.Lechevallier@inria.fr (Y. Lechevallier).

http://dx.doi.org/10.1016/j.ins.2014.03.089
0020-0255/

Table 1
Streaming events (1), their corresponding navigations (2) and the navigations in B_n, the batch that contains the last six events (3).

(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						

analysis and most commercial websites would like to lower it.[1] Actually, let us consider a website C. When a user clicks through a paid advertisement 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 advertisements. 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, for instance, 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.*

1.1. Measuring the bounce rates for web usage 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 is a tuple $e_i = \langle uid, time, page \rangle$, where i is the identifier of e, 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 $e_x = \langle u_j, t_k, p_l \rangle, k \in [0, t]$, where $u_j = u_i$.

According to Definition 1, an event data stream contains the user 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 this paper, 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 model is based on batches of events [17,23,29], 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 (which will be detailed in Section 2) and its principle on a toy dataset.

Example 1. Let us consider the events given in Table 1 (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 1 (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 do not 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 1 (3). The main observation is that the navigations of users u_1 and u_3 are truncated. This has important consequences on the data analysis:

[1] Meanwhile, some websites will not 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).

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

1. 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).
2. Let us consider the clusters that would be obtained in B_n and the ones obtained in the total data. 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 $\{c\}$. Obviously, the clustering result on B_n is very different from the one obtained on the whole data (or a larger set of recent data).
3. Truncated navigations cannot be reliably retrieved.

The main point from Example 1 is the observed bounce rate according to existing models in the literature is higher than the real one. This is due to the fact that existing data stream models are based on removing obsolete events. Actually, the data stream models from the literature have a common point: *they sort the events by timestamp and they maintain a maximum number of events in the model*. Therefore, the navigations of many 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 keep such a valuable information.

1.2. Data intralinking in event streams

In many applications such as Web usage analysis and online shopping/auction systems, records that are already in the event streams may be "intralinked" (i.e. associated) with others arriving before or after them. For instance, when users access internet through mobile phones, due to the operation inconvenience, they are unable to visit several Web pages within a very short period. Their requests are thus separated into several records occurring at non-successive time steps. In the case of Web usage, over time, new users are added. Moreover, when existing users request new URLs, the URLs are added to the tuples of these users. Let the Web page request list of a user be the behavior summary of this user, we see that the behavior summaries of the users in usage streams are evolving along time. The intralinkings of records in the Web usage streams are therefore very important for investigating user behaviors.

In essence, an event stream with data intralinkings is an event stream having records with user identifiers, with the full usage records of each user being distributed as separate records inside the streams. We would like to clarify that "an event stream with data intralinkings", is not a very novel definition, since a lot of the data (e.g. Web usage data) for mining and management are similar to the one used in this paper. However, existing methods often assume that the raw data has been pre-processed before analyzing the data. For instance, in Web usage mining, identifying users and user sessions, and defining transactions are both needed before association rule mining [14,33]. Yet such data preparation methods, were off-line ones and do not apply to data streams.

Actually, there are two levels of data linking–interlinking and intralinking. Both linkings are important. The interlinking of Web data resources has received a lot of research focus [4,5,22,16,38,9], yet few investigations have been made in the intralinking of records within a single Web usage stream. For Web usage data, the interlinking of different Web data resources enables the interconnection of information from different usage data; while the intralinking of Web usage records within the same usage dataset identifies the connections between the data records so that we can make more complete and authentic descriptions of user behaviors. Existing data stream models, however, have not taken into account such data intralinkings. Hence, the need of a new data stream model that is able to support the processing of the intralinkings between records in Web usage streams.

1.3. Contributions

We introduce the user-centric **ABS** (**A**nti-**B**ouncing **S**tream) model, a new model relying on a novel point of view. The goal of ABS is to maintain a reliable representation of the recent data in the stream while avoiding to break down the navigations. Using ABS, we can

1. connect and group the usage records by their users. Without ABS, these usage records would otherwise be inappropriately processed as independent ones by existing data stream models.
2. maintain a set of events that represents the recent data of the stream while avoiding to split down the navigations.
3. lower the observed bounce rate (avoid the drawback of existing models, as the one illustrated in Example 1).

We show through theoretical and experimental analyses that owing to its simple but effective management principle, ABS lowers the negative influence of a restricted observation window. It gives a more authentic summarization of user behaviors and a more reliable measurement of bounce rate. In this work, we consider the case of bounce rate and clustering, but ABS should have the same advantages for other techniques such as pattern extraction [7,13], behavioural modeling [6,26,15], recommendation [40] or stream analysis in social networks [30].

In this paper, we extend the work of [48] and elaborate in depth about the data-intralinking problem in Web usage streams and the solutions, including detailed techniques and experimental results.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

The rest of the paper is organized as follows. We introduce the related work in the following section, then present our model and theoretical studies in Section 3. Next, we analyze the complexity of different data stream models in Section 4 and show the experimental results in Section 5. We conclude the paper in Section 6.

2. Related work

In this section, we first give an overview of Web access log analysis in the case of usage mining. Then, we focus on stream mining, since this work is towards Web usage streams. Finally, we discuss existing work on measuring the bounce rates of Web usage streams.

2.1. Web log analysis

The initial function of the Web access logs was to log all the requests, detect the errors (for instance 404) and fix them. Usually, the log files record the information about each access to the website that includes the IP address of the machine used to access the site, the access time, the resource (URL) requested and the result of this request. Soon, the size of the Web access logs grew so large that analyzing them through manual check became very difficult. In this context, people began to use data mining techniques to automatically extract patterns and knowledge from the log files. That was the beginning of Web log analysis.

In [35], the issue on the automatic adaptation of websites is investigated. The goal is to improve the arrangements of the websites by exploiting Web access patterns obtained from Web usage data. In [27], the authors summarize existing Web mining techniques and highlight the relationship between such techniques and the content, structure and usage of Usenet services. Then, a discussion on data mining patterns and knowledge extracted from those sites and how they correspond to users' needs and expectations is given.

In [45], the authors adopt the general point of view of Information Systems and propose a study on techniques dedicated to improving users' navigations, interface customization and adaptation. Their initial work focuses on Web site design by means of Web log data mining. Their approaches allow discovering patterns from such logs to achieve the automatic creation of index pages. In [18], the authors propose a Web log clustering approach. They consider the web log sessions as transactions and adapt *K*-means to cluster such transactions.

All the above approaches handle static data stored in logs by information systems. However, these static methods do not fit to streaming data, where users' requests arrive on the fly and data analytics need to be performed in real-time on such streaming data. In this case, data stream mining techniques are needed. Below, we introduce techniques on data stream mining.

2.2. Data stream mining

2.2.1. Data stream models

The work in [23] gives an interesting comparative study of Batch and **S**liding **W**indow (SW) models. The authors propose two approaches designed towards anytime algorithms and their exploitation in data streams. The principles of the Batch model [17,23,29] and the sliding window model [42,23,3] are illustrated by Fig. 1 and 2.

When an event data stream is observed through batches (C.f. Fig. 1), the navigations corresponding to the events of that batch are built and processed. With sliding windows (C.f. Fig. 2) we need to maintain a list of current events in the model. The main difference from the Batch model lies in (i) the frequent updates (each new transaction has to be added to the

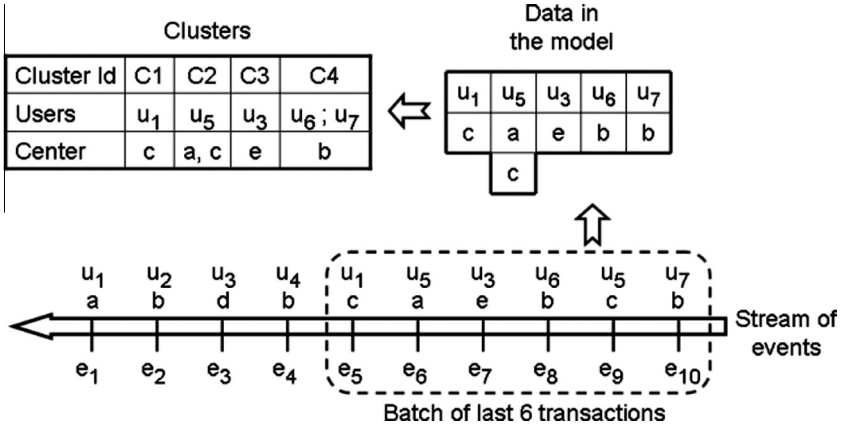

Fig. 1. Representing the data stream with batches of six events. The users' navigations are built and analysed for each batch.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

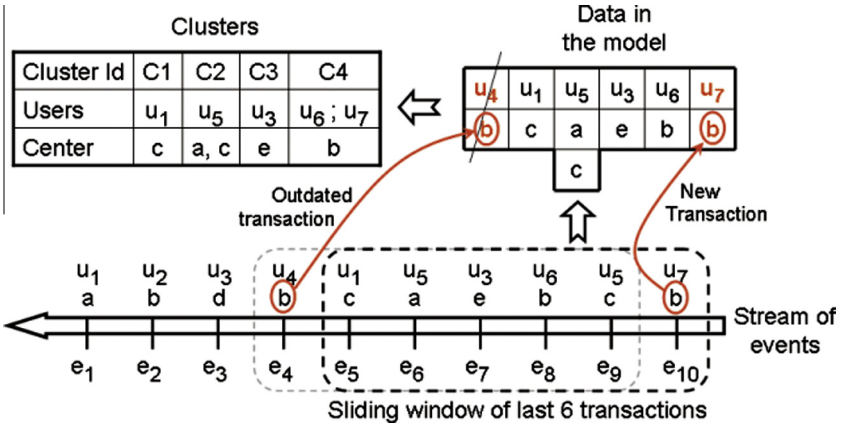

Fig. 2. Representing the data stream with a sliding window of six events. The users' navigations might be added, updated or removed from the model.

corresponding navigation, and removing the oldest event requires to update or delete the corresponding navigation) and (ii) the availability of the model at anytime. However, when a batch is complete and ready for analysis at time *t*, the sliding windows model at time *t* contain the exact same events (and navigations).

Let us also mention some other data stream models such as the landmark windows [28], where the analysis is maintained for a window ranging from one fixed point in the past to the current time, and the decaying factor [10,12] which aims to give higher weight to recent events in the analysis.

2.2.2. Clustering data streams of feature vectors

Most papers on data stream mining have considered streams of feature vectors [1,43,12,44,47], 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 this paper, the users' requests arrive in parallel and event after event). In [1], the authors introduced the concept of micro-clusters and the CluStream algorithm. A micro-cluster is a data structure that maintains statistical information about the data locality. It contains feature vectors that capture, for each record in a micro-cluster: the sum of squares of the data values, the sum of the data values, the sum of squares of the time stamps, the sum of the time stamps and the number of values. Such a structure allows for easy retrieval of the centroid of a micro-cluster and for fast update of the features (it has an additive property that make it suitable for data streams). Micro-clusters can then replace the original values at a higher granularity and be used in a global, off-line, clustering step when decided by the end-user.

2.2.3. Itemset mining over data streams

Itemset discovery is a major concern in data mining, because it is the basis for correlation analysis and association rules discovery. It has been thoroughly studied in the literature with different principles [17,24,8,41]. The work in [17] exploits the FPGrowth algorithm in a batch environment and uses a decaying factor to manage the history of extracted patterns. The algorithm is based on the Batch model, for each batch, an FP-tree is built [21] and analyzed for frequent pattern extraction. Then, the extracted patterns for each batch are stored in a global tree where the support of this pattern is maintained with a tilted time window (allowing for multiple granularity of representations). In [29], the authors propose to extract sequential patterns with a batch principle and an alignment technique. In [24], the authors propose a sliding window structure and encode the transactions in a bit-sequence representation. Thanks to this representation, the left bit-shift technique can be used to shift the window efficiently. In [8], the model is different from those presented above (*i.e.* sliding window, batches or landmark) since the authors propose to find the period of time during stream processing that optimizes the support of an itemset. Their goal is therefore to find both the itemsets and the periods when the itemsets are frequent. [11] proposes to discover frequent sequential patterns with an algorithm based on prefixspan [34] and to optimize the process by incorporating prior knowledge in the discovery technique.

2.2.4. Data streams with tuple revisions

In this paper, we consider the case of data streams where the events belong to global objects. 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.

Viewing the Web usage streams with data intralinking from the user perspective, we see that the usage record set describing a user's Web usage behavior can be revised in data streams. The tuple revision problem in data streams has already been studied in the literature [37,32,20,31]. But all these work deal with query processing.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

- "...Many data stream sources (e.g., commercial ticker feeds) issue "revision tuples" (revisions) that amend previously issued tuples..." Esther Ryvkina et al. in [37].
- "...A data stream is an append-only sequence of time-stamped items...While this is the commonly accepted definition, there are more relaxed versions; for example, revision tuples, which are understood to replace previously reported data, may be considered so that the sequence is not append-only...", Tamer Özsu and Patrick Valduriez in [32].

We would like to clarify that tuple revisions are different from tuple updates. The main problem for the latter is how to keep the patterns or query answers up-to-date when new tuples come and/or old tuples expire [19]. In [37], Esther Ryvkina et al. clarified the definitions of tuple revisions and updates, and pointed out the differences between them: "...Note that revisions .. are not the same as updates...Revisions are corrections as they invalidate previously processed inputs, and by implication, all query results that were produced from them. On the other hand, updates close the time interval during which previously processed inputs were valid, and therefore do not invalidate any previously output query results...".

To minimize the staleness of query results over streams with revision tuples, Alexandru Moga et al. proposed an efficient storage-centric framework for load management over the streams [31]. They proposed two respective storage models for streams with keys having uniform revision frequencies, or with non-uniformly updated keys.

Parisa Haghani et al. studied the problem of continuous top-k query processing over multiple non-synchronized streams, where the attributes of an object arrive separately in different streams such that the value of the object is incomplete [20]. Exploiting the dominance relationship of the objects in terms of the estimated aggregate value interval and time to life, the authors proposed an approximate algorithm that provides highly accurate results and reduces greatly the number of candidates for top-k objects.

2.2.5. Scalability issue in data stream mining

Scalability is a major concern for data stream mining methods. Owing to the speed and amount of the streams, there are the two main requirements on data streams mining algorithms. First, the algorithms should be able to process the data in real-time. Second, the algorithms have to work with limited memory. It might be possible to store the whole streams on disk, but the processing power may be inadequate to analyze the large amount of streaming data in real-time. Therefore, scalability is a very important evaluation criteria for data stream mining algorithms.

In [25], the authors propose a monitoring system for traffic stream. It is based on distributed roadway traffic mining and prediction. The data stream of current traffic is redirected to a central node, where a pattern mining algorithm is applied as a training phase. Then, the discovered patterns are communicated to the sensors on the roads where traffic predictions are calculated. In case of abnormal traffic, an alarm is raised. The scalability of the system has been studied in the experiments, with respect to the number of nodes (sensors).

The authors of [36] propose Astrolabe, an information management service that continuously computes data summaries by means of aggregation operators. Those operators are applied on-the-fly and considered as a type of data mining capability. Astrolabe is based on a distributed structure and a gossiping protocol between hosts. Scalability is validated through experiments where the system performance is tested with respect to the number of participants.

2.3. Measuring the bounce rates for web usage streams

Bounce rate is a recent, though important, metric that is relatively unstudied in the literature. It is a significant metric for measuring users' experiences when they navigate a website. Commercial companies usually refer to this metric when optimizing their websites. In [39], 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.

To summarize this section about related works: in the literature on data streams we mainly find papers dedicated to a number of models (*e.g.* Batch, Sliding Window, decaying factor), to clustering data streams of feature vectors and to frequent pattern mining in data streams of events, to update the results of the monitoring queries over data streams with tuple revisions or updates. However, there is no comparative study on models for event data streams nor proposals for lowering the observed bounce rate in usage data streams.

3. ABS: the anti-bouncing stream model

As stated before, existing data stream models share some characteristics:

1. They sort the events by timestamp, and their observation windows depend on that principle.
2. They cut the users' navigations and do not reflect the recent content of the stream beyond the observation window.

So, how to keep an accurate and seamless representation of the recent events? How to make the model available for any time analysis, while avoiding the drawbacks of existing models? How to be as straight as a batch processing while maintaining cohesive navigations? And, ideally, how to answer all those questions at the same time?

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

3.1. A new management and pruning principle

The key idea of ABS 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.

3.1.1. 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 does not 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, as stated by Property 1.

Property 1. *At any time a, let u_i be a user at the ith position in the model, then the following property holds:*

$$\forall e_x = \langle u_i, t_k, p_l \rangle,\ \forall j < i,\ \exists e_y = \langle u_j, t_r, p_m \rangle, r > k$$

(where $j < i$ means that u_j is before u_i in the list, and $r > k$ means that t_k is older than t_r).

Proof 1. Let us consider that Property 1 does not hold for a value of $j < i$ (*i.e.* we consider that $\nexists e_y = \langle u_j, t_r, p_m \rangle$ with $r > k$). Therefore, $\exists e_z = \langle u_i, t_w, p_v \rangle$ such that $w > r$. Consequently, at time w, u_i has been moved from its current position to the head of the list. Afterwards, no event has been added to u_j from time w to time a (otherwise the property $\forall j < i,\ \exists e_y = \langle u_j, t_r, p_m \rangle,\ r > k$ would hold). Then, the order between u_i and u_j did not change at time a and u_i is before u_j, which does not correspond to the statement of Property 1. □

3.1.2. 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 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:

1. The number of events in ABS and Batch or SW is approximately the same at any time. Actually, the number of pages contained in the user removed from ABS is negligible with regards to the number of events maintained in the model (the larger the maximum number of events allowed in main memory, the more similar the amount of events in each model at any time).
2. The pruning step of ABS is faster than SW, since we do not need to retrieve any user for a removed event. We just remove the user at the end of the structure. Moreover, that operation is not required for each new event (only when the model has reached the maximum number of events).

Property 2. *Let k be the maximum number of events and $|M|$ be the size of model M. $\forall t > k,\ |SW| = k$ and $k - r < |ABS| \leqslant k$ with r the number of events in the navigation of u_i, the last user removed from ABS.*

The proof of Property 2 is straightforward since it corresponds to the principle of ABS (C.f. Fig. 5). *Owing to this straightforward pruning principle, ABS does not need to maintain the list of monitored events for user removal.*

3.1.3. Illustration

Let us consider again the dataset in Table 1 with a memory size of six events. It is not relevant to directly describe the content of ABS for the last six events, since it strongly depends on the arriving order of the events in the stream. We need to consider each event, from the beginning, in order to describe the evolution of the model over this toy dataset. From e_1 to e_4 the users are added to the model, from the oldest to the newest one, as illustrated by the first table in Fig. 3. After e_5, the

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

navigation of user u_1 is updated with a new page (c) and u_1 is moved from its current position (oldest) to the head of the structure (most up to date user). Next (event e_6) a new user (u_5) is added to the model. After e_7, the navigation of u_3 is updated and u_3 becomes the most up to date user. Furthermore, since memory is limited to six events and the model contains seven events at that step, we have to remove a user. The last user, u_2 is removed with all of its navigation. The process goes on until e_{10}.

Finally, the data in ABS after e_{10} with the example dataset is illustrated by Fig. 4, which can be compared to Figs. 1 and 2.

3.1.4. Algorithm

As can be seen in the pseudo-code of ABS (Fig. 5), 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, ABS 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 3.2.

3.2. Does ABS avoid to break 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 $f(n, \eta) = \frac{\eta^n . e^{-\eta}}{n!}$, where η is a positive number, equal to the expected number of occurrences that occur during the given interval time $[0, T]$. For instance, if the events occur seven times per minute in average, and we are interested in the number of events occurring in a period of 10 min, the model would be a Poisson distribution with $\eta = 70$.

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:

$$P[N_i = k] = \frac{(\lambda_i T)^k \cdot e^{-\lambda_i T}}{k!}. \tag{1}$$

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 $t_b \in [0, T]$, the event (b, t_b, p_{t_b}) occurs, associated with b, and the most recent event associated with a is $(a, 0, p_0)$. 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. $P[\tau_a \leqslant T / \tau_a > t_b]$, this

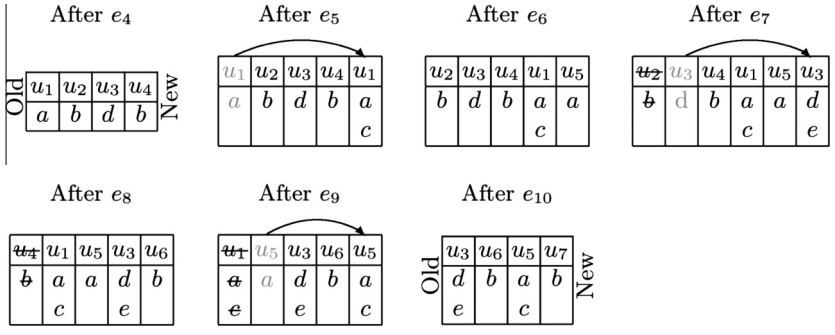

Fig. 3. Principle of ABS detailed for six events in our toy example.

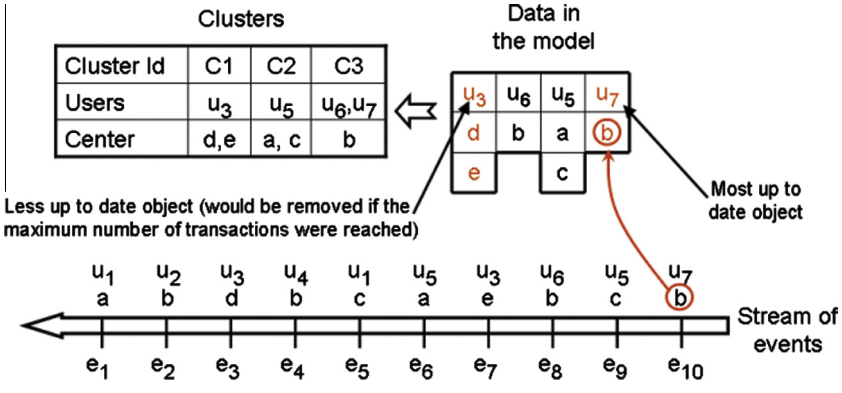

Fig. 4. A data stream with ABS on six events. The less up to date user is removed when the model is full.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

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

Fig. 5. Algorithm for the ABS model.

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 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 $\lambda_b = 0.5$. Fig. 6 gives the values of P_a and $P_{\neg b}$ for $T - t_b \in [0..4]$. In this case, at time $T - t_b = 2$, 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, at time $T - t_b = 4, P_{\neg b} \simeq 0.15$ is small 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 $U(t') = \{u \in U / \exists (u, t, p_t)\}, t \in [0, t']$ 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 $P[\tau_a \leqslant T - t'] = 1 - e^{-\lambda_a(T-t')}$ must be compared to $min\{P[\tau_u \leqslant T - t_u / \tau_u > t' - t_u] = e^{-\lambda_u(T-t')} / u \in U(t')\}$ 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, ABS 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 $E\left[\sum_{u \in U} N_u\right] = \sum_{u \in U} \lambda_u T$. Consequently, the value T of the period $[0, T]$ can be estimated by:

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

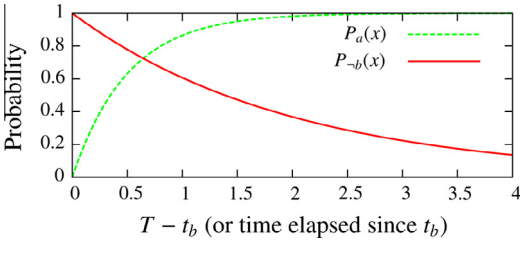

Fig. 6. Probabilities that an event occurs for of user a and does not arrive for user b.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

3.3. Discussion

The advantage of SW over Batch is to make its data available at any time. On the other hand, this feature is costly, both from the CPU and memory points of view. Actually, SW needs (i) to keep the list of events in the observation window and (ii) to update the navigations both upon arrival and removal of events. Modeling the stream with Batches of events is much simpler, but the data is available only after the batch is fulfilled. Furthermore, both models, as well as (to our knowledge) other existing models in the literature are not designed towards a lower bounce rate and a seamless representation of the recent data in streams of usage. ABS has the same advantages as Batch and SW since it allows for anytime analysis while relying on a fast and low memory consumption management principle. Besides, as shown in 3.2, ABS is well suited for usage data streams where the navigations should not be interrupted in the model.

As discussed above, with ABS the navigations can expand up to a number of pages that is not attained by other models in the literature. In Fig. 7, we compare the models on the toy example after reading e_{10}. The clusters' centers in ABS are exactly the same as those obtained on the original data. Meanwhile, Batch or SW give very different centers. Furthermore the objects in ABS and the whole data are classified in the same clusters. This is not the case for Batch/SW, where u_1 and u_5 should be in the same cluster. The bounce rate (column 'BR') observed in ABS (0.5) is very close to the real bounce rate in the original data (0.57), which is not the case for Batch/SW (0.8). The same remark can be made for the average length of the navigations in each model.

4. Complexity analysis

In this section, we discuss how to utilize the Batch model, the Sliding Window (SW) model and the ABS model to summarize event streams with intralinkings and estimate the corresponding costs. The cost is measured in terms of CPU cost and memory usage.

4.1. CPU cost

We use the Batch model, the SW model and the ABS model to summarize event streams with intralinkings. To analyze the efficiency of each candidate model, we consider the price to pay for the operations of *search*, *insertion*, *deletion*, and *update* on the model as computational overhead.

The Batch model waits until the buffer is full of events. When a new event arrives, we insert it to the event buffer. When the buffer is full, to summarize events we need to build a map to identify and merge the events with identical identifiers (C.f. Fig. 1). For each event, the *total cost* is the *search cost* plus the *insertion cost* and the *updating cost*.

To summarize event streams with intralinkings, the sliding window model needs to keep the events in a sliding window, meanwhile it needs to maintain and update the summarized objects in extra data structures on the fly such that the arranged objects are always available for analysis (C.f. Fig. 2). Each time a new event arrives, (1) we should not just insert it in the sliding window, but also search and update the corresponding objects and (2) when the sliding window is full, the oldest event should be removed; we also have to *search* the object that contains the oldest event to be removed and *update* the objects correspondingly.

Using ABS model, when a new event comes, we look up the map to check whether the according identifier already exists in the map. As each identifier is unique, the cost for the search over the map is O (1). Then, we will have three possible operations on the data associated with the identifier: *insertion, update* and *deletion*. If it is a new identifier, we will insert a key in the map. Additionally, we will create a new object then insert it in the *tail* of the list. The *total cost* for insertion is the *search*

	Data in the model	Clusters	*BR*	Avg. length
ABS	u_3 u_6 u_5 u_7 d b a b e c	Users u_5 u_6, u_7 u_3 Center a, c b d, e	0.5	1.5
Original data	u_1 u_2 u_3 u_4 u_5 u_6 u_7 a b d b a b b c e c	Users u_1, u_5 u_2, u_4 u_6, u_7 u_3 Center a, c b d, e	0.57	1.42
Batch/SW	u_1 u_3 u_5 u_6 u_7 c e a b b c	Users u_1 u_3 u_5 u_6, u_7 Center c e a, c b	0.8	1.2

Fig. 7. ABS and Batch/SW compared to the original data (navigations, clusters, bounce rate and average navigation length).

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

Table 2
Memory cost of three candidate models.

Candidates	Memory for modeling	Memory for data	Overall memory usage
Batch	map size	data size	map size + data size
SW	map size + data size	data size	map size + 2* data size
ABS	map size	data size	map size + data size

cost plus the *insertion cost*. If it is an existing identifier, then we will update the according object in the list and move it to the *tail* of the list. If we are going to remove an object from the list, we simply delete it from the list without looking up the map, since we always remove the oldest one from the *head* of the list.

In summary, when a new event arrives, the insertion and update costs are the same for the three models. Although the Batch model does not have deletion cost, the objects cannot summarized in real-time but delayed until the new batch is full. Comparing to the sliding window model, for each new event, ABS model requires less *deletion* than the sliding window model. Moreover, when deletions are needed, unlike the sliding window model, ABS does not result in extra *search* and *update* cost. Thus, the deletion cost of ABS is much lower than the SW model.

4.2. Memory usage

The memory cost for each model is listed in Table 2. The memory usage of Batch model contains two parts. One part is the memory for keeping the batch of events. The other part is the memory required for establishing a map to identify and merge the events with the same identifiers. In sliding window model, we have to not only keep the events in the sliding window, but also maintain a map and extra arranged objects for all the identifiers. As a result, the memory usage of the SW model is two times of the data size added by the map size. In ABS model, we need a map to identify and summarize events with identical identifiers into objects, the objects are directly linked in a list. As we do not keep other extra data structures, the total memory usage is the size of the data added by that of the map.

Therefore, that Batch model and the ABS model have the same memory usage. The SW model, however, uses much more memory than the ABS model and the Batch model.

5. Experiments

Our goal is to evaluate algorithms from three points of view: bounce rate (Section 5.1), clustering results (Section 5.2) and time response (Section 5.3). We have implemented and tested three models (Batch, SW and ABS) on two datasets. The first dataset contains 3 months of requests from Orange's subscribers[2] to their mobile portal. The original file is 7 GB and contains 19 millions requests. The second dataset comes from the WWW access log file of INRIA Sophia-Antipolis 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. Fig. 8 illustrates the principle of our reference model. The reference (in red) is based on the principle of a batch that is $\times$times larger than the evaluated model. With the same number of events, ABS (dashed lines) is expected to contain navigations that spread over a larger period than Batch (in black). During the stream processing, we randomly chose 200 random measure points. 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). As explained in Section 2, when a batch is ready for analysis, the data of SW at that point in the stream is the same. Therefore, our measure often compare ABS to one model called Batch/SW when it is clear from the context.

5.1. 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 Fig. 9 and 10 (left) clearly show a lower bounce rate for ABS. For instance, with a memory size of 120,000 events, the bounce rate on mobile usage 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 on the right in Figs. 9 and 10. We can observe that, unlike Batch/SW, ABS is very close to the reference in the mobile data, whatever the memory size. In the WWW data, ABS is even above the reference, which means that ABS is able to represent navigations that go beycond the maximum affordable reference on this machine. We will give details about this point in Section 5.2. We would like to note that we do not give here the real value of bounce rate and average length of navigations in the Orange usage data (these statistics are not publicly available). These numbers are obtained from observation windows on a biased sample.

[2] Orange is a major mobile operator in Europe.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

Fig. 8. Comparison of the evaluated models to a reference.

5.2. Cluster validation

We compared the clustering results on the data of ABS and Batch/SW with the reference. The clusters in these experiments come from an implementation of AP [49] applied at each random step. For a comparison, we only keep the users at the intersection between the evaluated model and the reference. For instance, with Fig. 7 if we wanted to evaluate the clusters of ABS compared to the clusters of the reference, we would keep only 4 users (u_3, u_5, u_6 and u_7). Afterwards, we measured the purity and entropy as described in the following. Let C be the set of clusters obtained on a model, C_i be the ith cluster, and R be the set of clusters obtained from the reference.

The entropy [2] value addresses the consistency of clustering C with R. The larger the entropy, the worse the clustering compared to the reference. If the objects which originally belong to the same clusters in C_i are scattered across the clusters in R, the entropy of this cluster will be high. Let n be the total number of objects, $|R|$ be the number of clusters and $C'_1 \dots C'_{|R|}$ are clusters for R, the entropy of a cluster C_i in C is given by: $E(C_i) = -\frac{1}{log(|R|)} \sum_{j=1}^{|R|} \frac{|C_i \cap C'_j|}{|C_i|} log\left(\frac{|C_i \cap C'_j|}{|C_i|}\right)$. The global entropy for clustering C, is sum of the entropy values of all the clusters: $E(C) = \sum_{i=1}^{|C|} \frac{|C_i|}{n} E(C_i)$.

Purity [2] is another consistency measure of a clustering. It allows to check if a set of objects that belong to the same cluster in R also belong to the same cluster in C. The purity of a cluster is given by: $P(C_i) = \frac{1}{|C_i|} max|C_i \cup C'_j|,\ j = 1, 3, \dots, |R|$. The higher the purity, the better the clustering. The global purity of C is the weighted sum of the individual purities: $P(C) = \sum_{i=1}^{|C|} \frac{|C_i|}{n} P(C_i)$.

Figs. 11 (left) gives the average purity of the clusters on ABS and Batch/SW compared to the reference with the mobile data. We can observe that ABS has a better entropy than ABS 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. Since, the average entropy value (average of all random points) does not give enough details, we propose a diagram with box plots in Fig. 11 (right). This diagram gives

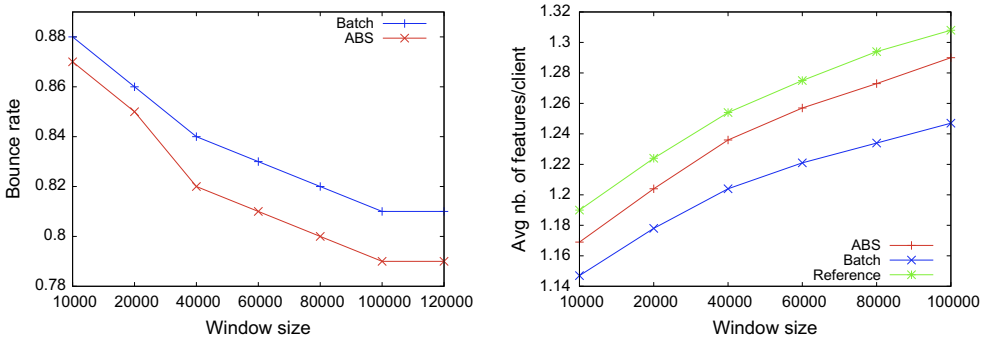

Fig. 9. Average bounce rate (left) and average length of navigation per user (right) in ABS and Batch/SW on the mobile usage data.

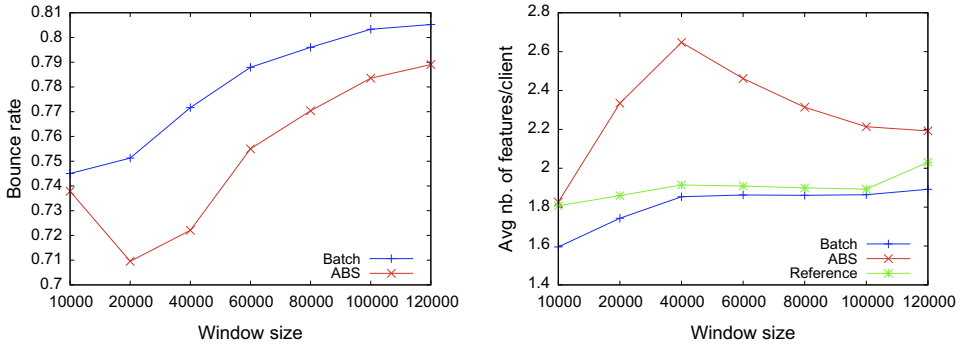

Fig. 10. Average bounce rate (left) and average length of navigation per user (right) in ABS and Batch/SW on the WWW usage data of INRIA Sophia-Antipolis.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

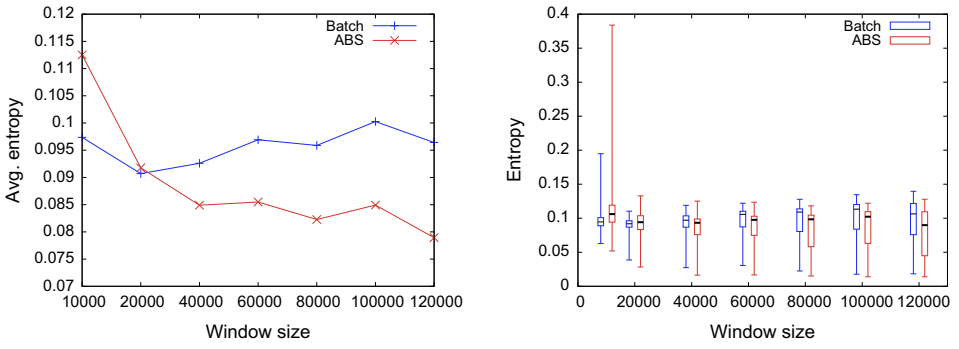

Fig. 11. Average entropy (left) and detailed entropy (right) on the mobile usage.

the minimum, first quartile, median, third quartile, and maximum entropy values for the clustering on ABS and Batch for all the random points (given a memory size). For instance, when the memory is set to 10,000 the maximum entropy of a clustering on ABS is 0.38 (and 0.2 for Batch/SW). In this case, the detailed results are close to the average but, as we will see in the following, that is not always the case. Fig. 12 (left) gives the average purity of the clustering of ABS and Batch with the mobile data. Once again, the higher the memory size, the better the clustering of ABS. We can see the detailed values in Fig. 12 (right). It is interesting to observe some outlying results, such as a minimum purity of 0.65 with a memory of 80,000 events. A purity of 0.65 is not a good result. Meanwhile, let us analyze that result. First, this is an outlier (and does not reflect the average trend). Second, we believe that our reference model does not allow to really evaluate how much ABS can spread its navigations over long periods. Let us consider $u_1 = (b, c)$ and $u_2 = (a, b, c)$, two users and their navigations. If u_2 has requested a very early, then Batch, SW and the reference will not keep a in his navigation. Therefore, they will group $u_1 = (b, c)$ and $u_2 = (a, b, c)$ in the same cluster when ABS will separate them. Hence, a lower purity of ABS in that case. A similar example could show that, sometimes, the reference does not cover an observation window that is large enough for assessing the entropy of a clustering in ABS. For the same reasons, we can find a larger value of the average navigation length in ABS (Fig. 10), even compared to the reference. This shows that ABS may contain navigations that spread on larger periods than the reference. Therefore, our comparison protocol is not perfect but (i) it is reliable for most results and (ii) our reference model is the largest we can afford, given the characteristics of the datasets and the machine. Eventually, we observe that ABS has a lower purity on the WWW data, whatever the memory size. Unfortunately, as shown in Fig. 10 (right), the reference is sometimes limited compared to ABS for this dataset, making the results difficult to analyze (see Figs. 13 and 14).

5.3. Time response

The most time consuming part of ABS is the analysis (*i.e.*, the clustering). The complexity of that step depends on the number of objects and their average number of features. In Fig. 15, we give the response time 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 time 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. We also observe that the analysis time upon

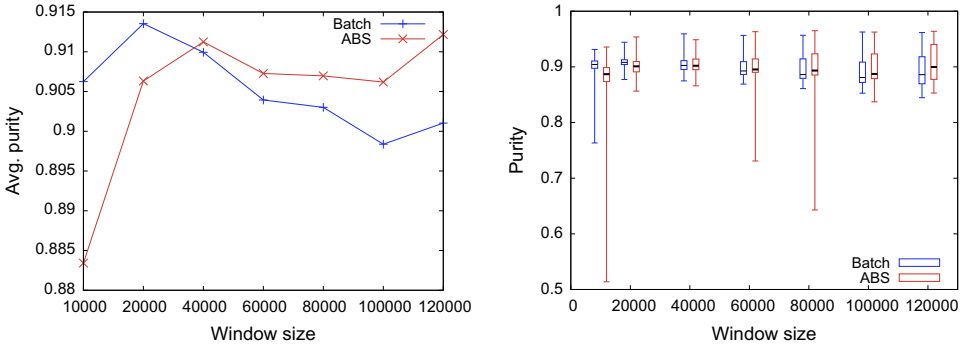

Fig. 12. Average purity (left) and detailed purity (right) on the mobile usage.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

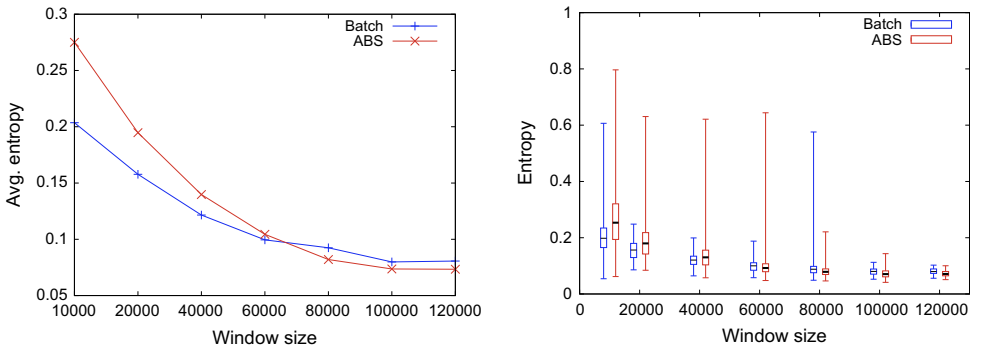

Fig. 13. Average entropy (left) and detailed entropy (right) on the WWW usage.

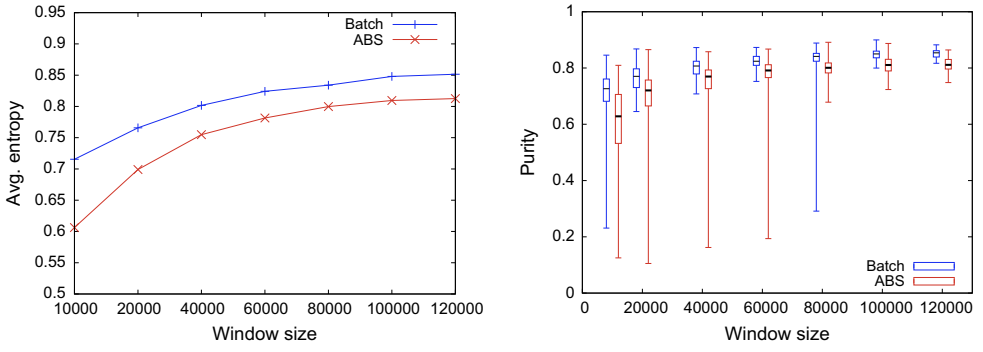

Fig. 14. Average purity (left) and detailed purity (right) on the WWW usage.

	WWW usage					Mobile usage				
M	ABS	Batch	SW	Ref	ABS Vs. Best	ABS	Batch	SW	Ref	ABS Vs. Best
10000	423	902	939	4439	46.9 %	164	159	294	505	103.1 %
20000	2591	3983	4200	17782	65 %	327	304	438	1254	107.5 %
40000	4163	5645	5686	24472	73.4 %	829	808	976	3806	102.6 %
60000	11241	15101	15168	73535	74.4 %	2090	1987	2231	9380	105.2 %
80000	9173	10587	10665	63163	86.6 %	1963	2947	3134	14380	66.6 %
100000	14991	23848	23751	days	62.9 %	2250	2119	2269	days	106.1 %
120000	14535	16806	16835	days	86.5 %	2857	2581	2737	days	110.7 %

Fig. 15. Response time (seconds) with varying memory size on the WWW and the mobile data.

each model increases linearly with the window size, but ABS uses less time than SW. From these points of view, ABS is better than SW for real-time data analytics. Finally, all the models have longer response times on the WWW usage data (compared to the mobile usage data) since the number of features is 8000 (versus 24 features in the mobile usage).

6. Conclusion

Lowering the 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 also point out that data intralinkings are common in Web usage streams, data stream models for Web usage streams should therefore be able to process such intralinkings. 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) better summarize the users' Web usage behaviors and avoid breaking down the navigations represented in the model. Our experiments showed that ABS allows a better representation of data streams while reducing the processing cost.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

Acknowledgements

This work was partially funded by ANR, Grant No. ANR-07-MDCO-008-01/MIDAS. The first author was also supported by the National Natural Science Foundation of China under Grants Nos. 61300215 and 61272545, and the Key Scientific and Technological Project of Henan Province under Grants Nos. 122102210053 and 132102210188.

References

[1] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu, A framework for clustering evolving data streams, in: VLDB '2003: Proceedings of the 29th International Conference on Very Large Data Bases, 2003, pp. 81–92.
[2] M. Ramiz, Aliguliyev, Performance evaluation of density-based clustering methods, Inf. Sci. 179 (20) (2009) 3583–3602. ISSN 0020-0255.
[3] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, Jennifer Widom, Models and issues in data stream systems, in: PODS '02: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2002, pp. 1–16.
[4] Christian Bizer, Tom Heath, Kingsley Idehen, Tim Berners-Lee, Linked data on the web (ldow2008), in: WWW, 2008, pp. 1265–1266.
[5] Christian Bizer, Tom Heath, Tim Berners-Lee, Linked data – the story so far, Int. J. Semantic Web Inf. Syst. 5 (3) (2009).
[6] Francesco Buccafurri, Gianluca Lax, Antonino Nocera, Domenico Ursino, Moving from social networks to social internetworking scenarios: the crawling perspective, Inf. Sci. 256 (0) (2014) 126–137.
[7] Luca Cagliero, Paolo Garza, Itemset generalization with cardinality-based constraints, Inf. Sci. 244 (0) (2013) 161–174.
[8] Toon Calders, Nele Dexters, Joris J.M. Gillis, Bart Goethals, Mining frequent itemsets in a stream, Inf. Syst. 39 (0) (2014) 233–255.
[9] Silvana Castano, Alfio Ferrara, Stefano Montanelli, Structured data clouding across multiple webs, Inf. Syst. 37 (4) (2012) 352–371.
[10] Joong Hyuk Chang, 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, 2003, pp. 487–492.
[11] Gong Chen, Xindong Wu, Xingquan Zhu, Sequential pattern mining in multiple streams, in: IEEE International Conference on Data Mining, 2005, pp. 585–588 (ISSN 1550-4786).
[12] Keke Chen, Ling Liu, He-tree: a framework for detecting changes in clustering structure for categorical data streams, VLDB J. 18 (6) (2009) 1241–1260.
[13] Ling Chen, Qingling Mei, Mining frequent items in data stream using time fading model, Inf. Sci. 257 (0) (2014) 54–69.
[14] Robert Cooley, Bamshad Mobasher, Jaideep Srivastava, Data preparation for mining world wide web browsing patterns, Knowl. Inf. Syst. 1 (1999) 5–32.
[15] Xiang Feng, Francis C.M. Lau, Huiqun Yu, Behavioral modeling with the new bio-inspired coordination generalized molecule model algorithm, Inf. Sci. 252 (0) (2013) 1–19.
[16] Alfio Ferrara, Andriy Nikolov, François Scharffe, Data linking for the semantic web, Int. J. Semantic Web Inf. Syst. 7 (3) (2011) 46–76.
[17] C. Giannella, J. Han, J. Pei, X. Yan, P.S. Yu, Mining Frequent Patterns in Data Streams at Multiple Time Granularities, in: H. Kargupta, A. Joshi, K. Sivakumar, Y. Yesha (Eds.), Next Generation Data Mining, AAAI/MIT, 2003.
[18] Fosca Giannotti, Cristian Gozzi, Giuseppe Manco, Characterizing web user accesses: A transactional approach to web log clustering, in: ITCC, 2002, pp. 312.
[19] Lukasz Golab, M. Tamer Özsu, Update-pattern-aware modeling and processing of continuous queries, in: SIGMOD Conference, 2005, pp. 658–669.
[20] Parisa Haghani, Sebastian Michel, and Karl Aberer, Evaluating top-*k* queries over incomplete data streams, in: CIKM, 2009, pp. 877–886.
[21] Jiawei Han, Jian Pei, Yiwen Yin, Mining frequent patterns without candidate generation, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00, ACM, New York, NY, USA, 2000, pp. 1–12 (ISBN 1-58113-217-4).
[22] Olaf Hartig, Christian Bizer, Johann Christoph Freytag, Executing sparql queries over the web of linked data, in: International Semantic Web Conference, 2009, pp. 293–309.
[23] Philipp Kranen, Thomas Seidl, Harnessing the strengths of anytime algorithms for constant data streams, Data Min. Knowl. Discov. 19 (2) (2009) 245–260.
[24] Hua-Fu Li, Suh-Yin Lee, Mining frequent itemsets over data streams using efficient window sliding techniques, Expert Syst. Appl. 36 (2, Part 1) (2009) 1466–1477.
[25] Ying Liu, Alok Choudhary, Jianhong Zhou, Ashfaq Khokhar, A scalable distributed stream mining system for highway traffic data, in: Proceedings of the 10th European Conference on Principle and Practice of Knowledge Discovery in Databases, PKDD'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 309–321.
[26] Mingqi Lv, Ling Chen, Gencai Chen, Mining user similarity based on routine activities, Inf. Sci. 236 (2013) 17–32.
[27] Giuseppe Manco, Riccardo Ortale, Andrea Tagarelli, Handbook of Research on Text and Web Mining Technologies: The Scent of a Newsgroup: Providing Personalized Access to Usenet Sites Through Web Mining, IGI Global, 2009, pp. 604–625 (Chapter XIX).
[28] Gurmeet Singh Manku, Rajeev Motwani, Approximate frequency counts over data streams, in: VLDB '02: Proceedings of the 28th International Conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 346–357.
[29] Alice Marascu, Florent Masseglia, Mining sequential patterns from data streams: a centroid approach, J. Intell. Inf. Syst. 27 (3) (2006) 291–307.
[30] Zachary Miller, Brian Dickinson, William Deitrick, Wei Hu, Alex Hai Wang, Twitter spammer detection using data stream clustering, Inf. Sci. 260 (2014) 64–73.
[31] Alexandru Moga, Irina Botan, Nesime Tatbul, Upstream: storage-centric load management for streaming applications with update semantics, VLDB J. 20 (6) (2011) 867–892.
[32] M. Tamer Ozsu, Patrick Valduriez, Principles of Distributed Database Systems, third ed., 2011 (ISBN 978-1-4419-8833-1).
[33] Balaji Padmanabhan, Zhiqiang Zheng, Steven O. Kimbrough, Personalization from incomplete data: what you do not know can hurt, in: ACM SIGKDD 2001, 2001, pp. 154–163.
[34] J. Pei, J. Han, B. Mortazavi-asl, H. Pinto, Q. Chen, U. Dayal, M. Hsu, Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth, in: ICDE '01: Proceedings of the 17th International Conference on Data Engineering, IEEE Computer Society, Washington, DC, USA, 2001, p. 215.
[35] Mike Perkowitz, Oren Etzioni, Towards adaptive web sites: conceptual framework and case study, Artif. Intell. 118 (2000) 245–275.
[36] Robbert van Renesse, Kenneth P. Birman, Dan Dumitriu, Werner Ogels, Scalable management and data mining using astrolabe, in: Revised Papers from the First International Workshop on Peer-to-Peer Systems, IPTPS '01, Springer-Verlag, London, UK, 2002, pp. 280–294.
[37] Esther Ryvkina, Anurag Maskey, Mitch Cherniack, Stanley B. Zdonik, Revision processing in a stream processing engine: a high-level design, in: ICDE'06, 2006, pp. 141–144.
[38] François Scharffe, Jérôme Euzenat, Melinda: An Interlinking Framework for the Web of Data, CoRR, abs/1107.4502, 2011.
[39] D. Sculley, Robert G. Malkin, Sugato Basu, 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, 2009, pp. 1325–1334.
[40] Yue Shi, Martha Larson, Alan Hanjalic, Unifying rating-oriented and ranking-oriented collaborative filtering for improved recommendation, Inf. Sci. 229 (0) (2013) 29–39.
[41] Bai-En Shie, Vincent S. Tseng, Philip S. Yu, Online mining of temporal maximal utility itemsets from data streams, in: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC '10, ACM, New York, NY, USA, 2010, pp. 1622–1626.
[42] Wei-Guang Teng, Ming-Syan Chen, Philip S. Yu, A regression-based temporal pattern mining scheme for data streams, in: VLDB, 2003, pp. 93–104.
[43] Li Tu, Yixin Chen, Stream data clustering based on grid density and attraction, ACM Trans. Knowl. Discov. Data 3 (3) (2009) 1–27.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089

[44] Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S. Yu, Kuan Zhang, Density-based clustering of data streams at multiple resolutions, ACM Trans. Knowl. Discov. Data 3 (3) (2009) 1–28.
[45] Daniel S. Weld, Corin Anderson, Pedro Domingos, Oren Etzioni, Krzysztof Gajos, Tessa Lau, Steve Wolf, Automatically personalizing user interfaces, in: IJCAI'03, 2003, pp. 1613–1619.
[46] Wikipedia. Bounce Rate. <http://en.wikipedia.org/wiki/Bouncerate> (last access date 03.03.14).
[47] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Mining top-*k* frequent itemsets from data streams, Data Min. Knowl. Discov. 13 (2) (2006) 193–217.
[48] Chongsheng Zhang, Florent Masseglia, Yves Lechevallier, Abs: the anti bouncing model for usage data streams, in: IEEE ICDM '2010: Proceedings of the 2010 Tenth IEEE International Conference on Data Mining, 2010, pp. 1169–1174.
[49] Xiangliang Zhang, Cyril Furtlehner, Michèle Sebag, Data streaming with affinity propagation, in: ECML/PKDD (2), 2008, pp. 628–643.

Please cite this article in press as: C. Zhang et al., The anti-bouncing data stream model for web usage streams with intralinkings, Inform. Sci. (2014), http://dx.doi.org/10.1016/j.ins.2014.03.089