
On Clustering Visitors of a Web Site by Behavior and Interests

Natascha Hoebel¹ and Roberto V. Zicari²

¹ Database and Information Systems, Computer Science Institute, J.W. Goethe University Frankfurt, Germany
hoebel@dbis.cs.uni-frankfurt.de

² Database and Information Systems, Computer Science Institute, J.W. Goethe University Frankfurt, Germany
zicari@cs.uni-frankfurt.de

Summary. This paper addresses the issue of how to define clusters of web visitors with respect to their behavior and supposed interests. We will use the non-obvious user profiles (NOPs) approach defined in [10], and present a new clustering algorithm which is a combination of hierarchical clustering together with a centroid based method with priority, which allows to cluster web users by similar interest in several topics.

1 Introduction

Companies today operate in an increasingly competitive environment. Therefore, finding and retaining customers is a major success factor for most businesses, off-line and online. One of the keys to building effective E-customer relationships is an understanding of consumer behavior online [12]. However, analyzing the behavior of customers online is not necessarily an indicator of their declared interest. In order to measure the supposed interest of a web user, we have introduced in [10] the concept of *non-obvious profiles* (NOPs).

This paper presents a new algorithm to define clusters of web visitors with respect to their behavior and supposed interests using the NOP approach defined in [10] and implemented by the Gugubarra Engine [8], [9]. The rest of the paper is structured as follows: In Sect. 2 we present our cluster algorithm. An example is given in Sect. 3. In Sect. 4 we briefly describe the current implementation and evaluation results. Related work is presented in Sect. 5 and conclusion in Sect. 6.

2 Clustering Algorithm

2.1 Initial Considerations

In this subsection we will look at how to define clusters, how to give an interpretation of a cluster, and how to partition users into clusters.

We will start with the following pre-considerations: To recognize trends in interests, we will not distinguish between interest weight values such as for example 0.3 and 0.35, but rather we will use a nominal scale: such as $\{no\ interest, little\ interest, strong\ interest, total\ interest\}$ or $\{small\ interest, medium\ interest, high\ interest\}$ etc.

For that we introduce the parameter g (granularity), which indicates how fine the owner would like to differentiate in the interest scale. For example, $g = 4$ means an interest scale composed of $\{no\ interest, little\ interest, strong\ interest, total\ interest\}$. We then split the range from Zero to One into g intervals and every interval has a centroid which tells us how to interpret the resulting cluster. The centroids are calculated with the function $G(x) = x/(g - 1)$ with $0 \leq x \leq g$. Note that these are not centroids in traditional sense, because they are fixed and are never recalculated as balance point of the users. They are centroids in the sense of representative of each cluster.

Supposed we want to define clusters based on only one topic Tp_i , using $g = 4$, we will get the four clusters with fixed centroids shown in Table 1.

Table 1. Concern topic Tp_i ; Intervals for $g = 4$ and scale of interest

Name	x	Centroid $G(x)$	Interval	Code	Interest
Cluster 1	0	0	[0.00; 0.16]	00	no
Cluster 2	1	0.33	[0.17; 0.49]	01	little
Cluster 3	2	0.66	[0.50; 0.83]	10	strong
Cluster 4	3	1	[0.84; 1.00]	11	total

For clustering over several topics $|Tp|$, we need an amount of maximum $g^{|Tp|}$ clusters with $|Tp|$ is the amount of topics. For example if we have $|Tp| = 2$ and $g = 4$ it results in 16 centroids. Note that this is the maximum number, that will normally be not reached by our algorithm. In our approach, besides defining the number of clusters, we also associate a *meaning* to each cluster in correlation to the scale of interest (see the cluster interpretation in Fig. 1).

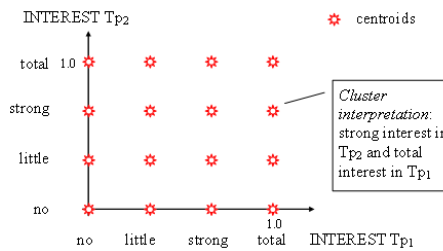


Fig. 1. 16 centroids for $g = 4$ and 2 topics Tp_1 and Tp_2

2.2 The Takahe Algorithm

In this section we present a new *cluster-by-priority algorithm*, that we call *Takahe*. Takahe is a divisive clustering algorithm, which creates a tree structure top down visualized in a dendrogram. A dendrogram is a tree diagram to visualize the arrangement of the clusters produced by a cluster algorithm. Every vertex of the dendrogram represents a cluster.

The algorithm sorts in the initialization the set of n topics, Tp_1, \dots, Tp_n , according to their priority. A higher priority is assigned to a topic if the interest of all current registered users u for this topic is on the average higher. The algorithm produces a sorted vector of n elements, each of which is the average value of the interest of all users for that topic. We call such vector, *the Community NOP (CNOP(t))*. The running time for the initialization is $O((\log u + |Tp|) * u)$.

The algorithm then determines the number of clusters which should be created, depending on the number and interest of the users and a predefined threshold-value. A threshold-value is used to reduce the number of topics considered by the algorithm. In the following we call the border topic *the threshold topic* Tp_T . In this way, we do not cluster users for topics that are not considered relevant. The running time for the termination condition is $O(1)$.

The algorithm then divides the set of users by sequential inspection of the sorted topics in the $CNOP(t)$. At the beginning, all users are placed into one cluster (top vertex in the dendrogram). Then we calculate the nearest centroid for the topic with highest priority in the $CNOP(t)$ for every user. This is a combination of hierarchical clustering with the k-means algorithm [6], as we search the nearest centroid instead of the nearest user. For the new centroid, we then add a child to the current cluster and a code for the centroid as an edge label.

Until the termination condition is fulfilled, we continue to partition the clusters successively (level for level to construct the dendrogram). We therefore have a single path clustering algorithm in relation to one topic because we do only one iteration. As we continue the algorithm under inclusion of a further topic until Tp_T , we keep the computation time of $O((\log u + |Tp|) * u)$.

The algorithm terminates when all topics in $CNOP(t)$ above the given threshold are considered and the users are distributed in the clusters created by the dendrogram. The pseudo code for the Takahe algorithm is as follows:

```

initialize(){
    calculate vector CNop;
    calculate MaxCentroids; }

determineTerminationCondition(){
    if(thresholdTopic not reached) terminate=false;
    else if(centroidsCount < maxCentroids) terminate=false;
    else terminate=true; return terminate; }

```

```

ClusterUsersByPriority(clusters,  $t_x$ ){
  for each cluster in clusters do
    for each user in cluster do
      centroidCode=getNearestCode( $w_x$  of user);
      if(child centroidCode not exists in cluster)
        add new child centroidCode to cluster;
        add new child centroidCode to children;
      add user to child centroidCode;
    if(!determineTerminationCondition())
      ClusterUserByPriority(children,  $t_{x+1}$ ); }

```

When the algorithm is finished, it visualizes the extraction of topic Tp_T (see for example Fig. 3). Then, if necessary the owner can display the clusters upwards in the tree (summary) or continue the division of the displayed clusters by inclusion of the next topic Tp_{T+1} .

The Takahe algorithm works well with high number of topics, because the threshold allows us to discard topics at the beginning that are considered not relevant for the clustering. A further way to reduce the topics is to build a hierarchy of related topics as e.g. used by [11].

3 An Example

We will now show with an example how the algorithm works. With $g = 4$ we define the interest scale (total, strong, little, no) interest. Thus we get the four clusters shown in Table 1 (see Sec. 2.1). Since $g = 4$, $x \in \{0, 1, 2, 3\}$, we can calculate the centroids, using the formula $G(x) = x/(g - 1)$. We obtain four centroids, with values 1, 0.66, 0.33 and 0. From the centroids we calculate the corresponding four intervals. We associate a code to each interval, that will be used to build the dendrogram. As result, each interval has a coded meaning in the interest scale e.g. *total interest*, defined by interval 0.84-1.00.

Let's assume we want to cluster 10 users (including Tom and Joe) visiting the web site www.frankfurt.de. For this web site we consider 5 topics: "Frankfurt at a Glance" (Tp_1), "Culture" (Tp_2), "Sports" (Tp_3), "Fair" (Tp_4) and "Zoo" (Tp_5). The NOPs for these users are shown in Table 2, where w_i corresponds to the value of the NOP for topic Tp_i . A value 0 means no interest; a value 1 means total interest.

In the initialization we calculate $CNOP(t)$, shown in Table 3. In the example, we have set a threshold value of 0.1. Therefore topic Tp_1 is the threshold topic Tp_T . If we look at Table 3, we see that topic Tp_5 will not be considered by the clustering algorithm. This is the mechanism we use to reduce the number of topics taken into account by the algorithm. The threshold is set by the owner of the web site and depends on the application requirements.

The owner might consider a subset of registered users for the calculation of $CNOP(t)$. Indeed the algorithm always works on a set of users, as the amount of users is a dynamic value and changes with time.

Table 2. NOPs at time t with 5 topics for 10 users

USERID	w'_1	w'_2	w'_3	w'_4	w'_5
1	0.3	0.9	0.1	0	0
2	0.27	0.84	0	0	0.1
3 (Joe)	0.3	0	0.65	0.24	0.3
4	0.9	0.5	0.5	0.4	0
5	0.2	0.68	0.12	0	0
6 (Tom)	0.29	0.15	0.45	0.16	0
7	0.3	0.7	0	0.4	0.24
8	0.05	0.05	0.6	0.9	0.1
9	0.3	0.6	0.1	0.2	0.1
10	0	0	0.55	1	0

Table 3. CNOP(t), sorted

	w'_2	w'_4	w'_3	w'_1	w'_5
$CNOP(t)$	0.44	0.33	0.31	0.29	0.08

We can now proceed to construct the dendrogram (see Fig. 2). We start by creating a root node (one cluster) in which we place all 10 users. We then consider the topic which has the highest value in $CNOP(t)$, that is Tp_2 , with $w'_2 = 0.44$. For all 10 users we look at their NOP value for Tp_2 defined in Table 2, and we check in which interval of Table 1, this value fits in.

In the dendrogram, we create an edge from the root with the appropriate code if we find at least one user who has an interest value which fits into the corresponding interval. For topic Tp_2 , we create three edges with code 11, 10 and 00 ending into the nodes A, B and C, in depth one. The nodes A, B and C are clusters of users with “similar” interest in topic Tp_2 . The algorithm proceeds by taking into account the next topic, which is Tp_4 until the threshold topic Tp_1 . Afterwards the algorithm ends. Fig. 2 shows the resulting dendrogram of the algorithm.

The algorithm visualizes the resulting clusters like shown in Fig. 3. A, B, C denote clusters for level 1 of the dendrogram. D, E, F, G, H denote clusters for level 2 and I, J for level 3.

In particular, when considering Joe and Tom: Joe is placed in cluster B, which means no interest in Tp_2 (“Culture”). Joe is also in the inner cluster E, which means little interest in Tp_4 (“Fair”) and Tp_1 (“Frankfurt”), strong interest in Tp_3 (“Sports”). Tom is also placed in cluster B, so Tom and Joe share this property. However, Tom is placed in another inner cluster F, which means no interest in (“Fair”) and little interest in “Sports” and “Frankfurt”.

Once we have clustered web users by NOPs, we now have a breakdown of users that can be targeted offering them personalized information and services,

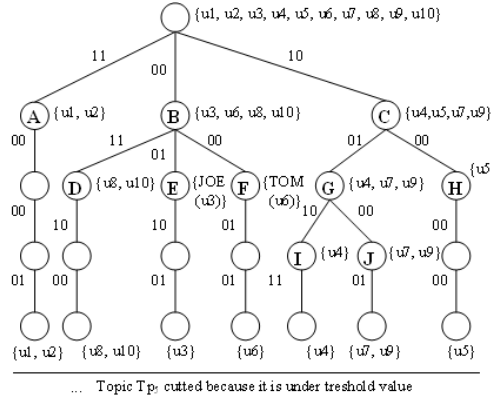


Fig. 2. Dendrogram of the clusters produced by the Takahe Algorithm with $g = 4$

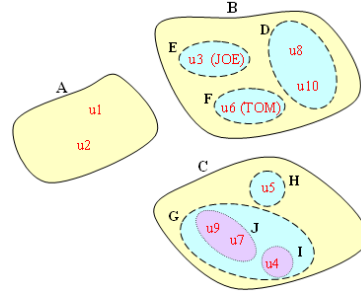


Fig. 3. Visual Extraction for Tp_1 (level 4) with $g = 4$

if and only if they request so. For example, as a result of this clustering we could avoid offering Tom information on “Culture” and/or “Fair” in “Frankfurt at a Glance”, but rather we could target him for a personalized information related to a major “Sports” event in “Frankfurt at a Glance”: the “World Soccer Cup 2006”.

Choosing the number of clusters and the scale of interest is a key for the clustering. Our approach is flexible and gives the owner of a web site the possibility to tune the clusters. The $G(x)$ does not necessarily have to be linear. The definition of function $G(x)$ is application dependent, it reflects the requirements of the application, and of course it influences how users at the end are clustered.

4 Implementation and Evaluation

We have implemented a prototype to evaluate the Takahe algorithm [5]. The module produces clusters of users based on the criteria and rules defined from the web site owner. It is a Java rich client application based on Eclipse 3.1.2 [3]. We used data sets with 1,000 (Set A), 2,500 (Set B), 5,000 (Set C), 50,000 (Set D) and 100,000 (Set E) users and 5 topics. We have benchmarked the Takahe algorithm ($g = 5$) against the k-means algorithm ($k = 20$) and against agglomerative hierarchical clustering using manhattan distance and complete linkage (HC) [2], [5]. We used these algorithms because they are widely known. In Table 4 the corresponding real execution times are shown using the 5 data sets. The test results confirmed the theoretical computation time of Takahe. The positive result is that Takahe depends $u \log u$ on the number of users. On the contrary, when using k-means on larger data sets, the algorithm produces an out-of-memory exception (OoM). The same happened with HC on Set C.

We have also compared the computation time of the three algorithms based on the number of topic. The results are presented in Fig. 4. Takahe has a computation time $O((\log u + |Tp|) * u)$, therefore it works well as long as the number of topics used is not too large.

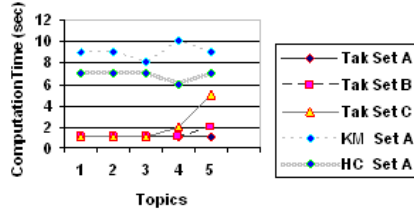


Fig. 4. Performance related to $|Tp|$

Table 4. Performance Results (sec)

	Takahe	K-Means	HC
SetA	1	9	7
SetB	2	OoM	86
SetC	8	OoM	OoM
SetD	22	-	-
SetE	37	-	-

As a result the Takahe algorithm can scale well for applications such as Web portals with a large set of registered visitors (e.g. 1,000,000) and hundreds of topics.

5 Related Work

Our clustering algorithm differs from most related work as it takes into account the time spent by the user on a page. Furthermore it looks at the topics associated to a set of pages, and clusters users based on their supposed interest in these topics. It gives an interpretation to the clusters as a value of a predefined scale of interest. Many related work cluster users on the base of sequences by visited pages, often also without inclusion of the time spent on it [4], [7]. Banerjee and Ghosh [1] include the time in a similarity function.

In some related work [4], [7], [13], [14], users are clustered over sequences of pages without looking at the content of the pages. We do not use sequences of pages; instead we look at each page's content to find the interests of the visitor. Instead of proceeding from a semantic structuring of a web site in folders [1], [4], [13], [14], we have automatically a reduction of the dimension by the allocation of the pages to topics.

6 Conclusion

We have presented a new algorithm to cluster web users based on their profiles. A word of caution is needed here, the authors are aware that this research area is sensitive in several issues, from the ethic perspective to the privacy and data protection issues. The technology presented in this paper needs to be carefully used and not misused. Users need to be aware of the existence of the profile, and most importantly they need to trust and know the usage of such profiles. This relationship of trust between the web site owner and his community is a of

importance. Improper use of profiles will potentially harm rather than help. It is the responsibility of all of us in the research community to raise the awareness of such issues.

Acknowledgments

The authors would like to thank the members of the Gugubarra team: B. Brandt, D. Guettinger, S. Kaufmann, N. Mustaq and K. Tolle, who gave valuable feedback, helped with the implementation of the algorithm and reviewed earlier drafts of this paper.

References

1. Banerjee A, Ghosh J (2001) Clickstream Clustering using Weighted Longest Common Subsequences. In: Proceedings Web Mining Workshop, Chicago
2. Chakrabarti S (2003) Mining the Web. M. Kaufmann, San Francisco
3. Eclipse Foundation. web resource www.eclipse.org
4. Fu Y, Sandhu K, Shih M (1999) Fast Clustering of Web Users Based on Navigation Patterns. In: Proceedings of SCI/ISAS'99, Orlando
5. Guettinger D (2006) Rich-Client-Entwicklung am Beispiel von Clusterverfahren und Benutzerprofilen. MA thesis, DBIS, JW Goethe University, Frankfurt
6. Hartigan J, Wong M (1979) K-Means Clustering Algorithm. Applied Statistics
7. Hay B, Wets G, Vanhoof K (2001) Clustering navigation patterns on a website using a Sequence Alignment Method. Limburg University Centre
8. Hoebel N, Kaufmann S, Tolle K, Zicari R (2006) The Design of Gugubarra 2.0: A Tool for Building and Managing Profiles of Web Users. In: Proceedings of Web Intelligence Conference, 18-22 December, Hong Kong
9. Hoebel N, Kaufmann S, Tolle K, Zicari R (2006) The Gugubarra Project: Building and Evaluating User Profiles for Visitors of Web Sites. IEEE Workshop on Hot Topics in Web Systems and Technologies, 13-14 November, Boston
10. Mushtaq N, Tolle K, Werner P and Zicari R (2004) Building and Evaluating Non-Obvious User Profiles for Visitors of Web Sites. CEC 04, 6-9 July, San Diego
11. OpenDirectory Project. web resource www.dmoz.org
12. Turban E et al. (2004) Electronic Commerce 2004. Pearson Prentice Hall
13. Wang Q, Makaroff D, Keith Edwards H (2004) Characterizing Customer Groups for an E-Commerce Website. In: Proceedings of the 5th ACM Conference on Electronic Commerce, ACM Press, New York
14. Wang W, Zaïane O (2002) Clustering Web Sessions by Sequence Alignment. In: Proc. of the 13th Int. Workshop on DB and Expert Systems Applications