Clustering Web Sessions by Sequence Alignment

Weinan Wang Osmar R. Zaïane

Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
{weinan, zaiane}@cs.ualberta.ca

Abstract

Clustering means grouping similar objects into groups such that objects within a same group bear similarity to each other while objects in different groups are dissimilar to each other. As an important component of data mining, much research on clustering has been conducted in different disciplines. In the context of web mining, clustering could be used to cluster similar click-streams to determine learning behaviours in the case of e-learning, or general site access behaviours in e-commerce or other on-line applications. Most of the algorithms presented in the literature to deal with clustering web sessions treat sessions as sets of visited pages within a time period and don't consider the sequence of the click-strem visitation. This has a significant consequence when comparing similarities between web sessions. We propose in this paper a new algorithm based on sequence alignment to measure similarities between web sessions where sessions are chronologically ordered sequences of page accesses.

1 Introduction

Clustering web sessions is the problem of grouping web sessions based on similarity and consists of maximizing the intra-group similarity while minimizing the inter-group similarity. The problem of clustering data sets in general is also known as unsupervised classification, since no class labels are given. The problem of clustering web sessions is part of a larger work of web usage mining which is the application of data mining techniques to discover usage patterns from Web data typically collected by web servers in large logs [18]. Data mining from web access logs is a process consisting of three consecutive steps: data gathering and pre-processing for filtering and formatting

the log entries, pattern discovery which consists of the use of a variety of algorithms such as association rule mining, sequential pattern analysis, clustering and classification on the transformed data in order to discover relevant and potentially useful patterns, and finally, pattern analysis during which the user retrieves and interprets the patterns discovered [21].

Session cluster discovery is an important part of web data mining. In the context of e-learning, our application of interest, the function of clustering can have a myriad uses, such as grouping learners with similar on-line access behaviour, grouping pages with similar access or usage, or grouping similar web sessions to determine different learning behaviours in a given on-line course. Most of these groupings are concerned with categorical data. Learners, pages or sessions are indeed represented by vectors, either feature vectors for learners and pages, or sequences in the case of sessions. Unfortunately, most current clustering algorithms cluster numerical data. Very few are particularly suitable for clustering categorical attributes.

In our study, we are interested in clustering sessions in order to identify significant or dominant learning behaviours in online courses. The ulimate goal is to provide educators with a tool to evaluate not only on-line learners, but also evaluate the course material structure and its effective usage by the learners. In order to cluster sessions, after identifying the sessions in a pre-processing phase, we used clustering algorithms known for their ability to handle categorical data: ROCK [5] an algorithm that acts on a sample of the dataset, CHAMELEON [9], which is based on graph partitioning, and a new algorithm TURN for discrete distributions that we introduced in [3]. All of these algorithms, when used in the past for clustering web sessions, have treated sessions as unordered sets of clicks. The similarity measures used to compare sessions were simply based on intersections between these sets, such as the cosine measure or the Jaccard coefficient. This was also the case for our work in [3] where we also applied the Jaccard coefficient which basically measures the degree of common visited pages in both sessions to be compared. While it is the common practice, it is not an adequate measure since the sequence of events is not taken into account. If page A is visited just before page B, it is different from the statement acknowledging that pages A and B were visited in the same session, disregarding the possible pages visited in between.

In this paper, we introduce a new method for measuring similarities between web sessions that takes into account the sequence of event in a clickstream visitation. This measure also considers similarities between pages visited in a session. This method can be used to cluster web sessions using any clustering algorithm that allows the usage of an arbitrary similarity measure as a distance function for grouping similar data objects. Our preliminary experiments using the algorithms ROCK, CHAMELEON and TURN show

that the clusters discovered are more meaningful than those discoved when considering sessions as simple sets of visited pages.

The remainder of the paper is organized as follows: Section 2 gives a general overview of pre-processing the web logs and presents the data input to our clustering problem. Section 3 presents some clustering algorithms recently proposed for clustering web sessions, and underlines some shortcomings of these algorithms vis-à-vis the notion of sequence of accesses as web sessions, which has a significant impact on the effectiveness of the clustering outcome. Section 4 describes our similarity measures for comparing pages as well as sequences of pages accesses. We discuss some preliminary experiments in Section 5 using different clustering algorithms for categorical data. Finally, Section 6 concludes our study and gives general directions for future work.

2 Pre-processing the data

For a given raw web-log, algorithms have been developed to clean the web log, identify users, user sessions, and create IDs for web pages. The cleaning involves the removal of entries in the web log which contain an error flag, requests for images and other embedded files, applets and other script codes, requests generated by web agents whose function is to pre-fetch pages for caching, requests from proxies, and requests reset by visitors, etc.

It was demonstrated that in the general context of e-commerce sites, sessions could be accurately determined by identifying idle times between page accesses [13]. While we do not advocate this sessioninzing practice for web sessions in the context of e-learning [20], we have adopted this method in this paper for the sake of simplicity and proof of concept. In this study, user sessions have been identified using a 25-minute timeout threshold between page access. Web page IDs are constructed based on assigning an ID to each component of the URL so two web page IDs can be compared for similarity or "closeness". Also, in our particular application, learners are uniquely identified in the web log and the usual user identification problem in web mining did not apply here. After cleaning the raw web log data, two data files are provided which contain the cleaned data: on one hand the session and on the other the accessed pages. The format of the session file is as the following example:

01000102 962 945458058
01000102 962 945458060
01000102 483 945458060
01000102 484 945458060
01100001 965 937344265
01100001 963 937340669
01100001 964 937340670

01100001 964 937341439

Each column of this file represents: **session number**, **page number**, and **time stamp** respectively. The accessed page file has the following format:
07 : 0060 : /Courses/TECH142/TeachingStaff/index.html
08 : 007 : /Courses/TECH142/index.html
09 : 008 : /Courses/TECH142/side.html
10 : 01 : /Courses/TECH150
11 : 0100 : /Courses/TECH150/CourseDescription/index.html
12 : 0110 : /Courses/TECH150/Evaluation/index.html

Each column of this file represents: **page number**, **page ID** and **URL** respectively. **page ID** is a unique string used to represent a web page; each letter in this string represents one level of URL path.

Our problem of web session clustering is based on these two session and page files. The output is a file containing a list of session numbers with their respective cluster label. In these above example files, we do not have information about the users. However, the learners ID could be attached to the session numbers. The results of clustering can give insight in the user's behaviour in a web site and have significant applications in personalization, recommendation system, adaptive sites, etc.

3 Related work on clustering web sessions

Most of the studies in the area of web usage mining are very new, and the topic of clustering web sessions has recently become popular in the field of real application of clustering techniques. Shahabi et al. [15] introduced the idea of Path Feature Space to represent all the navigation paths. Similarity between each two paths in the Path Feature Space is measured by the definition of Path Angle which is actually based on the Cosine similarity between two vectors. In this work, k-means cluster method is utilized to cluster user navigation patterns. Fu et al. [4] cluster users based on clustering web sessions. Their work employed attribute oriented induction to transfer the web session data into a space of generalized sessions, then apply the clustering algorithm BIRCH [22] to this generalized session space. Their method scaled well over increasing large data. However, problems of BIRCH include that it needs the setting of a similarity threshold and it is sensitive to the order of data input. The paper does not discuss in detail how they measure the closeness between sessions and how they set the similarity threshold which are very important for clustering. Mobasher et al. [12] used clustering on a web log using the Cosine coefficient and a threshold of 0.5. No detail is mentioned of the actual clustering algorithm used as the paper is principally on Association Rule mining. One recent paper which bears some

similarity to our work is by Banerjee and Ghosh [1]. This paper introduced a new method for measuring similarity between web sessions: The longest common sub-sequences between two sessions is first found through dynamic programming, then the similarity between two sessions is defined through their relative time spent on the longest common sub-sequences. Applying this similarity definition, the authors built an abstract similarity graph for the set of sessions to be clustered, then the graph partition method was applied to "cut" the abstract graph into clusters. Our method has a similar basic idea on measuring session similarity, but we consider each session as a sequence and borrow the idea of sequence alignment in bioinformatics to measure similarity between sequences of page accesses. However, we look into more detail of each web page by first defining a similarity between each two pages, then instead of simply finding the longest common sub-sequence, our method utilizes dynamic programming to find the "*Best Matching*" between two session sequences. In our method, similarity between sessions are measured through their *Best Matching*.

Other works indirectly related to the topic of web session clustering include: Pitkow et al. [8] explored predictive modeling techniques by introducing a statistic Longest Repeating Sub-sequence model which can be used for modeling and predicting user surfing paths. Spiliopoulou et al. [14] built a mining system, WUM, for the discovery of interesting navigation patterns. In their system, interestingness criteria for navigation patterns are dynamically specified by the human expert using WUM's mining language MINT. Mannila and Meek [6] presented a method for finding partial orders that describe the ordering relationships between the events in a collection of sequences. Their method can be applied to the discovery of partial orders in the data set of session sequences.

4 Similarity Measures for Web Sessions

The first and foremost question needed to be answered in clustering web sessions is how to measure the similarity between two web sessions. A web session is naturally a stream of hyper link clicks. Most of the previous related works apply either Euclidean distance for vector or set similarity measures, Cosine or Jaccard Coefficient. Shortcomings for doing this is obvious: (1) the transferred space could be of very high dimension; (2) The original click stream is naturally a click sequence which cannot be fully represented by a vector or a set of URLs where the order of clicks is not considered; (3) Euclidean distance has been proven in practice not suitable for measuring similarity in categorical vector space.

Here we propose to consider the original session data as a set of sequences, and apply sequence similarity measure to measure similarity between sessions. Sequence alignment actually is not a new topic; there exist several

algorithms for solving sequence alignment problems [10]. Our method for measuring similarity between session sequences borrows the basic ideas from these algorithms. However, most sequence alignment algorithms for DNA sequencing consider very long sequences consisting of a limited vocabulary (i.e. "C", "A", "T" and "G"). In our case, the sequences are relatively short (hundreds of clicks at most per session) but the vocabulary is very large (in the order of thousands of different pages).

There exist two steps in our definition of session similarity. First we need to define similarity between two web pages because each session includes several web pages; the second step is to define session similarity using page similarity as an inner function.

4.1 Similarity Between Web Pages

If we do not consider the content of pages but simply the paths leading to a web page (or script), we notice that there exist similarities between many different web pages. One example is like the following two URLs:
URL#1:
http://www.cs.ualberta.ca/labs/database/current.html
URL#2:
http://www.cs.ualberta.ca/labs/database/publications.html
Similarity between these two URLs is obvious: They are very similar pages with a similar "topic" about the research work in the Database group of the University of Alberta.

In another example, the similarity between the two URLs is not that obvious, but the faint similarity definitely exists:
URL#1:
http://www.cs.ualberta.ca/labs/database/current.html URL#3:
http://www.cs.ualberta.ca/theses/
URL#1 is about the current research work in the database lab of the Department of Computing Science at the University of Alberta; URL#3 is about the theses finished in the recent years in the Department of Computing Science with the University of Alberta. Here the similarity is simply the fact that both pages come from the same server. We feel that there is some similarity between URL#1 and URL#3, but the similarity is of course not as strong as the similarity between URL#1 and URL#2 in the previous example. We need a systematic method to give a numerical measure for the similarity between two URLs.

In order to measure the similarity between two web pages, we first represent each level of a URL by a token; the token string of the full path of a URL is thus the concatenation of all the representative tokens of each level. This process corresponds to marking the tree structure of a web site as shown in Figure 1. Notice here that we assume that the URL path can fully reflect the content of URL, also we assume that the URL connection is tree

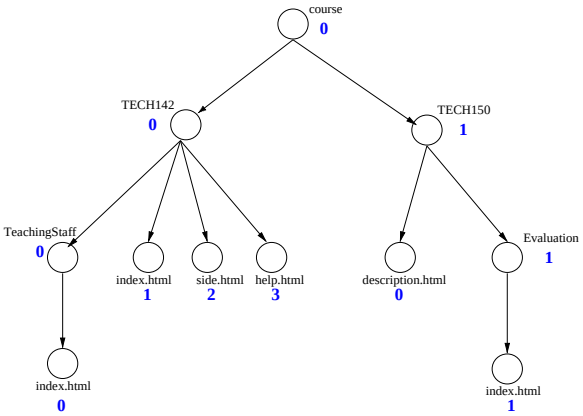

Figure 1: Labeling a tree structure of a web site

structured and there are no loops or cycles. This is often true for on-line learning environments such as the e-learning system for which we analyzed the web logs, an on-line course delivery system at the Technical University of British Columbia (TechBC), a Canadian university that delivers most its courses on-line.

The web page "/course/TECH142/index.html" in Figure 1, is represented by the token string "**001** ", the webpage "/course/TECH150/description.html" is represented by the token string "**010**". The computation of web page similarity is based on comparing the token string of web pages.

Our web similarity computation works in two steps:

- **Step1:** We compare each corresponding token of the two token strings one by one from the beginning, and this process stops at the first pair of tokens which are different. For example, let us compare the web pages "/course/TECH142/TeachingStuff/index.html" and "/course/TECH142/side.html". The token string of the first web page is "**0000**", and the second web page's token string is "**001**". Now compare the two token strings in Figure 2.

  From Figure 2 we see that the two token strings have two same corresponding tokens. Notice that if we compare the token string "**0111**" and "**0101**", they have only two same corresponding tokens because the comparing process stops at the first pair of different tokens.

- **Step2:** compute the similarity of two web pages. The similarity between two web pages are computed in this way: suppose the length of the first token string is $length1$, and the length of the second token string is $length2$, select the the longer string's length $longer\_length = (length1 > length2)\ ?\ length1 : length2$, then we give weight to each

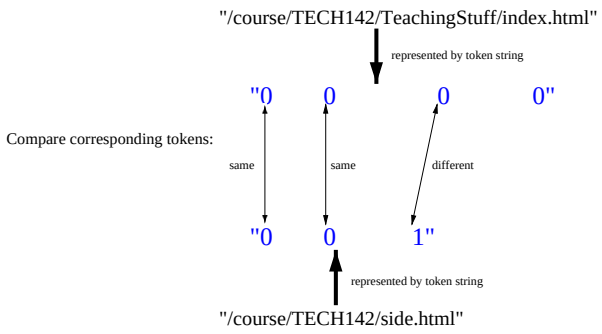

Figure 2: Compare token strings

Weight of each token:	4	3	2	1
token string1:	0	0	0	0
token string2:	0	0	1	

Figure 3: Weight each token level

level of the longer token: the last level is given weight 1, the second to the last level is given weight 2, the third to the last level is given weight 3, and so on and so forth, until the first level which is given weight $longer\_length$. For the given example, the length of token string "**0000** " is 4, and length of the token string "**001** " is 3, thus $longer\_length = 4$, and weight for each level is shown in Figure 3

Next, the similarity between two token strings is defined as the sum of the weight of those matching tokens divided by the sum of the total weights. For the given example, we see that their similarity is thus:

$$(4 + 3)/(4 + 3 + 2 + 1) = 0.7$$

For any pair of URLs, using this similarity measure, the two pages' similarity is between 0.0 and 1.0. If the two pages are totally different, i.e. no same corresponding token, their similarity is 0.0. If the two pages are exactly same, their similarity will be 1.0.

The reason for giving higher weight to higher level of web pages is because we think that higher path level usually more important than lower level. For example, people will think that the URL
http://foo.ca/research/labs/database/current.html

and the URL
http://foo.ca/research/labs/database/publications.html
are very similar although their last path level are different. By our similarity measure, the similarity between this two web pages is 0.93. This reflects the truth that the two URLs are very similar, but they are not identical.

4.2 Similarity Between Sessions

Using the similarity definition in the web page level, now we can define the similarity between two web sessions.

Our basic idea of measuring session similarity is to consider each session as a sequence of web page visiting, and use dynamic programming techniques to find the best matching between two sequences. In this process, web similarity technique discussed in the previous section serves as a page matching goodness function. The final similarity between the two sequences is based on their matching goodness and the length of the sequences.

One difference between our similarity measure and many of the previous works is: we consider session as a sequence, while many of previous results measure session similarity in either Euclidean space or sets, for example Jaccard Coefficient is widely used. The definition of Jaccard Coefficient is as follows:

$$sim(T_1, T_2) = \frac{T_1 \cap T_2}{T_1 \cup T_2}$$

By this definition, if two sessions contain no common web page, their similarity is 0; if two sessions contain same set of web pages, their similarity is 1.

We argue that a URL sequence can better represent the nature of a session than a set. For example, using Jaccard Coefficient similarity measure there is no difference between the session "**abcd**", "**bcad**" and "**abdc**". Using our session sequence similarity measure, it can tell you that the three are different, and "**abcd**' is more similar to "**abdc**" than to "**bcad**".

There are many papers [17] [19] in the area of bioinformatics area talking about sequence alignment. Their objects are DNA or protein sequences instead of web page sequences. One difference between web page sequences and DNA sequences is: Each DNA sequences contains a sequence of amino acids, and there are tens of different amino acids; However for web session sequences, each sequence contains a sequence of web pages, and there can be thousands of different web pages. Another difference between our web page sequences, i.e. web sessions and their protein sequences is that a protein sequence is typically hundreds of elements, while a session sequence is usually much shorter than a protein sequence. In our real session data set, the average session length is about 11.371 web pages. We don't need to consider some typical problems such as the tradeoff between memory efficiency and computational efficiency in protein sequence alignment.

We use a scoring system which helps finding the optimal matching between two session sequences. An optimal matching is an alignment with the highest score. The score for the optimal matching is then used to calculate the similarity between two sessions. These are the principles in matching the sequences:

- The session sequences can be shifted right or left to align as many pages as possible. For example, session#1 includes a sequence of visiting to URLs **1, 2, 21, 22**, here each web page is represented by its token string as described in the web page similarity part. Suppose session#2 includes a sequence of visiting to URLs **2, 21, 22**. The best matching between the two session sequences can be achieved by shifting session#2:

  ```
  session#1: 1  2  21  22
  session#2: -  2  21  22
  ```

  In our program, each identical matching, i.e. a pair of pages with similarity 1.0, is given a positive score 20; Each mis-matching, i.e. a pair of pages with similarity 0.0 or match a page with a gap, is given a penalty score -10. For a pair of pages with similarity α, where $0.0 \leq \alpha \leq 1.0$, the score for their matching is between -10 and 20. The final score for the best matching of this example pair of sessions is 50.

- Gaps are allowed to be inserted into the middle, beginning or end of session sequences. This is helpful for achieving better matching. For example, for the following two sessions, a inserted gap in session#2 helps getting the best matching. The final score for the best matching of the following pair of sessions is also 50.

  ```
  session#1: 1 2 21 22
  session#2: 1 2 -  22
  ```

- We do not simply count the number of identical web pages when we are aligning session sequences. Instead, we create a *scoring function* based on web page similarity measure. For each pair of web pages, the scoring function gives a similarity score where higher score indicates higher similarity between web pages. A pair of identical web pages is only a special case of matching – the *scoring function* return 1.0 which means the two pages are exactly the same. One example of matching non-identical pages is like following:

```
session#1: 1 2 21
session#2: 1 2 22
```

URL "**21** " in session#1 is matched with URL "**22**" in session#2, and the *scoring function* returns that the similarity between the two web pages is 0.67. The final score for the best matching of this pair of sessions is also 50.

The problem of finding the optimal matching can typically be solved using dynamic programming [10], and its process can be described by using a matrix as shown in Figure 4. One sequence is placed along the top of the matrix and the other sequence is placed along the left side. There is a gap added to the start of each sequence which indicates the starting point of matching. The process of finding the optimal matching between two sequences is actually finding a optimal path from the top left corner to the bottom right corner of the matrix. Any step in any path can only go right, down or diagonal. Every diagonal move corresponds to matching two web pages. A right move corresponds to the insertion of a gap in the vertical sequence and matches a web page in the horizontal sequence with a gap in the vertical sequence. A down move corresponds to the insertion of a gap in the horizontal sequence and matches a web page in the vertical sequence with a gap in the horizontal sequence.

In solving the optimal matching problem, the dynamic programming algorithm propagates scores from the matching start point (upper-left corner), to the destination point (lower-right corner) of the matrix.

The optimal path is then achieved through back propagating from destination point to starting point. In the given example, the optimal path found through back propagating is connected by arrows where the numbers in brackets indicate the step number in back propagating. This optimal path tells the best matching pattern.

The score of any element in the matrix is the maximum of the three scores that can be propagated from the element on its left, the element above it and the element above-left. The score that ends up in the lower-right corner is the optimal sequence alignment score [10]. One example of our matching process and its result is shown in Figure 4.

After finding the final score for the optimal session alignment, the final similarity between the two sessions is computed by considering the final optimal score and the length of the two sessions. In our method, we first get the length of the shorter session - $lengthShort$, then the similarity between the two sessions is achieved through dividing the optimal matching score by $20 * lengthShort$ because the optimal score can not be more than $20 * lengthShort$ in our scoring system. For the example in Figure 4, the similarity between the two sessions is $65/(20 * 5) = 0.65$.

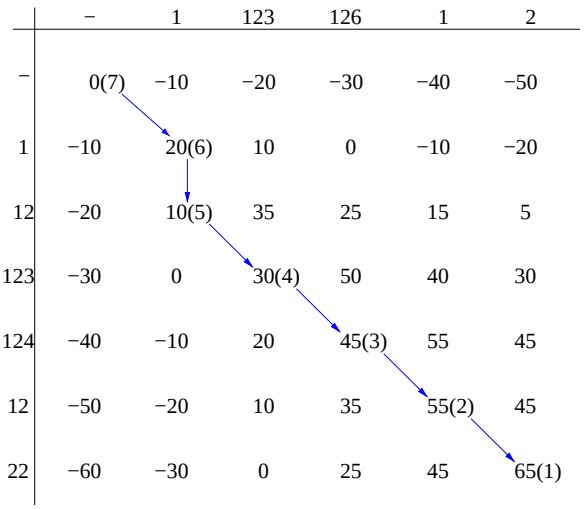

Figure 4: Session matching example

We argue that our similarity measure is better than previous set similarity measures, for example Jaccard Coefficient. This is due to two reasons: (1) considering session as sequence of URLs is better than considering session as a set of URLs. As mentioned before, Jaccard Coefficient cannot differentiate session "**abcd**" from "**bcad**" and "**abdc**", here each token "**a**, **b**, **c** and **d** represents a URL. Our method can not only tell the difference, but also precisely measure the cross similarity between each two of them. (2) In measuring the similarity between sessions, our method considers URL similarity. This has been proven effective in reflecting session similarity in many cases. For example, for the following two sessions:

```
session#1: 12  123  124
session#2: 1   125  126
```

The two sessions have no common URLs, but they are actually about a similar topic. Jaccard Coefficient will tell us that the similarity between the two sessions is 0.0, however our method tells that the two sessions still bear some similarity and their similarity is 0.67. This result better reflects the true connection between the two sessions.

5 Web Sessions Clustering

The session similarity method described in the previous section can be applied to compute the similarity between each pair of sessions, and construct a similarity matrix. Proper clustering algorithms will be applied to this similarity matrix to find the session clusters.

For the known clustering algorithms, we tried ROCK[5], CHAMELEON [9] and TURN [3] on our testing data set. K-modes is also applicable for categorical data [7], but its similarity measure is tightly based on vector space which is different from our sequence similarity measure, also considering the common problem of k-means family algorithms, which assumes clusters of spherical shapes , we did not try k-mode in our implementation. DBSCAN [2] and WaveCluster [16] could also be applied to some special categorical data sets, however, they require that any dimension of the categorical data space be somehow converted into numerical order. In this sense, they are not truly algorithms for general categorical data.

Another important issue is how to evaluate the quality of clusters in the result. Clustering Validation is a field where attempts have been made to find rules for quantifying the quality of a clustering result [11]. This issue, however, is a difficult one and typically people evaluate clustering results visually or compare to known manually clustered data. Visually inspecting clusters in 2-dimensional numerical data is to achieved by drawing the 2-dimensional clusters into 2-dimensional graphs. However, it is much harder to evaluate categorical data, like session data. What we did is to order the resulting clusters according to their descending sizes, and draw a three dimensional picture to represent the cross similarity between sessions in different clusters. For example, suppose for an ideal case that we have three clusters in a 1000 sessions data set, cluster #1 with 400 sessions, cluster #2 with 300 sessions, and cluster #3 has 300 sessions. The cross similarity between each pair of sessions within a same cluster is 1.0, and cross similarity between each pair of sessions from two different clusters is 0.0. For the 3-dimensional picture, we use x-axis and y-axis to represent 1000 sessions, z-axis is used to represent the similarity between the two sessions from x-axis and y-axis. For this given very simple and idealistic example, we expect to see a 3-dimensional picture like Figure 5. For this very special example, we see blocks along the diagonal of the x-y two dimensional space. This tells us the similarities between sessions within a same cluster are high, and the similarities between sessions which belong to different clusters are low. For real problems, the cluster result can rarely be so clear as the given example, but for successful clustering, we should see higher square areas along the diagonal of x-y space, and all the other areas should be lower.

Our testing session set used in our experiments has 1000 randomly selected sessions from a real e-learning system web log. Both Jaccard similarity and our Dynamic-Programming-Based similarity methods were used to provide similarity matrices for the given session set. ROCK, CHAMELEON and TURN were then applied on the similarity matrices to each produce clustering result. From the clustering results, we found that ROCK tends to find bigger clusters with lower average similarity. ROCK indeed tends

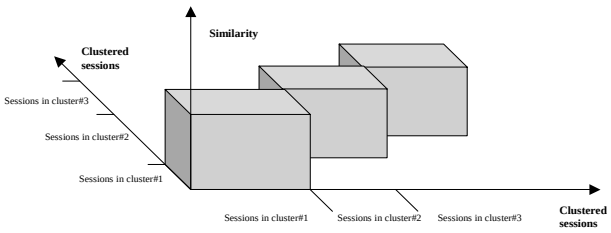

Figure 5: Session clustering result visualization example

to merge several genuine clusters into a bigger cluster. CHAMELEON and TURN can find clusters with high internal cross similarity. The difference between CHAMELEON and TURN is that TURN can identify outliers while CHAMELEON cannot. Rare sessions dissimilar to most other sessions are identified by TURN, while CHAMELEON forces them to belong to a given cluster.

Using the Jaccard Coefficient as a similarity measure for sessions tends to give more clusters than our Dynamic-Programming-Based similarity measure. In general when evaluated manually, the cluster quality between clusters using the Dynamic-Programming-Based similarity measure was better than when using the Jaccard Coefficient similarity measure. The clusters where simply more meaningful, which is an expected result since we took in consideration the sequence of clicks in a session. However, we do not currently have the means to compute quantitatively this cluster quality, and it would be very difficult to manually evaluate and compare the quality of the clusters resulting from the different similarity measures when the dataset is very large. Nevertheless, our method scales well with the size of the dataset to cluster, and we are confident, given our preliminary tests with the 1000 session set, that the web session clustering with sequence alignment would always yield more significant results than the commonly used approximation of sessions with sets.

6 Conclusions and Remarks

Session clustering is an important task in web mining in order to group similar sessions and identify trends of web user access behaviour. This is useful not only in e-commerce for user profiling, but also in e-learning for online learner evaluation. Accurate clustering of web sessions depends on good similarity measures between sessions. In this paper, through analysis and examples, we introduce a new similarity measure based on sequence alignment using dynamic-programming. This measure also considers the notion of similarity between pages. In our experiments, we compared the clustering characteristics of three algorithms (TURN, ROCK and CHAMELEON) on the session similarity measures: Jaccard Coefficient and Dynamic Pro-

gramming Based measure. Among the three algorithms, we determined that TURN was the winner based on our 3D graph for the visualisation of cluster "goodness". Our sequence alignment approach produced more meaningful clusters than the commonly used Jaccard coefficient. However, we do not have a quantitative measure to ascertain the righteousness of sequence alignment in session clustering with certitude. This can be achieved by testing the clustering on labelled data, where the exact cluster to wich a session should belong is known a-priori and hidden from the algorithm. Precise measure of the quality of clustering can be computed by comparing the results with the known cluster labels.

In addition to this specific evaluation, we are also considering sessions as sequences of learning actions rather than merely page accesses. Most requests in an e-learning site are requests to scripts generating dynamic pages. These accesses are mapped to learning actions such as sending a message to a forum, accessing a demo, answering a quiz question, reading a message or module, uploading an assignment, etc. Using a matrix representing similarity between learning actions, we are experimenting with clustering sessions of learning actions using our sequence alignment method as similarity between sequences of actions. Educators seem to better understand such clusters of action sessions than clusters of page access sessions.

References

[1] Arindam Banerjee and Joydeep Ghosh. Clickstream clustering using weighted longest common subsequences. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM2001)*, pages 33–40, Chicago, April 2001.

[2] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 1996 Int. Conf. Knowledge Discovery and Data Mining (KDD'96)*, pages 226–231, 1996.

[3] Andrew Foss, Weinan Wang, and Osmar R. Zaïane. A non-parametric approach to web log analysis. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM2001)*, pages 41–50, Chicago, April 2001.

[4] Yongjian Fu, Kanwalpreet Sandhu, and Ming-Yi Shih. Clustering of web users based on access patterns. *Workshop on Web Usage Analysis and User Profiling (WEBKDD99)*, August 1999.

[5] Studipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: a robust clustering algorithm for categorical attributes. In *15th Int'l Conf. on Data Eng.*, 1999.

[6] H.Mannila and C.Meek. Global partial orders from sequential data. In *Proc. 6th Intl. Conf. on Knowledge Discovery and Data Mining (KDD2000)*, pages 161–168, August 2000.

[7] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2:283–304, 1998.

[8] J.Pitkow and P.Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proc. 2nd USENIX symposium on Internet Technologies and Systems (USITS'99)*, October 1999.

[9] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, August 1999.

[10] K.Charter, J.Schaeffer, and D.Szafron. Sequence alignment using fastlsa. In *International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS'2000)*, pages 239–245, 2000.

[11] M. Vazirgiannis M. Halkidi, Y. Batistakis. On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2-3):107–145, December 2001.

[12] B. Mobasher, R. Cooly, and J. Srivastava. Automatic personalization based on web usage mining. Technical Report TR99-010, Department of Computer Science, Depaul University, 1999.

[13] B. Mobasher, M. Spilipoulou, and J. Wiltshire. Measuring the accuracy of sessionisers for web usage mining. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM2001)*, pages 7–14, Chicago, IL, April 2001.

[14] M.Spiliopoulou and L.C.Faulstich. Wum: A tool for web utilization analysis. In *Extended version of Proc. EDBT Workshop WebDB'98*, pages 184–203, Springer Verlag, 1999.

[15] C. Shahabi, A.M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *workshop on Research Issues in Data Engineering*, Birminghan, England, 1997.

[16] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: a multi-resolution clustering approach for very large spatial databases. In *24th VLDB Conference*, New York, USA, 1998.

[17] S.Needleman and C.Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[18] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations*, Jan 2000.

[19] T.Smith and M.Waterman. Identification of common molecular sequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[20] Osmar R. Zaïane. Web usage mining for a better web-based learning environment. In *Proc. of Conference on Advanced Technology for Education*, pages 60–64, Banff, AB, June 2001.

[21] Osmar R. Zaïane and Jun Luo. Towards evaluating learners' behaviour in a web-based distance learning environment. In *Proc. of IEEE International Conference on Advanced Learning Technologies (ICALT01)*, pages 357–360, Madison, WI, August 2001.

[22] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *1996 ACM SIGKDD Int. Conf. Managament of Data*, pages 103–114, June 1996.