

On Using Page Cooccurrences for Computing Clickstream Similarity

Ravi Kothari *, Parul Mittal, Vivek Jain and Mukesh Mohania

Abstract

Clickstream analysis provides valuable insight into the behavior of users and can be translated into better business opportunities and increased user satisfaction. A fundamental problem in clickstream analysis is the computation of the distance (or the similarity) between two clickstreams. While, there exists a considerable amount of literature which propose methods of computing path similarities, they rely on the edit distance or the related longest common subsequence to align the two clickstreams. The edit distance provides a least cost sequence of transformations that result in the two clickstreams to be identical. Often, measures of path similarity are defined on these “aligned” clickstreams. However, the replacement cost used in the “alignment” process used by the edit distance is assumed to be fixed and ignores the degree of similarity of the two page views. Proposed in this paper is a method for computing the replacement cost that is based on the assumption that the degree of similarity between two page views is proportional to their relative frequency of cooccurrence. We define a method, which includes the order of the sequence as well as the time spent on each page, for obtaining the replacement cost of two arbitrary web pages. Though less accurate than content based analysis, our experiments with data generated from a simulator as well as data from an actual web site show that our assumption is well founded and that the proposed method provides a fast and accurate method of computing the similarity between two page views.

1 Introduction.

Broadly speaking, a clickstream may be defined as a sequence of *page views* accessed by the user. Though less informative than a *user-session* (page views across sites) [12], *server-session* based clickstreams provide considerable insight into the behavior of the user visiting a site and can be translated into better business opportunities and increased user satisfaction. For example,

- Clickstream data, by itself or in association with other information, can be used to cluster the users

into segments allowing for segment specific recommendations [7, 5, 8, 9, 1].

- Clickstream data allows the ranking of pages based on the frequency of access leading to improved site design [4]. In addition, access patterns can be used to create more effective advertisement campaigns that span across frequently visited paths.
- Correlation between subsequent page views (when page view attributes are known) allows one to determine if a user is randomly browsing the site or is looking for something in a focused manner. Recommendations in either case can be customized.

In the above mentioned applications and indeed in most applications involving clickstream analysis, a fundamental problem that arises is: *How to compute the similarity (or distance) between two clickstreams?* The popular approach to finding the distance between two clickstreams is based on using the *edit distance* [17, 16] or its variants [1]. The edit distance reflects the minimum amount of effort required in transforming two sequences such that they are identical. The transformation typically consists of three operations namely, insertion, deletion, and replacement with fixed and assumed insertion, deletion and replacement costs. Consider for example, a clickstream $Q_1 = \{a, b, c, d, e\}$ and another clickstream $Q_2 = \{a, c, f\}$. With a fixed replacement cost (i.e. the cost of replacing one symbol with a different symbol is the same irrespective of the two symbols), transformations of Q_2 such that $Q_2 = \{a, -, c, f, -\}$ or $Q_2 = \{a, -, c, -, f\}$ are equally admissible. However, an optimal solution would prefer the first solution if the distance between page views d and f is less than that between page views e and f . Thus to some extent, finding the similarity (or distance) between two clickstreams requires finding the distance between two page views.

Finding the *true* distance between two page views requires a semantic analysis of the page views. However, the present state of technology and the considerable computation time required in such an analysis precludes its use. It is possible to use a simple representation of a page view such as a vector space representation (each element of the vector reflects the presence or absence of a word) and use *TF-IDF* or *cosine* like

*The authors are with IBM - India Research Lab., Block I, IIT Hauz Khas, New Delhi 110016, INDIA. Corresponding author's email: rkothari@in.ibm.com

measures to characterize page view similarity [14, 13]. However, there are several drawbacks to such an approach. Besides ignoring the sequence of web pages, such approaches also require a considerable amount of preprocessing time (to form the *dictionary*; to remove *stop words* and *stemming*). In addition, a page view may contain information which is not necessarily textual in nature.

Computing the similarity of two clickstreams thus requires estimating the similarity between the page views in the clickstreams. In addition, a method of computing the similarity between two clickstreams must also include two additional characteristics. First, a clickstream is an ordered sequence of page views and any method for computing the similarity must take the order into account. Second, the amount of time spent on each page must be taken into account. While the amount of time spent on a page view derived from time stamp of access is subject to various sources of error (a user may simply get up, roam around, and come back to access the next page), an approach which removes extraordinarily large times may provide at least a qualitative indication of the amount spent by the user in viewing a page.

These difficulties notwithstanding, there is a large amount of literature pertaining to the analysis of clickstreams. Nonetheless, we believe, that there is considerable scope in developing heuristics that address the underlying problem — that of estimating the distance between two page views — in a computationally efficient manner.

In this paper, we propose a method for computing the similarity between two page views (and two clickstreams) that is based on the assumption that the joint probability of occurrence of two pages in user traversal paths is proportional to their similarity. The method we propose for estimating the joint probability takes the page view order as well the time spent on a page into account, is extremely fast to compute, and as we show later, provides good results.

We have organized the rest of the paper as follows. In Section 2, we provide a concise review of prevailing approaches related to computing clickstream similarity. Often the computation of clickstream similarity is not directly addressed as the focus of the paper may be something different. For example, a new clustering algorithm may simply assume an existent method of computing the distance between two clickstreams. Our focus in Section 2 is not on the higher level analysis (clustering, for example) but rather on the measure used for computing the distance between two clickstreams. In Section 3, we provide details of using the joint probability of occurrence of two page views to estimate

the distance between two page views. These distances are then used as the *replacement cost* when using the edit distance to align two sequences. We obtain the similarity (or distance) between two clickstreams based on the summation of pair wise distances (as obtained from cooccurrences) of aligned clickstreams.

Often, due to the lack of “ground truth” validation of the results becomes difficult. To circumvent this difficulty, we describe in Section 4 a simulator that we wrote to generate data consistent with specified user interests, page attributes, and site layout. Results obtained from clickstream analysis can then be compared with the results that would be obtained knowing the user interests and page attributes. We also present results obtained by using the proposed method on real-life data obtained from an actual eCommerce web site. In Section 5 we present our conclusions and highlight some possible extensions of our work.

2 A Concise Review.

In this section we present a concise review of approaches that have been proposed for computing clickstream similarity. We represent a clickstream of length n (sequence of n page views) as the ordered sequence of tuples $\mathcal{Q}_1 = \{(q_{11}, t_{11}), (q_{12}, t_{12}), \dots, (q_{1n}, t_{1n})\}$. Note that the first subscript represents the clickstream number and the second subscript represents the order of the page view. Thus, q_{12} represents the second page view of clickstream number 1. Similarly, t_{12} represents the time at which the second page view in clickstream number 1 is accessed. When the time spent on a page is ignored, as is done in some approaches, the second member of the tuple can be omitted.

Perhaps the most widely used method for computing the distance between two clickstreams is based on the so-called *Levenshtein distance* or, as it is more commonly called, the *edit distance* [17, 16]. The edit distance reflects the minimum amount of effort required in transforming two sequences such that they are identical. The transformation typically consists of three operations namely, insertion, deletion, and replacement with associated costs C_I , C_D , and C_R respectively. The edit distance $d(\mathcal{Q}_1, \mathcal{Q}_2)$ between two sequences, say \mathcal{Q}_1 and \mathcal{Q}_2 of lengths n and m respectively, can then be obtained from the following recurrence,

$$\begin{aligned}
 d_{0,0} &= 0 \\
 d_{i,0} &= d_{i-1,0} + C_D \\
 d_{0,j} &= d_{0,j-1} + C_I \\
 (2.1)_{i,j} &= \min \begin{cases} d_{i-1,j} + C_D \\ d_{i,j-1} + C_I \\ \begin{cases} 0 & \text{if } q_{1i} = q_{2j} \\ d_{i-1,j-1} + C_R & \text{if } q_{1i} \neq q_{2j} \end{cases} \end{cases}
 \end{aligned}$$

The recurrence of Equation (2.1) can continue until $i = n$ and $j = m$ yielding $d(Q_1, Q_2) = d_{nm}$ and naturally extends itself to a solution using *Dynamic Programming* [3].

When used for computing the distance between two clickstreams the essential step is to define the costs associated with the three basic operators, namely, C_I , C_D , and C_R . While C_I and C_D can be easily assumed to be equal to some value, defining C_R is problematic since C_R reflects the cost of replacement of a page view with another page view. As noted in the Introduction, an accurate estimate of C_R requires analyzing the page view contents of the two page views.

The approach that has been typically adopted is to use a fixed cost, for example, $C_I = C_D = 1, C_R = C_I + C_D$. For example, the *association measure* [8, 6] reduces to using a fixed cost. The *sequence alignment method* reported in [8] also falls in this category. From the perspective of clickstream analysis, it is clear that the distance between any two pages is not necessarily the same as the distance between any other pair of pages. Results obtained with using fixed costs are therefore less reliable. In [18], an algorithm based on the sequence alignment method is used to measure clickstream similarity and obtains clusters using ROCK, TURN and CHAMELEON clustering algorithms. The page similarity measure is based on the URL structure. It is claimed in [18] that the clusters formed based on sequence information are better than those formed without the use of sequence information. However, on any given web page there may be several links to unrelated pages and the URL structure is not necessarily indicative of the similarity. The actual sequence of page views visited by the user relies on the user to filter out these unrelated links.

A more elaborate scheme defines a large set of categories and finds the degree to which each category is represented on a page. At a finer level, one may use attributes though in either case the idea is to distribute across categories (attributes) the overall page content. Euclidean distance between the attribute vectors is then taken as C_R . Though this leads to greater accuracy, summarizing a page's attributes is laborious and subjective.

A technique closely related to the edit distance is the so-called *Longest Common Subsequence* (LCS) defined as the length of the largest subsequence common to both the sequences to be compared. Variations of LCS have also been proposed — for example, the weighted LCS which also takes into account the time spent on each page [1]. The similarity measure proposed has a “similarity component” which reflects how similar the two paths are in the region of their overlap and

an “importance component” which uses the geometric mean of the fraction of time occupied by the respective clickstreams in the region of overlap. The similarity is then defined as the product of the two components. However, the measure defined by the authors still uses a base LCS extractor [17] which uses the assumption that any two pages that are not identical are dissimilar to the same extent.

In [15] the notion of a *path feature space* is defined. The path feature space may be defined as the union of all possible sub-paths of a given path. The inner product in this path feature space is then used to compute the similarity between paths. Here again, sub-paths with no common pages are considered to be *perpendicular* which essentially reduces to the fact that any two pages that are not identical are dissimilar to the same extent.

As we noted in the Introduction, our purpose in this Section was not to present a review of literature pertaining to clickstream analysis and its use. Rather, we outlined some previous approaches to highlight that the more fundamental problem of computing the dissimilarity between non-identical pages persists and is often resolved in the current literature by simply treating non-identical pages as dissimilar. In the following Section we introduce a heuristic that provides a way of computing the distance between two non-identical pages without the use of content based analysis.

3 Using the Cooccurrences of Pages to Find C_R .

In the ideal case, the edit distance (or LCS) must use semantic analysis to compute the similarity (or distance) between two page views. However, the computational demands of such analysis or the present state of research does not make this a realistic proposition [11, 2]. A somewhat more plausible approach is to use document categorization techniques to create a universal set of categories and to associate with each page view a degree of membership in the universal set of categories. Euclidean distance between the membership vectors can then taken as C_R . However, even associating the contents of a page with various categories is a computationally expensive procedure. Nonetheless, our results with data generated from a simulator does compare the proposed approach with the attribute based approach¹.

To circumvent the difficulties associated with content and category based estimate of C_R , we propose a computationally efficient approach to computing C_R . Our approach is based on the hypothesis that the joint probability of occurrence of two page views is propor-

¹We point out that the attributes of a page reflect the contents at a finer granularity than the category.

tional to their degree of similarity.

Our assumption can be explained as follows. If users (on a statistical basis) visit a page (say a) and then visit another page (say b), there must clearly be a strong content based connection (and hence similarity) between the two pages. We first establish the mathematical framework for computing the joint probability of occurrence and later in the Section we provide the complete method for computing clickstream similarity.

We assume that there are a total of N pages in a site identified by q_1, q_2, \dots, q_N . In addition, let there be a total of M clickstreams and the i^{th} clickstream is represented as $\mathcal{Q}_i = \{(q_{i1}, t_{i1}), (q_{i2}, t_{i2}), \dots, (q_{in}, t_{in})\}$. Thus q_{i1} represents the first page view of clickstream i and corresponds to the page view q_1 . As before, t_{i1} represents the time at which the first page view in clickstream number i was accessed.

DEFINITION 3.1. A constraint vector $\theta = [\theta_g \theta_v \theta_e]$ is a 3-tuple comprising of $\theta_g \in \mathbf{Z}^+$, $\theta_v \in \mathbf{R}^+$, and $\theta_e \in \mathbf{R}^+$.

The constraint vector specifies three parameters and the role of each of them is as follows. θ_g is a positive integer (> 0) that is related to the maximum allowable “gap” between page views for them to be considered as cooccurring. Thus if θ_g is 2 and a sample clickstream (time ignored) is $\{a, b, c, d\}$ then a and c cooccur but a and d do not. A very large value of θ_g implies that the visitor does not change context in the server-session and that page views near the beginning and end of the server session are related by content. Such a large value of θ_g might be appropriate to use with clickstreams arising from very focused web sites (an on-line computer store for example). On the other hand a smaller value of θ_g is more appropriate when page views near the beginning and end of the server session are not related by content as might be the case with a news based web site.

θ_v is a positive (> 0) real number that is related to the “minimum” time that a page must be viewed for. A page viewed for a period shorter than θ_v implies that the user is just flipping through the pages and the sequence of pages viewed do not constitute feedback on content based relationship between the pages. Of course, θ_v depends on the amount of content on the page – for example, pages with images might require less viewing time than pages with more text.

θ_e is a positive (> 0) real number that is related to the maximum acceptable “elapsed” time between two page views. Pages that are separated by more than θ_e in time are not considered to cooccur. θ_e thus allows for the possibility that a user in interrupted or is multi-tasking in which case the connection between the page views might be suspect.

DEFINITION 3.2. Given a constraint vector, the joint

probability of occurrence of two pages is given by,

$$(3.2) \quad \Pr(q_l, q_m) = \frac{1}{A} \sum_{k=1}^M \sum_{l=1}^{|\mathcal{Q}_k|} \sum_{m=1}^{|\mathcal{Q}_k|} I((q_{kl}, t_{kl}), (q_{km}, t_{km})|\theta)$$

In Equation (3.2) $|\mathcal{Q}_k|$ denotes the length (number of page views) in a clickstream k and $I(\cdot)$ is an indicator function that is defined as,

$$(3.3) \quad I((q_{kl}, t_{kl}), (q_{km}, t_{km})|\theta) = \begin{cases} 1 & \text{if } \|l - m\| \leq \theta_g; \\ & \|t_{k,l+1} - t_{kl}\| > \theta_v; \\ & \|t_{k,m+1} - t_{km}\| > \theta_v; \\ & \|t_{km} - t_{kl}\| \leq \theta_e \\ 0 & \text{otherwise} \end{cases}$$

where, $\|\cdot\|$ stands for a suitable norm, A is the total number of pairs of pages admissible, and it is assumed that the difference between the time at which two successive pages are accessed corresponds to the time spent on viewing the first page.

It is possible to include the constraint of θ_g directly into the summation indexed by m in Equation (3.2) since there is no need to examine each pair wise page view but rather only those page views which are separated by an amount less than or equal to θ_g . However, we have presented it as in Equation (3.2) for clarity and simplicity and we rely on Equation (3.3) to allow only page views separated by less than or equal to θ_g to influence $\Pr(q_l, q_m)$.

An obvious property that follows from Equations (3.2) and (3.3) is that $0 \leq \Pr(q_i, q_j) \leq 1$. A high value of $\Pr(q_i, q_j)$ implies that the pages (q_i and q_j) were visited by many users in the same session.

DEFINITION 3.3. The distance between two pages q_i and q_j is given by,

$$(3.4) \quad d(q_i, q_j) = \begin{cases} \phi(\Pr(q_i, q_j)) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

where, $\phi(\cdot)$ is a monotonically decreasing function.

That is to say that as the joint probability of occurrence increases, the distance between the pages decreases. While there are many possible choices of ϕ , two simple ones that provide good results are: $\phi(x) = 1 - x$ and $\phi(x) = \exp(-x^2/(2\sigma^2))$ where σ controls the variance or in some sense the rate of decay of the function. With either choice of $\phi(x)$, $0 \leq d(q_i, q_j) \leq 1$.

3.1 Computational Considerations. Computationally, one can obtain the joint probability of occurrence of page views by initializing a matrix, say P , of size $N \times N$. An element of P , say $P(i, j)$

reflects $\Pr(q_i, q_j)$. One can simply go through each clickstream while incrementing the corresponding entry in P if the conditions of Equation (3.3) are satisfied. An appropriate normalization of P followed by the transformation specified by Equation (3.4) provides a matrix which contains the corresponding distances between any two page views. It may be noted that the cooccurrence matrix P may or may not be symmetric depending on the norm selected in Equation (3.3). In addition, the time aspect of clickstream analysis can be ignored if desired. At any rate, the framework is general and allows the use of ordering as well as time.

The time required to obtain the page view distance matrix is easily obtained by inspecting Equation (3.2). Including the constraint on the index of Equation (3.2) as discussed above reduces the actual computational time. However, since the order remains unchanged, we estimate the time required using Equation (3.2) directly. In essence, the number of times Equation (3.3) is used is given by,

$$(3.5) \quad \sum_{k=1}^M |\mathcal{Q}_k| * |\mathcal{Q}_k|$$

If the average length of a clickstream is given as $|Q|$, then the average time to compute the page view distance matrix is given by $O(M|Q|^2)$, i.e., it is linear in the number of clickstreams assuming that the number of clickstreams is much larger than the squared of the length of a clickstream. This is almost universally true as M can be in the hundreds of thousands and $|Q|$ is typically around 10 or a few tens.

Our overall scheme for obtaining the distance between two page views can thus be defined. We solve the Dynamic Programming problem of Equation (2.1) using the entries of P transformed through Equation (3.4) as the corresponding replacement cost. Note that the replacement cost of each pair of pages is different depending of course on their similarity. Through a traceback phase [3], we obtain the aligned clickstreams. Since the edit distance allows for insertions and deletions, it results in two clickstreams that look like, for example, $\mathcal{Q}_1 = \{a, b, c, d\}$ and $\mathcal{Q}_2 = \{a, -, c, e\}$. There are two possibilities. If the page access times are not used then we replace the insertions with either the preceding or the following page view depending on which provides the lower cost. Thus, $\mathcal{Q}_2 = \{a, a, c, e\}$ or $\mathcal{Q}_2 = \{a, c, c, e\}$ depending on whether the distance between b and a is lower or whether the distance between b and c is lower. This corresponds to letting a user spend an increased amount of time on a particular page. If the page access times are used, then a fixed cost (either insertion or deletion) results between a page view and a _.

In the following Section, we present the results obtained with synthetic data generated from a simulator as well as from data obtained from an actual eCommerce web site.

4 Results.

A particularly difficult aspect of research involving clickstreams, and in fact many other aspects of web mining, is the lack of *ground truth* that can be used to provide a definitive sense of error. To resolve this, we created a simple though realistic simulator. The ground truth being known, it allows for a definitive evaluation of the performance of our proposed algorithm. Later in this section, we also present results with data from an actual web site.

4.1 Synthetically Generated Data. To generate the synthetic data, we wrote a simulator which generates a web site with specified connectivity (which page is directly accessible from which page) between the pages. A “universal” set of attributes is defined and each page in the site is characterized by the degree to which each of the attributes are present on the page. That is to say that the sum of attributes that characterize a page is 1. A set of users, each with interests defined in terms of the “universal” set of attributes are defined. As in the case of page attributes, the sum of a users interests across various attributes is 1.

Each artificial user is then made to navigate the site while being mindful of the page connectivity constraints; at each step the user follows a link that most matches the users defined interests. A degree of randomness (ϵ) is also used such that on occasion a user may navigate to a page which is not necessarily the page whose attributes best match his own interests. The algorithm underlying the simulator is illustrated in Algorithm 1.

The advantage of writing this simulator and using the data generated from it for a part of our experimentation is that it allows for the validation of results based on actual page attributes or user interests. Note that in a real setting, characterizing each page in terms of its attributes is a laborious process, and even when done, is subjective. Since the data is generated based on known values of page attributes and users interests, it is possible to compare the cooccurrence based results with that obtained from using the actual page attributes.

For our simulation we generated $M = 20,000$ clickstreams (one corresponding to a user) on a web site which has $N = 100$ pages. We considered a set of 10 universal attributes.

```

Data : N (number of pages), NAttributes (number of attributes), NUsers (number of users)
Result : M Clickstreams — one for each user (= NUsers)

/* Set and normalize the attributes of each page. PageAttributes is N × NAttributes */
/* Treat Page 1 as the home page. SetAttribute (below) can return a vector with one or a group of
attributes high and the rest set to small random values */
for  $i = 1$  to  $N$  do
  PageAttributes( $i$ ) ← SetAttribute(1,NAttributes)
  PageAttributes( $i$ ) ← PageAttributes( $i$ ) / norm(PageAttributes( $i$ ))
end

/* Set the connectivity between the pages. PageConnectivity is N × N. PageConnectivity( $i,j$ ) is 1 if page
 $j$  is connected to page  $i$  */
for  $i = 1$  to  $N$  do
  for  $j = 1$  to  $N$  do
    if ( $\|$  PageAttributes( $i$ ) - PageAttributes( $j$ )  $\|$  < Threshold) then
      PageConnectivity( $i,j$ ) ← 1
    else
      PageConnectivity( $i,j$ ) ← 0
    end
  end
  /* All pages connect to the home page */
  PageConnectivity( $i,1$ ) = 1;
end

/* Set and normalize each users interest. UserInterests is NUsers × NAttributes */
for  $i = 1$  to NUsers do
  UserInterests( $i$ ) ← SetAttribute(1,NAttributes)
  UserInterests( $i$ ) ← UserInterests( $i$ ) / norm(UserInterests( $i$ ))
end

/* Let each user browse starting from a random page */
 $\epsilon$  ← random
for  $i = 1$  to NUsers do
  CurrentPage ← random
  ClickstreamLength ← random
  for  $j = 1$  to ClickstreamLength do
    A ← find(PageConnectivity(CurrentPage) == 1)
    NextPage ← Page in A which best matches (UserInterests( $i$ )+ $\epsilon$ )
    Append CurrentPage to  $\mathcal{Q}_i$ 
    CurrentPage ← NextPage
  end
end

```

Algorithm 1: The main function of the simulator package

We assigned the page attributes such that all the pages clustered into 10 clusters, i.e., for each page we set one attribute to be dominant (0.5) and the remaining 9 attributes randomly (combined membership in the remaining 9 attributes to 0.5). Page connectivity was established so that a subset of pages with similar attributes were linked. Page number 1 was treated as the “home page” and all pages had connectivity to the home page.

To model the user interests, we followed a similar strategy. Each of the M users had preference for a dominant attribute (0.5) but also had lesser random preferences in the remaining attributes (0.5). A user navigates to the next page based on best match with his or her own interests (attributes). To make it interesting and realistic we included some randomness — once in a while the next page is not necessarily the one that best matches the users interests. ϵ controls this randomness and a high value reflects a path which has little fidelity to a users interests while a low value of ϵ reflects a path which has complete fidelity to the users interest.

We ran two simulations with $\epsilon = 0$ (no randomness - the next page visited is the one that best matches the users interests) and with $\epsilon = 0.1$ (10% randomness - user interests are perturbed by 10%). Figure 1 shows a pictorial view of the page view distance matrices (distance from a page view to all other page views) using the fixed replacement cost (all entries are the same, diagonal entries are 0), Euclidean distance between the page view attributes, and using the proposed cooccurrence based page view distance computation. In Figure 1, the 10 clusters are visible near the diagonal (recall that the attributes of the page views were set such that all the page views clustered into 10 clusters). Pages within a cluster are closer to each other and hence appear darker (less distance) in the figure.

The Euclidean distance between page view attributes corresponds to the ideal situation — one that is impractical in a real setting due to the resources required for such categorization. The attribute page view distances corresponds to the *ground truth* and the advantage of cooccurrence derived replacement cost over a fixed replacement cost can be obtained by observing their agreement with the *ground truth*. It is clear from Figure 1 that the distance between page views as obtained from the proposed cooccurrence based method agrees strongly with the true page view distances. Figure 2 makes this agreement more explicit by showing the distribution of the error between the fixed and ground truth replacement cost as well as the distribution of the error between the cooccurrence derived and ground truth replacement cost. We only show this for the case when $\epsilon = 0$ — however the plots are similar for $\epsilon = 0.1$.

Note that the errors of the cooccurrence derived replacement costs are centered around 0 (mean of the error is 0.0899) whereas the errors with the fixed replacement are displaced (mean of the error is 1.3638). Both the pictorial representation of Figure 1 as well as the quantification in Figure 2 show the poor approximation of the true page view distance matrix when fixed replacement cost is used. In contrast, page view distances as obtained with the cooccurrence derived replacement cost represent a high fidelity approximation of the true page view distances.

We also ran additional simulations with the synthetic data to compare the relative performance of fixed replacement cost and the proposed cooccurrence based replacement cost. Here for a random subset of the clickstreams, we find the clickstream that is nearest to a clickstream based on using (i) fixed replacement cost, and (ii) joint probability of occurrence as the replacement cost. Once a clickstream nearest to a specified clickstream is found, we find the distance between these clickstreams using the sum of the pair wise page distances (computed as the Euclidean distance between the page attributes). For comparison, we also computed the nearest clickstream using the Euclidean distance between the page views as the replacement cost. The cost of insertion and deletion (C_I and C_D) are fixed and the same in each case. Figure 3 shows the distances obtained with cooccurrence based replacement cost (vertical axis) against page attribute based replacement cost (horizontal axis). Note that the horizontal axis (page attribute based replacement cost) corresponds to the *ground truth* and agreement between the proposed cooccurrence based replacement cost and the page attribute based replacement cost corresponds to a diagonal line in the panel. For comparison, we also plot the distances obtained with a fixed replacement cost (vertical axis) against the page attribute based replacement cost (horizontal axis).

The Figure shows two panels — the left being for smaller clickstream distances (< 7) and the right one being for larger clickstream distances (≥ 7). Our rationale for showing two sub-plots is as follows. When very dissimilar page views are part of the clickstream (corresponding to a larger overall distance between the clickstreams), cooccurrence prescribes a high replacement cost which agrees with the fixed replacement cost. Thus results obtained by the two methods are very similar. It is only when the page view distances are small (corresponding to a smaller overall distance between the clickstreams), that cooccurrence prescribes a small replacement cost which (correctly) disagrees with the fixed replacement cost. Thus the results are shown separately in two sub-plots.

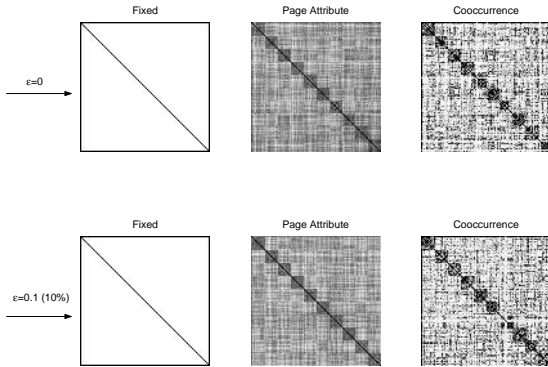


Figure 1: The page view distance matrix (distance of a page view from all other page views) as obtained from fixed replacement cost (left panel), from the Euclidean distance between the true page attributes (center panel), and using the proposed cooccurrence based method (right panel). The top row corresponds to $\epsilon = 0$ and the bottom row corresponds to $\epsilon = 0.1$. In each case, the proposed cooccurrence based method provides a high fidelity approximation of the actual distance between the page views. Darker shade signify smaller distance.

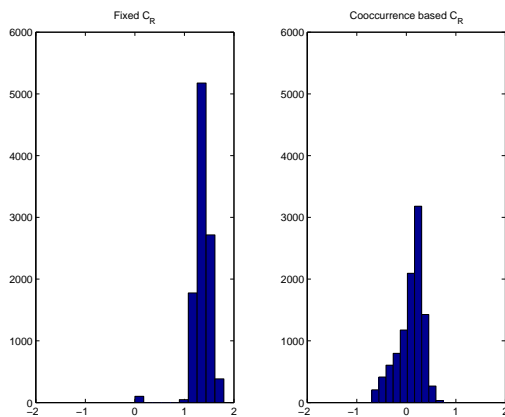


Figure 2: Distribution of the errors when $\epsilon = 0$. The left panel shows the distribution of the errors with a fixed replacement cost while the right panel shows the distribution of the errors with the cooccurrence derived replacement cost. Similar plots are obtained when $\epsilon = 0.1$.

Table 1 summarizes the mean (μ) and standard deviation (σ) of the error for smaller and larger clickstream distances when cooccurrence based and fixed replacement costs are used. Clearly, when the clickstream distances are small, the proposed cooccurrence based replacement costs do better than the fixed replacement cost.

The difference in results between cooccurrence and fixed replacement cost depends on the distribution of errors i.e. the difference between fixed (or cooccurrence based) replacement costs and the true page attribute based replacement cost. While, there were simulations where this difference was much more than that shown in Table 1, we chose a conservative difference to report here. A mathematical characterization of the anticipated difference is presently underway and we believe that the more favorable results would be more suitable to report when accompanied by the mathematical characterization.

4.2 Real Data. We also ran the proposed algorithm on a single day of web log data captured from an actual eCommerce web site. User sessions were extracted from web logs assuming that a single IP address refers to a single user.

Initially the data comprised of 9,454 unique URLs. We grouped large numbers of URLs together — for example, all product display pages were treated as a single URL, all add to shop cart events were treated as single URL (irrespective of the page from which the command was invoked and the parameters used). This reduced the total number of URLs to 31. Table 2 shows the contents of the 31 page views (URLs).

The initial web log had a total of 129,935 user sessions. Sessions with less than 5 and more than 80 clicks (such long session typically originate from shopbot or crawler actions) were removed resulting in 20,584 sessions. The distance between page views was obtained as described by Equations (3.2) through (3.4) with $\theta = [\infty \ 0 \ \infty]$. Our choice of θ was mitigated by the focused nature of this particular site.

We then chose a random sub-sample of 800 clickstreams to see if the clickstreams sensibly cluster using the proposed similarity measure. The distance between these clickstreams was computed as proposed in Section 3, and a hierarchical clustering algorithm [10] was used to cluster the clickstreams. Once the clustering was done, we examined the frequency of pages accessed within each of the six clusters.

| C_R Method | Small clickstream dist. | | Large clickstream dist. | |
|--------------|-------------------------|----------|-------------------------|----------|
| | μ | σ | μ | σ |
| Fixed | 0.1116 | 0.1403 | 0.0575 | 0.0472 |
| Cooccurrence | 0.0837 | 0.1008 | 0.0639 | 0.0472 |

Table 1: The mean and standard deviation of the deviation of the error between the distance between clickstream obtained from cooccurrence derived and fixed replacement cost.

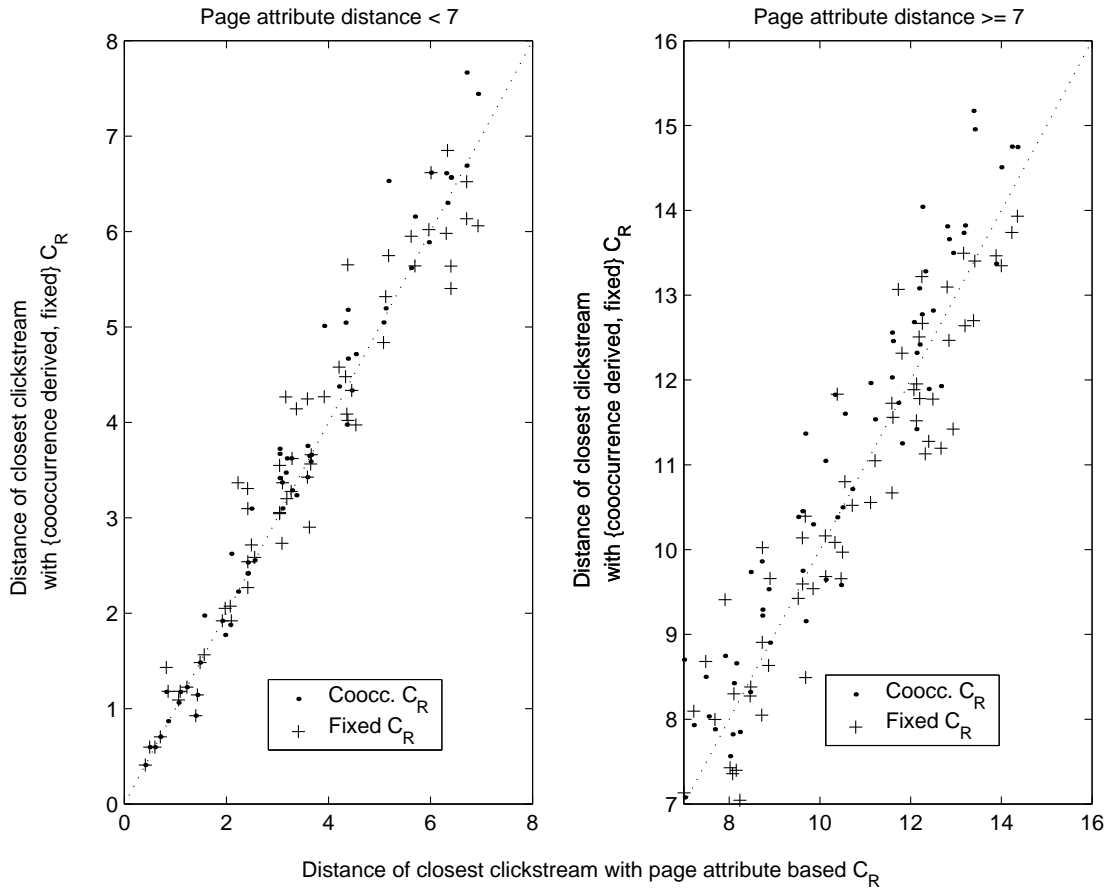


Figure 3: Comparison of the performance obtained with using the joint probability of occurrence as opposed to a fixed replacement cost. See text for details.

| # | Content | # | Content |
|----|------------------|----|-----------------|
| 1 | Accessories | 17 | SendBCFD |
| 2 | AddShopCart1 | 18 | Register |
| 3 | AddReplaceConfig | 19 | CopySaveCart |
| 4 | CategoryDisplay1 | 20 | DeleteSavedCart |
| 5 | Affiliates | 21 | Checkout |
| 6 | Auctions | 22 | ShopCart |
| 7 | AddShopCart2 | 23 | SavedCart |
| 8 | CategoryDisplay2 | 24 | Promotion1 |
| 9 | Footnote | 25 | Logon |
| 10 | SeasonGift | 26 | OrderItemDelete |
| 11 | HomePage | 27 | OrderCancel |
| 12 | ProductDisplay1 | 28 | ShipTo |
| 13 | OneClickPurchase | 29 | Product1 |
| 14 | SpecialOffers | 30 | Promotion2 |
| 15 | Upgrades | 31 | Others |
| 16 | CheckBCFD | | |

Table 2: The page views and the corresponding page content. # corresponds to the page numbers.

The results are shown in Figure 4. Each panel represents the frequency of pages accessed by members in the cluster. Several interesting observations can be made. First, each of the clusters show distinguishing page accesses. Further, when the content of page views are examined (see Table 2), physical meaning can be attached to the clusters.

Cluster 1 (sub-panel (a) in Figure 4) is a set of users who have browsed more category display pages and less product pages. This corresponds to individuals who are casually browsing the site. Cluster 2 (sub-panel (b) in Figure 4) is a set of users who have browsed more product pages and promotion pages relative to cluster 1 users. This corresponds to users who are more focused in their browsing behavior and perhaps getting the information in preparation for a purchase. Cluster 3 (sub-panel (c) in Figure 4) is a set of users who arrived at the site from advertisements at third party affiliate sites and have left immediately. Cluster 4 (sub-panel (d) in Figure 4) is a set of users who have browsed special offer pages, auction pages, promotion pages, added product to shop carts and left without buying (no checkout). This corresponds to users who did not have a specific need but were hunting for a bargain. Cluster 5 (sub-panel (e) in Figure 4) and cluster 6 (sub-panel (f) in Figure 4) is a set of users who have made purchases. The difference is that cluster 5 users are more certain of what they want to buy (less browsing of category, product display and promotion pages) compared to cluster 6 users.

The results demonstrate two things. First, that the proposed measure of computing page view distance is effective since it allowed the discovery of physically meaningful clusters. Second, clickstream analysis can be used to distinguish various categories of shoppers (for example, bargain hunters from more focused buyers etc.).

5 Conclusions.

In this paper, we proposed a measure which uses the joint probability of occurrence of pages accessed by users of a web site to approximate the distance between two page views and an extension for computing the distance between two clickstreams. The estimated page view distance can be used to provide realistic replacement cost estimates when using the edit distance and its extensions to compute the distance between two clickstreams can be used in applications such as clustering. Experiments with synthetic and real data indicate that the proposed cooccurrence derived measure provides results that are close to the results that would be obtained had the actual attributes of pages were known.

The overall problem of mining clickstream data has considerable practical benefits. Experience with eCommerce sites has shown that most registered users do not login until they are ready to checkout or need to access customizations provided by the site. However, until such time that a registered user logs in, and for unregistered users, clickstream remains the only source of information that can be utilized for making personalized marketing recommendations. For example, if a set of prototypical clickstreams along with their associated profiles is maintained, then by finding the prototypical clickstream closest to the clickstream of a user, it is possible to use the profile associated with the closest prototypical clickstream as an approximation of the profile of the unknown user. Personalization is thus made possible.

In addition, correlation between the attributes of subsequent page views (when page view attributes are known) allows one to determine if a user is randomly browsing the site or is more focused. Recommendations in either case can be customized.

In these and other situations, finding clickstream similarity is required and we anticipate the proposed method to be of significant value. Of course, there is the possibility that if all visitors to a site are random browsers, the joint probability of occurrence of the pages will be equal. However, under the hypothesis that a larger proportion of the visitors follow related links, we believe that the proposed method provides more accurate results compared to using a fixed replacement cost.

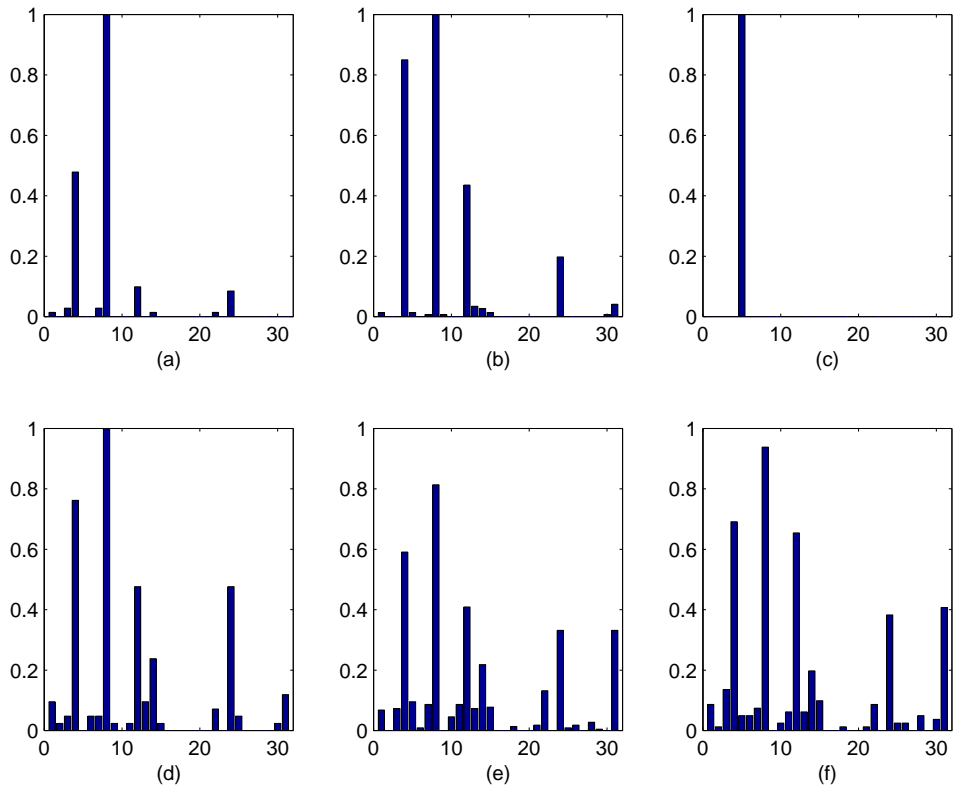


Figure 4: Pages (horizontal axis) and the frequency of access (vertical axis) of the individual pages by users in each of the six clusters.

Acknowledgements

Helpful discussions with Raghavendra Singh are gratefully acknowledged.

References

- [1] A. Banerjee and J. Ghosh, *Clickstream clustering using weighted longest common subsequences*, in Proceedings of the Web Mining Workshop at the 1st SIAM Conference on Data Mining, Chicago, USA, 2001.
- [2] S. Baumann, M. H. Malburg, H. Hein, R. Hoch, T. Kieninger, and N. Kuhn, *Document analysis at DFKI part 2: Information extraction*, Tech. Report RR-95-03, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH, 1995.
- [3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Athena Scientific, 2001.
- [4] K.-P. H. C. Theusinger, *Analyzing the footsteps of your customers: A case study by ASK—net and SAS Institute GmbH*, in WebKDD-2000, 2000.
- [5] R. W. Cooley, *Web Usage Mining: Discovery and Application of Interesting Patterns from Web Data*, ph.d. dissertation, University of Minnesota, Minnesota, USA, 2000.
- [6] B. Everitt, *Cluster Analysis*, Halsted Press, New York, USA, 1980.
- [7] Y. Fu, K. Sandhu, and M.-Y. Shih, *Fast clustering of web users based on navigation patterns*, in Proc. World Multiconference on Systemics, Cybernetics and Informatics (SCI/ISAS'99), Orlando, USA, 1999, pp. 560–567.
- [8] B. Hay, G. Wets, and K. Vanhoof, *Clustering navigation patterns on a website using a sequence alignment method*, in Proc. IJCAI's Workshop on Intelligent Techniques for Web Personalisation, 2001.
- [9] H. Hirsh, C. Basu, and B. Davison, *Learning to personalize*, Communications of the ACM, 43 (2000), pp. 102–106.
- [10] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, USA, 1988.
- [11] R. Kosala and H. Blockeel, *Web mining research: A survey*, SIGKDD Explorations, 2 (2000), pp. 1–15.
- [12] B. Padmanabhan, Z. Zheng, and S. O. Kimbrough, *Personalization from incomplete data: what you don't know can hurt*, in Knowledge Discovery and Data Mining, 2001, pp. 154–163.
- [13] M. B. S. Robertson, S. Walker, *Okapi at TREC-7: Automatic ad-hoc, filtering, vlc and interactive*, in Proceedings of the 7th Text Retrieval Conference (TREC-7), 1998.
- [14] G. Salton, *Automatic Text Processing – The Transformation, Analysis and Retrieval of Text by Computer*, Addison-Wesley, Reading, MA, 1989.
- [15] C. Shahabi, A. Zarkesh, J. Adibi, and V. Shah, *Knowledge discovery from users web-page navigation*, in Proceedings of the 7th International Conference on Research Issues in Data Engineering, 1997, pp. 20–29.
- [16] E. Ukkonen, *On approximate string matching*, in Proceedings International Conference on Foundations of Computational Theory (LNCS 158), Springer-Verlag, 1983, pp. 487–495.
- [17] R. A. Wagner and M. J. Fischer, *The string-to-string correction problem*, Journal of ACM, 21 (1974), pp. 168–173.
- [18] W. Wang and O. R. Zaiane, *Clustering web sessions by sequence alignment*, in Proceedings DEXA, 2002.