ISSN 2723-6471

An Artificial Ant-Based Approach Using Polynomial Algorithms to Tackle the Text Aspect of Clustering Web Pages

Souad Moufok[1,*], Khaled Belkadi[2], Fatima Zohra Lebbah[3]

[1,2]*Faculté des Mathématiques et Informatique, Université des sciences et de la Technologie d'Oran Mohamed Boudiaf (USTO-MB), Laboratoire Signal-Image-Parole (SIMPA), BP 1505 El M'naouer, Oran 31000, Algeria*

[3]*Higher School of Electrical and Energetic Engineering of Oran, Laboratoire de Recherche en Informatique de Sidi Bel-Abes (LabRI-SBA), Equipe de Computational Intelligence and Soft Computing (CISCO), Ecole Supérieure en Informatique, Oran 31000, Algeria*

(Received: November 18, 2024; Revised: December 14, 2024; Accepted: January 17, 2025; Available online: March 4, 2025)

Abstract

Nowadays, the web clustering problem represents a scalable research area, which is based on deep study and efficient analysis of the user's browsing behavior. Managing huge amounts of unstructured data that are given through web pages is described as a hard and primary task. In this article, we analyze clusters by grouping users based on the similarity of the web pages they have visited. Our work focuses on cleaning, analyzing, and clustering web data to facilitate users' access to relevant content. Thus, we propose a novel algorithm, called WCLARTANT, to cluster WEB pages, which consists of finding groups of sessions according to the corresponding Web access patterns. We propose a new approach based on the ANTTREE algorithm, inspired from the self-assembling behavior observed in real ants and the binary search tree concept. The combination that we present in our approach is applied for the first time in web usage mining clustering. More precisely, different topologies are built in terms of different similarity measures, such as SBS, Euclidean, Jaccard and Cosine. Afterward, the clusters are extracted from the binary tree, which is built by the prefix depth algorithm. In other words, the proposed algorithms in this manuscript provide the corresponding binary tree to the sessions' matrix, where each node models a WEB session and each branch represents a cluster. In addition, we use the Silhouette index to evaluate and to analyze the clustering performance of WCLARTANT relative to the DBScan algorithm. WClArtAnt combined with the similarity measure SBS provides the best results compared to DBScan. The performance of our algorithm varies between 0.62 and 0.39, which are considered good. The considered log files are coming from NASA and contain all HTTP requests for a month period, from 1st July, 1995, to 31st July, 1995, for a total of 65,194 entries.

Keywords: Web session, ANTTREE, Binary Search Tree, Similarity Measure, WCLARTANT Algorithm

1. Introduction

Since the 90's, the Internet has marked our daily lives by changing our habits and purposes. Thanks to the appearance of the Web as a space to share information through websites, that makes Internet technology very popular. Each website is composed of web pages that are growing exponentially in terms of number. This phenomenal growth, in terms of the number of users visiting the websites, provides a huge amount of data, stored by web servers in log files that contain information about the actions of each user visiting the corresponding website during his navigation.

WEB developers analyze the log files to study deeply the browsing behavior, in order to improve the content and the structure of the website, and finally, to make the browsers' use simpler and easier. The raw nature of the data saved in log files requires a pre-processing step to remove navigation patterns that are unnecessary for web clustering techniques. This step identifies WEB sessions, where a session is composed of a set of WEB pages consulted by a user. Because of the exponential number of Web sessions, Web developers use clustering techniques to classify browsing sessions into groups of users called clusters. The sessions contained in a cluster should have similar browsing behavior. An efficient clustering process does not depend only on the selected algorithm. On the other hand, a meticulous choice of the similarity measure between sequences of WEB sessions is really important.

***Corresponding author: Souad Moufok (souad.moufok@univ-usto.dz)**
DOI: https://doi.org/10.47738/jads.v6i2.546

ISSN 2723-6471

In this work we propose a new method WCLARTANT, based on the ANTTREE algorithm. The original Ant-tree algorithm proposed by Azzag et al in 2003, is cited and modified in [1]. These two aspects allow the construction of a binary tree, where each node models an ant, which represents a WEB session to be grouped, and an ant's movement in the tree is done according to its similarity measure. In addition, the silhouette index computation is used to evaluate the clustering performance obtained via our algorithm WCLARTANT against DBSCAN, using different similarity measures, namely SBS, Euclidean, Jaccard and Cosine. These implemented algorithms are applied to web log files coming from the NASA database. In this work, we modify and improve the proposed algorithms in the data clustering field combined with our proposed binary structure for hierarchical clustering. By using an incremental process, the ANTTREE method builds a living structure, where ants are progressively attached to the existing support, and the remaining ones are successively attached to other previously treated ants. Thus, the final binary tree is constructed, where each branch represents a web cluster. More precisely, the movement of an ant ai from a node to another is determined through a similarity measure between ai and an already connected ant. The AntTree algorithm [2], [3].

The rest of this paper is structured as follows. In Section 2, we introduce the necessary definitions of web mining and related works. In Sections 3 and 4, we present our algorithm in detail and the results of our approach and DBSCAN applied to NASA log files with a deep analysis of the provided schedules. Finally, in Section 5, we conclude and explain future steps of our work.

2. Problem Definition and Related Works

Web Usage Mining (WUM) aims to treat web data mining deeply, to make the web users' tasks easier, simpler, faster, more secured and more interesting. In the literature of web mining, a research work differs from another according to the considered characteristics and the desired objectives.

In [4] a new method for clustering web sessions using a recently developed algorithm called FOGSAA (Fast Optimal Global Sequence Alignment Algorithm), and the authors in [4], [5] adopted the Hierarchical Agglomerative Clustering technique, for regrouping patterns based on the sequence of occurrence of WEB pages and T. M. Shami et al. in [6] proposed a fuzzy clustering technique. Moreover, other proposed clustering methods are coming from natural behaviors, such as Swarm Intelligence (SI) to tackle the web sessions clustering problem [7], and flocks of agent-based clustering [8], inspired from flocks of birds.

On the other hand, each proposed approach takes into account different characteristics, such as [9] where the user's IP - address is considered, [10] which achieves secured communication. In addition, most algorithms found in the literature are based on the k-means method [11], or CURE algorithm [12], DBSCAN [13] and k-medoids [14].

The authors analyze and categorize swarm intelligence studies [15], [16], focusing on their applications and providing a classification scheme that includes agent-based data clustering methods, and Ant-Clustering, inspired by the natural behaviors of ants, such as brood sorting [17], foraging based on the pheromones of real ants [18], AntClust using colonial odors [19] and using the self-assembly behavior of real ants for hierarchical clustering [2]. The other researchers [20], the authors introduced a new hierarchical clustering and visualization technique called SOT (Self-Organized Tree), which is based on the SOM (Self Organizing Method).

In this work, we focus on three fundamental aspects that we exploit in our proposed approach: similarity measurement techniques, binary tree structures and an efficient web clustering algorithm.

As given in [13], DBSCAN is a density-based clustering algorithm, known for its robustness against outlines. It is capable of detecting both clusters and noise within a spatial database [13], [21], [22]. The execution of the DBSCAN algorithm requires two input parameters: Eps and Minpts, where Eps defines the radius of the neighborhood region, and Minpts represents the minimum of neighbors in Eps.

A point P in a dataset can be categorized into three types based on its neighbors. It is considered a core point if the count of its neighbors is greater than or equal to MinPts. If the count of P's neighbors is less than MinPts but P lies within the Eps neighbors of a core point, it is classified as a border point. Finally, a noise point is defined as a point that does not qualify as either a core point or a border point.

 ISSN 2723-6471

This clustering algorithm can be used in different applications, such as segmenting e-commerce customers [23], analyzing a log web server [24], [25], healthcare [26]. Since DBSCAN can be adapted to be applied to the same type of Web Mining problem that we tackle in this paper, we will take it into consideration in the experiments part to be compared to our proposed algorithm. The DBSCAN's flowchart is illustrated in figure 1.

BSCAN process can be described in four sequential steps. First, a point P is randomly selected from the given dataset D. Next, clusters are identified within the region defined by a radius Eps around this point. If the Eps-neighborhood of point P contains at least MinPts objects, a new cluster is formed with P as the central object. Subsequently, data objects with direct density reachability are iteratively identified from these central objects, which may result in the merging of densely connected clusters.

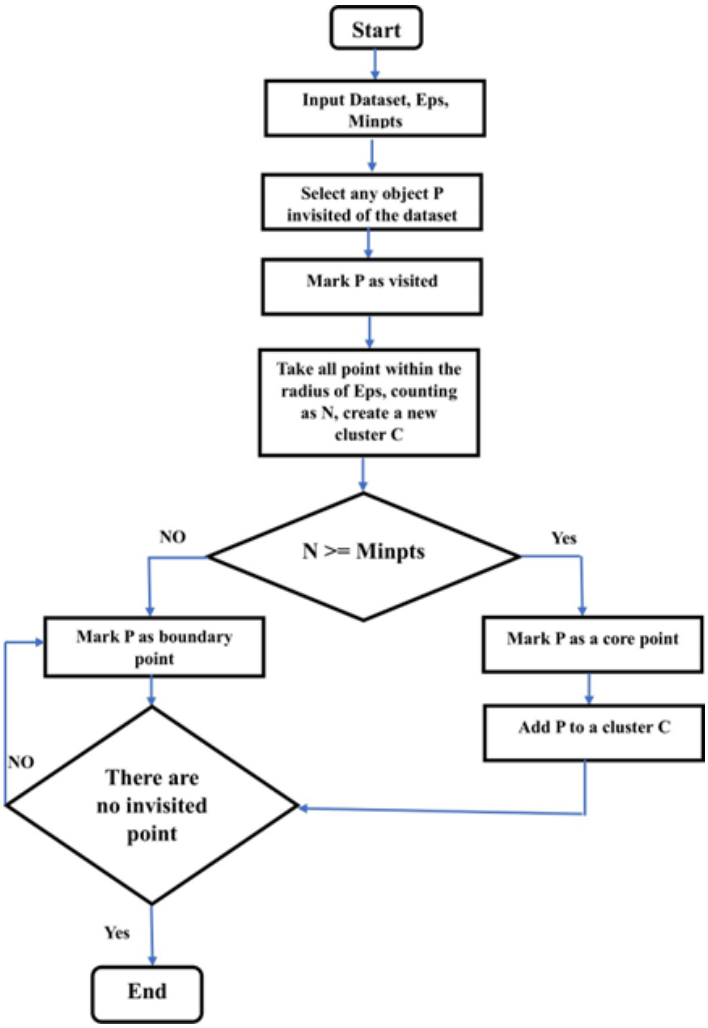


Figure 1. DBSCAN's flowchart.

DBSCAN is the method considered in this article due to several advantages. Unlike K-means, which requires specifying the number of clusters in advance and does not handle outliers, DBSCAN does not require a predefined number of clusters and effectively eliminates outliers. Additionally, DBSCAN relies on two principal hyperparameters, eps and MinPts, which are straightforward to set, whereas CHA requires specifying both the distance metric and the aggregation method, and K-means depends on the K parameter, which often necessitates the use of heuristics for proper determination. Furthermore, unlike K-means, which struggles to generate clusters with varying densities, DBSCAN excels in this aspect.

Thus, compared to K-means and CHA, DBSCAN is more efficient since it is a density-based clustering algorithm that finds high-density regions and outliers, and easy to implement since it does not require specifying the number of clusters and works well with clusters of arbitrary shapes.

3. Methodology

3.1. Similarity Measurement Techniques

Several similarity measuring techniques exist in the literature to identify the similarities between WEB sessions. The other researchers [4] consider the sequences of sessions that have the same length and use the normalized cosine of the angle between two vectors as a similarity measure. When the sequences of sessions are of different lengths and the elements' order is ignored, the Jaccard measure (see Definition 2) is adopted. We can also mention the works [27] where the similarity is measured by determining the longest common sub-sequence (LCS) between the two sequences of pages visited by each user, and [28], the similarity between web sessions is computed according to the principle of sequence alignment in bioinformatics [29]. The other researchers [30], the authors address identifying user groups based on their web navigation patterns, using sequence alignment and similarity matrix thresholding. A new method was proposed to measure the similarity between WEB sessions while creating subsets of representative sessions [4]. The authors consider that two sessions are similar if one session is a subset of another and the sequence of the URLs navigated by a user for a given time frame is a session.

In our article, we propose new algorithms based on the principle of the sessions' inclusion [4]. We take into consideration the similarity measures: SBS (see figure 2) ecluidean (see Definition 3), Jaccard (see Definition 2) and Cosine (see Definition 1).

3.1.1. Similarity Between Sessions (SBS):

To realize an efficient study and analysis of the users' paths during a determined period, we exploit the algorithm SBS (Similarity Between Sessions) [4]. This algorithm introduced by Anupama et al., allows to compute the similarity matrix from a 0-1 matrix describing the users' paths. The authors use an (n×m) 0-1 matrix called S, where n is the sessions' number and m the number of the visited pages. In other words, all the visited pages by a specific user are represented by a specific row (called vector) of the matrix. More precisely, SBS metric is characterized as follows:

It is particularly suited to group web sessions because it takes into account the order and sequence of pages visited, which allows capturing user behaviors that are often sequential and hierarchical, and it is ideal when the goal is to understand how users navigate from one page to another. It is essential for analyses such as content personalization or user journey optimization, where the order of pages is significant.

Thus, we use algorithm SBS computes the similarities between sessions providing the similarity matrix, called Sim, which will be the input of our algorithm WCLARTANT.

In our case, the sessions can be of different sizes and the visiting order of the pages is not considered. Thus, we opted for SBS, which only takes into account the size of the session. According to the SBS algorithm, the similarity between the sessions Si and Sj is computed as follows (see (1)):

$$sim(S_i,S_j) = \begin{cases} 0 \quad if\ S_i = S_j, \\ S_j.length \quad if\ S_i.length > S_j.length \\ S_i.length \quad if\ S_j.length > S_j.length \end{cases} \tag{1}$$

Si.length is the number of the visited pages by the user i, or during the session I, and $sim(S_i,S_j)$ is the similarity between the sessions Si and Sj.,

In figure 2, we illustrate the behavior of the SBS algorithm on the matrix S (n × m), which contains n sessions: S1, S2, … Sn. SBS is based on two steps. The first one computes the similarities between each pair of sessions, and the second computes the similarity matrix.

 ISSN 2723-6471

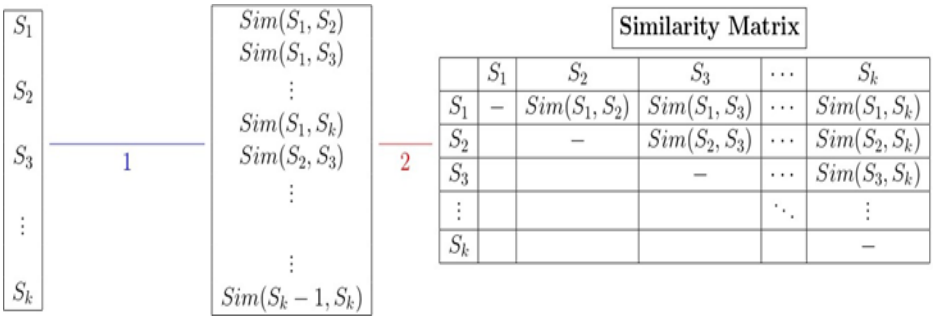


Figure 2. Computing similarity matrix from sessions matrix.

Algorithm 1 in figure 3, called SimComp introduces a method based on the SBS principle, to compute the similarities between pairs of sessions.

Algorithm 1 *SimComp*

Require: S_i, S_j :Sessions' matrix
Ensure: Similarity matrix between S

```
for i := 1 to n do
    for j := i + 1 to n do
        if S_i ⊂ S_j ∨ S_j ⊂ S_i then
            Sim(S_i, S_j) ← min(S_i.length, S_j.length)
        else
            Sim(S_i, S_j) ⇐ 0
        end if
    end for
end for
```

Figure 3. Pseudo code SimComp algorithm

For instance, consider table 1 which describes the 0 − 1 sessions' matrix S(8 × 6). An element Si,j of S is defined as follows:

$$S_{ij} = \begin{cases} 1 & if\ page\ P_i\ is\ visited\ throught\ the\ session\ S_i \\ 0 & else \end{cases} \quad (2)$$

Si represents a session and Pi is the page visited by Si.

Table 1. An instance of sessions' matrix.

	P1	**P2**	**P3**	**P4**	**P5**	**P6**
S1	1	1	1	1	1	0
S2	1	1	1	0	1	1
S3	0	1	1	1	1	0
S4	0	0	1	1	0	0
S5	0	0	1	0	1	1
S6	0	1	1	0	0	0
S7	0	1	1	1	0	0
S8	0	0	0	0	1	1

Table 2 gives the similarity matrix Sim(8×8), which is the result of Algorithm 1 applied on the matrix S (see table1)

 ISSN 2723-6471

Table 2. The corresponding similarity matrix to the sessions' matrix given in Table 1.

	S1	S2	S3	S4	S5	S6	S7	S8
S1	-	0	4	2	0	2	3	0
S2	0	-	0	0	3	2	0	2
S3	4	0	-	2	0	2	3	0
S4	2	0	2	-	0	0	2	0
S5	0	3	0	0	-	0	0	2
S6	2	2	2	0	0	-	2	0
S7	3	0	3	2	0	2	-	0
S8	0	2	0	0	2	0	0	-

Figure 4 gives the implementation result of SBS on the sessions matrix given in table 1, where figure 4a, figure 4b, figure 4c, figure 4d, figure 4e , figure 4f show the computing process of sim(S1, S2), sim(S1, S3), sim(S1, S4), sim(S2, S3), sim(S2, S4) and sim(S3, S4), respectively.

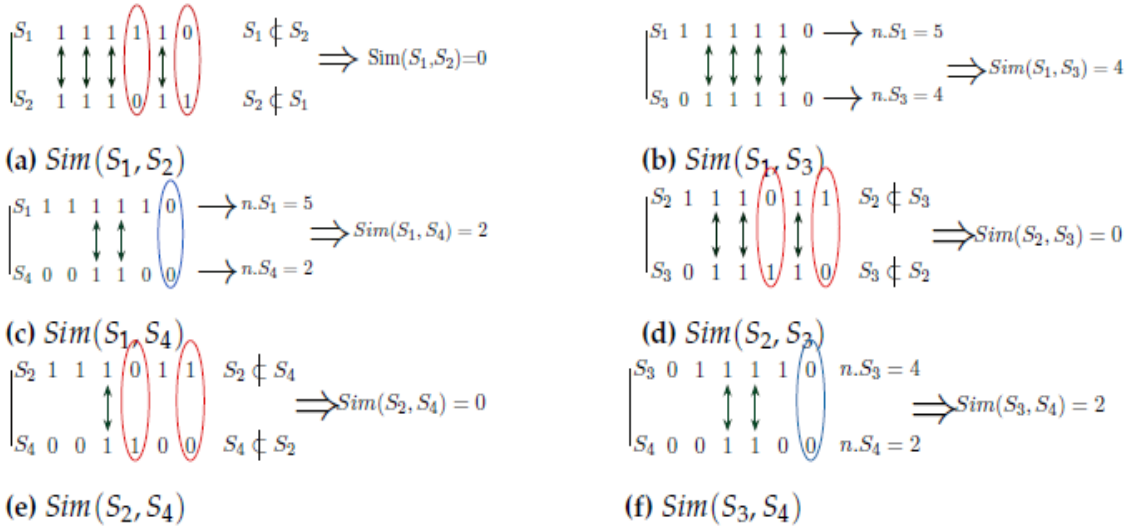


Figure 4. Implementation result of SBS on the sessions, matrix given in table 1.

3.1.2. Cosine similarity Measure

As given in Definition 1, the cosine similarity measure evaluates the similarity between two vectors in a multidimensional space. In the context of web sessions, each session's page visits are depicted as vectors, with each component representing a visited page and assigned a unique value.

The characteristics of this similarity measure are as follows: it evaluates the angle between two vectors, focusing on their direction rather than magnitude. This approach is commonly used in natural language processing and can be applied to web sessions by representing each session as a vector of the frequencies of visited pages. It is particularly effective for assessing the direction of vectors, meaning it identifies similar behaviors in sessions with common pages. However, it does not consider the order of visited pages, which limits its effectiveness in scenarios where the navigation sequence plays a crucial role.

Definitions 1 (cosine similarity measure). Cosine similarity calculates the angular distance between two sessions, treating them as vectors within a vector space. Specifically, cosine similarity is defined as the cosine of the angle between the two vectors, capturing a scale-invariant understanding of similarity. The cosine distance between two vectors, denoted as Si and Sj, is defined as follows in (3):

 ISSN 2723-6471

$$Sim(S_i, S_j) = \frac{\sum_{i,j=1}^{k} S(i,n) * S(j,n)}{\sqrt{\sum_{i,j=1}^{k} S(i,k)^2} * \sqrt{\sum_{i,j=1}^{k} S(j,k)^2}} \quad (3)$$

Si and Sj are vectors that represent the objects or sessions to be compared. Each vector contains values associated with specific dimensions, for example, characteristics or attributes of web sessions, and S(i,n) and S(j,n) denote the nth components (or values) of the vectors Si and Sj, respectively.

3.1.3. Jaccard Measure

On the other side, there are measures that take into account the sequences of sessions having different lengths, while the elements' order is ignored and the Jaccard measure is adopted (see Definition 2). We can also mention the works [27] where the similarity is measured by determining the longest common sub-sequence (LCS) between the two sequences of pages visited by each user, and [28], the similarity between web sessions is computed based on the principle of sequence alignment in bio-informatics [29].

Definitions 2 (Jaccard similarity measure). The Jaccard measure is based on the presence or absence of sessions. It is calculated by dividing the number of common sessions by the total number of sessions in the two sets. is defined, as fellow in (4):

$$Sim(S_i, S_j) = \frac{a}{a + b + c} \quad (4)$$

a: the number of times where S(i, k) = S(j, k) = 1/k ∈ {1, ..n} − {i, j},

b: the number of times where S(i, k) = 1 and S(j, k) = 0/k ∈ {1, ..n} − {i, j},

c: the number of times where S(i, k) = 0 and S(j, k) = 1/k ∈ {1, ..n} − {i, j}.

The context of Jaccard's use and its advantages are outlined as follows: the Jaccard coefficient measures the similarity between two sets by comparing the number of common elements to the total number of elements. In the context of web sessions, it is used to compare the pages visited by different users. This measure is particularly useful when the focus is on the presence or absence of visited pages, regardless of their order. However, it may underestimate similarity when the sets of visited pages are small or have limited overlap.

3.1.4. Euclidean Similarity Measure:

As described in Definition 3, Euclidean distance is a commonly utilized metric for measuring the distance between vectors in a vector space.

Definitions 3 (Euclidean similarity measure). It is the square root of the sum of the squared differences of the corresponding dimensions of the vectors. Mathematically, it is calculated using (5):

$$Sim(S_i, S_j) = \sqrt{\sum_{i,j=1}^{k} |S(i,n) - S(j,n)|^2} \quad (5)$$

Si and Sj are the web session vectors to compare, k is the sessions' number, and n is the number of pages visited by users i and j.

Our choice of this similarity measure is based on the following reasons: Euclidean distance measures the similarity between vectors in a multidimensional space. In the case of web sessions, each session is represented as a vector, where each dimension corresponds to a visited page (or an attribute of the session). This measure is simple and intuitive, as it does not consider the order of the visited pages. However, this can lead to less accurate results in analyses where the sequence of pages plays an important role.

3.2. Proposed Clustering Model

In this paper, we propose a new WEB clustering algorithm based on the Binary Search Tree principles, where a node models a session, a branch represents a cluster, and the prefix depth traversal algorithm is adopted. A binary tree is a data structure (see Definition 4) widely used in several disciplines, such as packets' classification [31], [32] unsupervised-data clustering [33] and multi-class data classification [34]. Definitions 4 (binary tree). [35] A binary

tree is a hierarchical structure composed of nodes, where each node has at most two children: left and right. Each node has a parent direction, which defines the child's type (right or left). It is defined as an unconnected graph composed recursively of left and right binary sub-trees, connected by a common node called the root.

Figure 5 illustrates an example of a binary tree and introduces the basic components of its structure, defined as follows: S1 is the root, representing the initial node of the tree without any children. Each node can have at most two children, positioned as left and right, such as S2 and S5, which are the only children of S1. The nodes S4, S7 and S9 are considered the leaves of the tree, as they do not have any children. Additionally, except for the root, every node has a parent, like S5, which is the parent of S6 and S8.

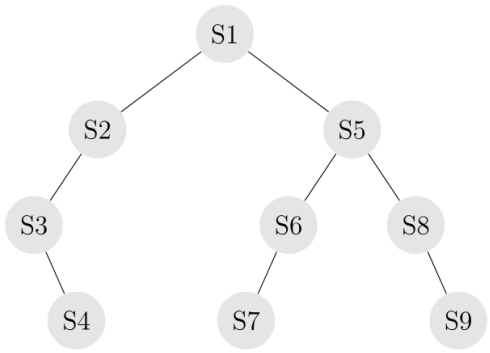


Figure 5. Example of binary tree structure

Searching for a node with a particular label is a recursive process. We start by examining the root. If the label of the root is the label we are looking for, the algorithm ends and returns the root. If the label we are looking for is less than, then it is in the left subtree, which we then recursively search. Similarly, if the label we are looking for is strictly greater than the label of the root, the search continues on the right subtree. If we reach a leaf whose label is not the one that we are looking for, then we know that this label is not in the tree.

Table 3 provides additional details about the binary tree illustrated in figure 5, as follows: n.index represents the node index, which holds information, data, or session details. p.n.index indicates the parent node index, specifying the name or identifier of the node with at most two children. p.r.direction denotes the parent's relative direction, indicating whether the node is positioned as left or right in relation to its parent node. The term " left child " refers to the first child of the immediate ascendant node, while " right child " refers to the second child of the immediate ascendant node.

Table 3. Parents' direction for nodes in the binary tree illustrated in figure 5.

n.index	p.n.index	p. rel. direc.	left child	right child
S1	None	None	S2	S5
S2	S1	Left	S3	None
S3	S2	Left	None	S4
S4	S3	Right	None	None
S5	S1	Right	S6	S8
S6	S5	Left	S7	None
S7	S6	Left	None	None
S8	S5	Right	None	S9
S9	S8	Right	None	None

The characteristics of a binary search tree (see Definition 5) lead to simple managing algorithms that allow simple manipulation on the nodes, such as searching, adding and deleting a node.

In the field of web session clustering, the use of a binary tree is based on its hierarchical and structured organization, where each node represents a session or a group of sessions and has at most two subgroups. This structure allows for

efficient and fast similarity searches, facilitating the processing of large volumes of data from log files. Sessions are inserted into the tree based on specific similarity criteria, such as page views or browsing behaviors, and are recursively divided into coherent subgroups. This approach optimizes data partitioning while providing high adaptability to dynamic environments, where new sessions can be integrated without completely reorganizing the tree. Thus, the binary tree is an effective method to improve scalability and accuracy in web session clustering.

Definitions 5 (binary search tree). A binary search tree is a binary tree with the following properties: all the keys in the left sub-tree are smaller than the key of the root, while all the keys in the right sub-tree are greater than the root's key. Additionally, both the left and right sub-trees must also adhere to the binary search tree properties. In other words, as given in figure 6,

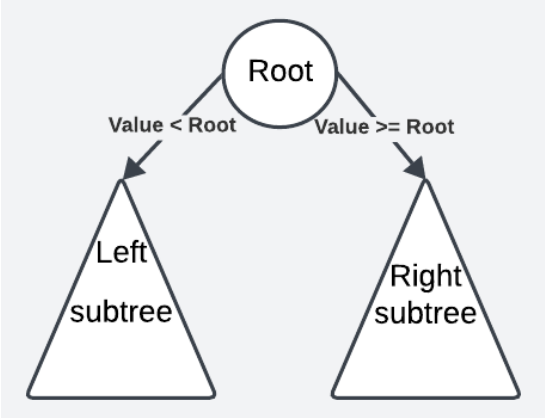


Figure 6. Rote-childs structure

In other words, by positioning the sessions on the nodes of the tree with respect to the binary search tree principle, the nodes of a branch of the tree provided by WCLARtAnt represent the sessions that form a cluster.

Since the binary search tree is constructed from our dataset, a list of nodes can be extracted by traversing the tree in four different ways: prefix depth traversal, where nodes are visited in the order of (root, le f t child, right child); infixed depth traversal, where the nodes are visited in the order of (le f t child, root, right child); suffixed depth traversal, where the root is visited last, following the order of (le f t child, root, right child); and breadth-first traversal, where the nodes are visited level by level, from left to right.

A binary search tree is used in general for data search. The authors [36] proposed an algorithm for building a binary search tree to identify the longest prefix match in the IP address lookup. The search is based on the comparison between two prefix values of different lengths.

3.3. WCLARtAnt Proposed Algorithm

In this section, we introduce our clustering algorithm, called WCLArtAnt. It provides a binary tree BT, which is inspired from the ants' self-assembly behavior. Precisely, we utilize the principle of the algorithm Antree proposed by Azzag et al. cited by other researchers [1], who exploit the search tree principle to tackle the documents' classification problem.

We conceived the algorithm WCLArtAnt (Algorithm 2) to build a hierarchical tree, where each node models an ant representing a session. The algorithm can be summarized in two steps expressed by two algorithms, namely InitTree (see Algorithm 3) and FinalBTTree (see Algorithm 4), to the initial binary tree and the tackling step, which provides the desired (or final) binary tree, respectively. In WCLArtAnt, we consider the variables Si and Sim(Si, Sj), that we define as follows:

Let Si represent the ith ant (ith session) to be classified, and Sj represent another session in the set, where j is a different index than i. For example, if Si is S1, Sj , then Sj could be S2, S3, and so on. Each session Sj is compared with Si among all the sessions compared, but it has not yet been placed in the binary tree. Sk is a session that is temporarily labeled as having the highest similarity value to Si among all the sessions that are compared but has not yet been placed in the binary tree. Once all sessions with lower similarity values have been placed, Sk is inserted into the tree on the

 ISSN 2723-6471

left side of the root. SR represents another session already connected to the tree, distinct from Si, Sj, and Sk , that has a higher similarity value than Sk . In other words, SR is a session that, when compared to Sk , has a higher similarity measure. Finally, Sim(Si, Sj) is the similarity measure between the distinct sessions Si and Sj, for i, j ∈{1, ..N}, where N is the sessions' number.

In InitTree, we start by positioning the session S1 on the root of the empty tree BT. Then, the remaining sessions Sj, j = 2. N, will be connected to S1, according to the search tree principle (see Section 3.2). The position of each session Sj is computed interms of the similarity measure Sim(S1, Sj). In other words, Sj is connected in the right side or the left side of the root S1, as follows:

if Sim(S1, Sj) = 0, then Sj is placed in the right side of the root. If Sim(S1, Sj) is equal to the greatest value of the previously placed session, then Sj is placed in the right side. If Sim(S1, Sj) is greater than the values of the previously placed sessions, then Sj, labeled Sg, will be considered after positioning all the sessions with smaller values than Sim(S1, Sj). If the session Sg, still have the greatest value, then it should be placed in the left side of the root. If Sim(S1, Sj) > Sim(S1, Sg), then Sg is placed in the right side and Sj will be labeled Sg (figure 7 and figure 8).

Algorithm 2 *WCLArtAnt*

Require: *Sim*: similarity matrix
Ensure: *BT*: binary tree
1: *InitTree* algorithm
2: *FinalBTTree* algorithm

Figure 7. Pseudo code WCLARtAnt algorithm

Algorithm 3 *InitTree*

Require: *Sim*: similarity matrix between
Ensure: *BT*: Binary tree
1: $i \leftarrow 1; L \leftarrow 0; K \leftarrow 0; R \leftarrow 0;$
2: **for** $j = i + 1$ to n **do**
3: **if** $Sim(S_1, S_j) = 0$ **then**
4: $Connect(S_j, S_1, rightside); S_j.statut \leftarrow 1;$
5: **else**
6: **if** $Sim(S_1, S_j) = L$ **then**
7: $Connect(Sj, S1, right.side); S_j.statut \leftarrow 1;$
8: **else**
9: $L \leftarrow max(L, Sim(S_i, S_j);$
10: **if** $L = Sim(S_i, S_j)$ **then**
11: **if** $K = 0$ **then**
12: $K \leftarrow j;$
13: **else**
14: $R \leftarrow K; Connect(S_R, S_1, right.side); S_R.statut = 1; K \leftarrow j;$
15: **end if**
16: **else**
17: $Connect(S_j, S_1, right.side); S_j.statut \leftarrow 1;$
18: **end if**
19: **end if**
20: **end if**
21: $Connect(S_K, S_1, left.side); S_K.statut \leftarrow 0;$
22: **end for**

Figure 8. Pseudo code InitTree algorithm

 ISSN 2723-6471

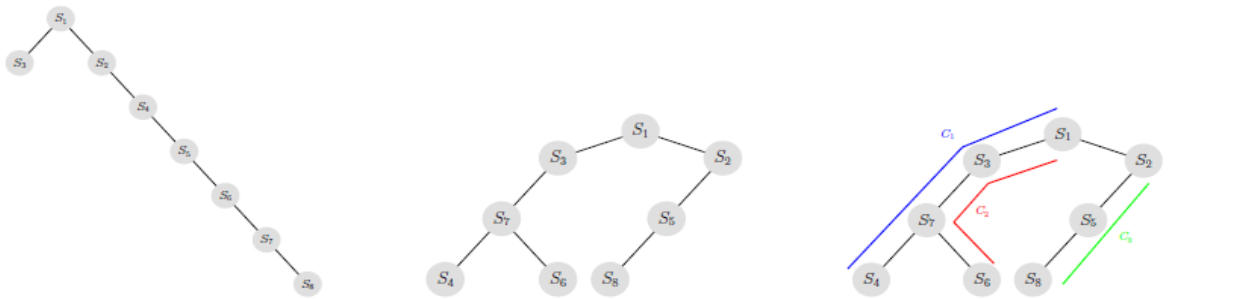
Since we obtain the initial binary tree BT, the second step is launched via the algorithm FinalBTTree (Algorithm 4), which is based on the moving process of the sessions from a position to another according to their similarity measures and their status (left child or right child). This process, which is based on iterated instructions, allows to create new clusters by moving sessions with respect to their similarity measures. In other words, each session

Si (i = 2..N) is compared to all the other sessions Sj (j = i + 1..N).

The principle of the moving process adopted in Algorithm 4, is detailed as follows:

If Sim(Si, Sj) is maximal, Sj and Si are connected to the left side and the right side of the same session, respectively. Two scenarios are possible: first, Sj does not have the left child, where Si is similar to Sj and it is moved to become the left child of Sj. Second, Sj has a left child, then Si is moved to be connected to the right side of Sj, becoming a leaf node.

If Sim(Si, Sj) is maximal (has the greatest value), and Sj is connected to the right side of Si, then Sj is put aside, until tackling the remaining sessions. At this level, two possibilities are occurred: first, Sj still have the maximal value, and Sj is considered as similar to Si. Thus, it is moved and connected to the left side of Si, becoming a leaf node. Second, another session Sa with the greater similarity value then Sj appeared, then Sj is connected to the right side of Si, becoming a leaf node, and Sa is put aside. Figure 9b shows the result of FinalBTTree (Algorithm 4) applied on BT, which was previously provided by InitTree (see Figure 9a).



(a) *BT* resulted from *InitTree*, applied on the similarity matrix given in Table 2

(b) *BT* provided by *FinalBTTree* applied on the initial binary tree given in Figure 5a

(c) *BT* resulted from *FinalBTTree*, traversed in depth first pre-order.

Figure 9. Implementation result of algorithms (InitTree, FinalBTTree).

As mentioned below, WCLArtAnt generates the corresponding binary tree BT to the similarity matrix Sim, where each branch from the left to the right represents a cluster. Figure 9c illustrates the process of extracting clusters from the tree. The first session, S1, serves as the root of the binary tree (BT). The tree is then traversed in depth using a pre-order traversal. The tree is divided into two sub-trees, ST1 and ST2, with S1 and the right child of S1 as the respective roots. Each branch of ST1, from the root to the leaf, represents a cluster, with the nodes modeling the sessions within that cluster. This process is repeated for the second sub-tree ST2 until ST1 becomes a leaf.

Consequently, if we apply our algorithm (Algorithm 4 in figure 10) on the similarity matrix given in table 1, we obtain three clusters (see Figure 9c):

Cluster1 = {S1, S3, S7, S4}

Cluster2 = {S1, S3, S7, S6}

Cluster3 = {S2, S5, S8}

 ISSN 2723-6471

Algorithm 4 *FinalBTTree*

Require: *Sim*: similarity matrix, *BT*: the initial binary tree generated by *Cas_support* algorithm
Ensure: *BT*: updated binary tree

```
1: for i = 2 to n do
2:     L ← 0; K ← 0;
3:     for j = i + 1 to n do
4:         if (Sim(S_i, S_j) ≠ 0) then
5:             if (Sim(S_i, S_j) = L) then
6:                 if (S_j.statut = 1) then
7:                     Connect(S_j, S_i, right.side); S_j.statut ← 1;
8:                 end if
9:             else
10:                L ← max(L, Sim(S_i, S_j));
11:                if (L == Sim(S_i, S_j)) then
12:                    if (S_j.statut = 0) then
13:                        if (S_i.statut = 1) then
14:                            if (S_j.left = NULL) then
15:                                Connect(S_i, S_j, left.side); S_i.statut ← 0;
16:                            else
17:                                Connect(S_i, S_j, right.side); S_i.statut = 1;
18:                            end if
19:                        end if
20:                    else
21:                        K ← j;
22:                    end if
23:                end if
24:            end if
25:        end if
26:    end for
27:    if (K ≠ 0) then
28:        Connect(S_K, S_i, left.side); S_K.statut = 0;
29:    end if
30: end for
```


Figure 10. Pseudo code FinalBTTree algorithm

Applying WCLARtAnt on the example given in table 2 provides the binary tree given in figure 9(c). In figure 11, we illustrate the first step that represents the details of the execution of the InitTree algorithm.

Step1 :

i=1	L=0 R=0	
	i=2	Sim (S1, S2) =0 connect (S2, S1, right. side) S2.statut=1
	i=3	Sim (S1, S3) =4 L=max (L, Sim (S1, S3)) = 4 L= Sim (S1, S3) = 4 K=j=3
	i=4	Sim (S1, S4) =2 L= max (L, Sim (S1, S4)) =4 connect (S4, S1, right. side) S4.statut=1
	i=5	Sim (S1, S5) = 0 connect (S5, S1, right. side) S5.statut=1
	i=6	Sim (S1, S6) = 2 L= max (L, Sim (S1, S6)) =2 connect (S6, S1, right. side) S6.statut=1
	i=7	Sim (S1, S7) = 2 L= max (L, Sim (S1, S7)) =4 connect (S7, S1, right. side) S7.statut=1
	i=8	Sim (S1, S8) = 0 connect (S8, S1, right. side) S8.statut=1
	K = 3 connect (S3, S1, left. side)	

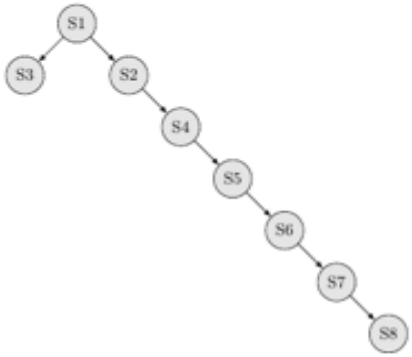


Figure 11. Implementation result of InitTree algorithm.

ISSN 2723-6471

Figure 12, figure 13, figure 14, figure 15, figure 16 and figure 17 show the execution of FinalBTTree of the steps 2, 3, 4, 5, 6 and 7, respectively.

Step 2:

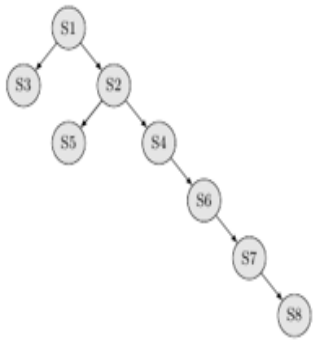
i=2	L=0 R=0	
	i=3 Sim (S2, S3) =0	
	i=4 Sim (S2, S4) =0	
	i=5 Sim (S2, S5) =3 L= max (L, Sim (S2, S5)) =3 L= Sim (S2, S5) = 3 K=j=5	
	i=6 Sim (S2, S6) = 2 L= max (L, Sim (S2, S6)) =3	
	i=7 Sim (S2, S7) = 0	
	i=8 Sim (S2, S8) = 2 L= max (L, Sim (S2, S8)) =3	
	K = 5 connect (S5, S2, left. side)	

Figure 12. Execution of step2 of FinalBTTree

Step 3:

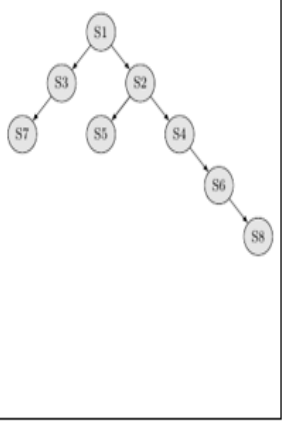
i=3	L=0 R=0	
	i=4 Sim (S3, S4) =2 L= max (L, Sim (S3, S4)) =2 L= Sim (S3, S4) = 2 K=j=4	
	i=5 Sim (S3, S5) =0	
	i=6 Sim (S3, S6) = 2 L= max (L, Sim (S3, S6)) =2	
	i=7 Sim (S3, S7) = 3 L= max (L, Sim (S3, S7)) =3 L= Sim (S3, S7) = 3 K=j=7	
	i=8 Sim (S3, S8) = 0	
	K = 7 connect (S7, S3, left. side)	

Figure 13. Execution of step3 of FinalBTTree

Step 4:

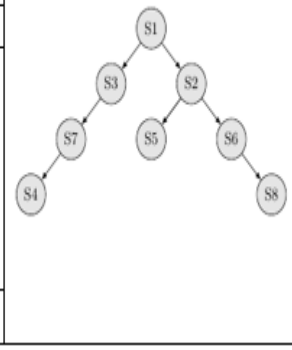
i=4	L=0 R=0	
	i=5 Sim (S4, S5) =0	
	i=6 Sim (S4, S6) = 0	
	i=7 Sim (S4, S7) = 2 L= max (L, Sim (S4, S7)) =2 L= Sim (S4, S7) = 2 S7.statut=0 S4.statut=1 and S4.left=null connect (S4, S7, left. side)	
	i=8 Sim (S4, S8) = 0	

Figure 14. Execution of step4 of FinalBTTree

Step 5:

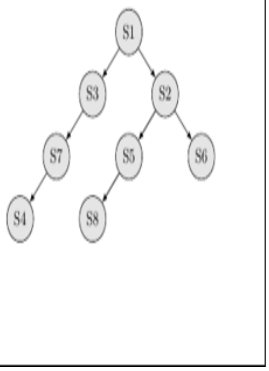
i=5	L=0 R=0	
	i=6 Sim (S5, S6) = 0	
	i=7 Sim (S5, S7) = 0	
	i=8 Sim (S5, S8) = 2 L= max (L, Sim (S5, S8)) =2 L= Sim (S5, S8) = 2 S8.statut=1 K=j=8	
	K = 8 connect (S8, S5, left. side)	

Figure 15. Execution of step5 of FinalBTTree

 ISSN 2723-6471

Step 6:

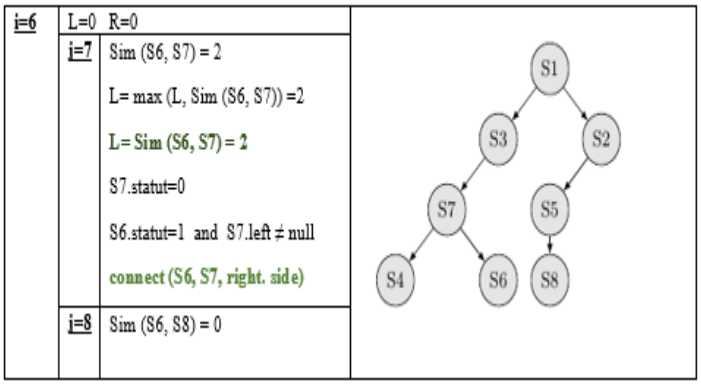


Figure 16. Execution of step6 of FinalBTTree

Step 7:

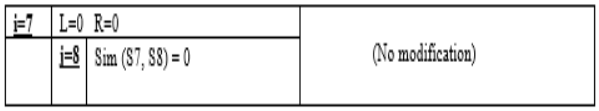


Figure 17. Execution of step7 of FinalBTTree

3.4. Clustering Process

To implement our proposed algorithm WCLARTANT as well as DBSCAN, we opted for datasets coming from NASA Web log files that are extracted from Kennedy Space Center's WEB server in Florida. These log files contain all HTTP requests for a month period, from 1st July, 1995, to 31st July, 1995, for a total of 65,194 entries. As given in figure 19, our adopted clustering process starts with a data preprocessing step (see Section 3.5), and ends with a results evaluation step (see Section 4). For more details, a web log file is a text file (see figure 18) in which each line contains one request and contains the following fields:

Host making the request. A host name, when it is possible, otherwise the Internet address if the name could not be resolved, Timestamp in the format "DAY MON DD HH:MM:SS YYYY", Time zone is -0400, Request given in quotes, HTTP reply code, and Bytes in the reply.

```
199.72.81.55 - - [01/Jul/1995:00:00:01 -0400] "GET /history/apollo/ HTTP/1.0" 200 6245
unicomp6.unicomp.net - - [01/Jul/1995:00:00:06 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
199.120.110.21 - - [01/Jul/1995:00:00:09 -0400] "GET /shuttle/missions/sts-73/mission-sts-73.html HTTP/1.0" 200 4085
burger.letters.com - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/countdown/liftoff.html HTTP/1.0" 304 0
199.120.110.21 - - [01/Jul/1995:00:00:11 -0400] "GET /shuttle/missions/sts-73/sts-73-patch-small.gif HTTP/1.0" 200 4179
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 304 0
burger.letters.com - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/video/livevideo.gif HTTP/1.0" 200 0
205.212.115.106 - - [01/Jul/1995:00:00:12 -0400] "GET /shuttle/countdown/countdown.html HTTP/1.0" 200 3985
d104.aa.net - - [01/Jul/1995:00:00:13 -0400] "GET /shuttle/countdown/ HTTP/1.0" 200 3985
129.94.144.152 - - [01/Jul/1995:00:00:13 -0400] "GET / HTTP/1.0" 200 7074
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /shuttle/countdown/count.gif HTTP/1.0" 200 40310
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786
unicomp6.unicomp.net - - [01/Jul/1995:00:00:14 -0400] "GET /images/KSC-logosmall.gif HTTP/1.0" 200 1204
```

Figure 18. NASA web server Log File

Figure 18 displays the NASA log file example for a site server. Web logs of user's HTTP requests to the NASA Kennedy Space Center World Wide Web server in Florida. It shows the purpose of the demonstration of the web log data format for better understanding of the weblog data set.

In our work, the NASA server log file containing 31 days of data was processed. Table 4 provides a comprehensive overview of the percentage reduction compared to the original size.

Table 4. Results of Preprocessed data

Server log file	NASA July-95
Duration	07/01/95 to 07/31/95
Records in original log file	65194
Records in cleaned log file	12006
Number of sessions	832

3.5. Data Pre-processing

The preprocessing step summarizes the techniques proposed by researchers [37], [38]. It includes two aspects, as given below:

Data cleaning: all log file entries that are not scanned, since they contain image files, sound files, error status codes, etc. Thus, all these elements should be removed.

Sessions Identification: the cleaned log file is further processed for sessions' identification. The threshold time considered as a period is thirty minutes [39]. In addition, the sessions of values less than 4 are not considered, since they do not provide any significant information for the clustering process.

Compared to the original dataset, the pre-processing step provides another new one which includes fewer requests and fewer sessions.

Data cleaning: In this phase, irrelevant or redundant entries are removed. This includes filtering requests for non-HTML files (such as images or audio files), eliminating entries with error codes (e.g., 404, 500), and ensuring that duplicate requests within the same session are not removed. The presence of duplicate requests may indicate repeated or intentional user actions, such as page refreshes or recurring navigation to the same resource.

Missing data handling: Missing data, particularly when certain fields in the logs (such as timestamps or URLs) are incomplete, is addressed during the preprocessing phase. Incomplete entries are eliminated to ensure the integrity of the sessions that are used for clustering.

Noisy entries: To handle noisy data, sessions with fewer than four queries are removed, as they lack sufficient information to contribute meaningfully to the clustering process.

Session Identification: After cleaning the log file, sessions are identified by grouping user interactions that occur within a 30-minute time threshold [40]. This time-based separation ensures that each session accurately represents a distinct period of user activity.

Through these preprocessing steps, we obtained a clean and structured dataset with fewer queries and sessions, which provides a solid basis for clustering. Regarding the duplicates, we have to mention that when grouping sessions from log files, it is generally preferable to keep duplicate data, as they may reflect repeated or intentional user actions, such as refreshing pages or navigating to the same resource repeatedly. These actions can be important for analyzing user behavior. Therefore, this kind of data is often kept to better represent behaviors and not lose information relevant to session analysis. Although the duplicates are removed when they are the result of technical errors or the purpose of the analysis justifies their removal.

3.6. Model Evaluation

Given that the clustering process relies neither on prior knowledge nor on predefined classes, it is essential to evaluate the quality and validity of the results obtained. This evaluation helps to better understand how clusters vary depending on the algorithm used and the nature of the data being grouped.

Various clustering evaluation techniques have been proposed in the literature. For instance, the Davies-Bouldin Index, D-B index, and Dunn Index, such as:

The Davies-Bouldin Index compares the separation and homogeneity of clusters. It can be influenced by clusters of different sizes or of irregular shapes. The D-B index can provide a less accurate picture if clusters are not spherical or uniformly distributed. The Dunn Index focuses on the maximum separation between clusters compared to the minimum

 ISSN 2723-6471

dissimilarity within clusters. However, it can be difficult to compute. It is less intuitive to interpret than the Silhouette index, especially in the case of a large number of clusters or when clusters are very close to each other.

In this study, we have chosen to use silhouette analysis [40], [41], an internal evaluation method [42] particularly effective in estimating the coherence of clusters and determining the optimal number of groups within the data.

The Silhouette Index offers numerous advantages. It evaluates cluster quality by measuring both cohesion, which is the average distance between a point and other points in the same cluster, and separation, which is the average distance between a point and the nearest cluster. This dual assessment allows for straightforward interpretation: a score close to 1 indicates that the points are well clustered, while a score close to -1 suggests potential misclassification of points. Unlike other indices that provide an overall assessment, the Silhouette Index calculates a score for each point individually, allowing for the identification of both the overall clustering quality and individual points that may be misclassified. This granularity is especially useful for refining the clustering results. Moreover, the Silhouette Index is less sensitive to outliers than some other indices, providing a more reliable evaluation of cluster quality even in the presence of noisy data or outliers. It is also flexible, as it can be used with different distance metrics, making it adaptable to various data types and cluster structures. This versatility makes the Silhouette Index applicable across different research contexts without major adaptations. Additionally, the scores can be visualized graphically, allowing for easy comparison of different clusters or clustering configurations. This feature helps in quickly determining the optimal number of clusters by maximizing the average silhouette score across all points.

The Silhouette index S(i) is particularly conceived to consider only the sessions and their similarities. The good classification of each session is checked by computing for each session Si, its value S(i) as follows (see (6)):

$$S(i) = \frac{b(i)-a(i)}{\max\,(b(i)-a(i))} \quad i\epsilon[1,..,n] \tag{6}$$

n: is the sessions' number;

a(i): is the average distance between Si and the other sessions in the corresponding cluster;

b(i): is the average distance between Si and the sessions belonging to the closest cluster.

If S(i) is close to 1, the session is considered as well classified one. On the other hand, else if S(i) is close to −1, that means that Si is badly classified.

 ISSN 2723-6471

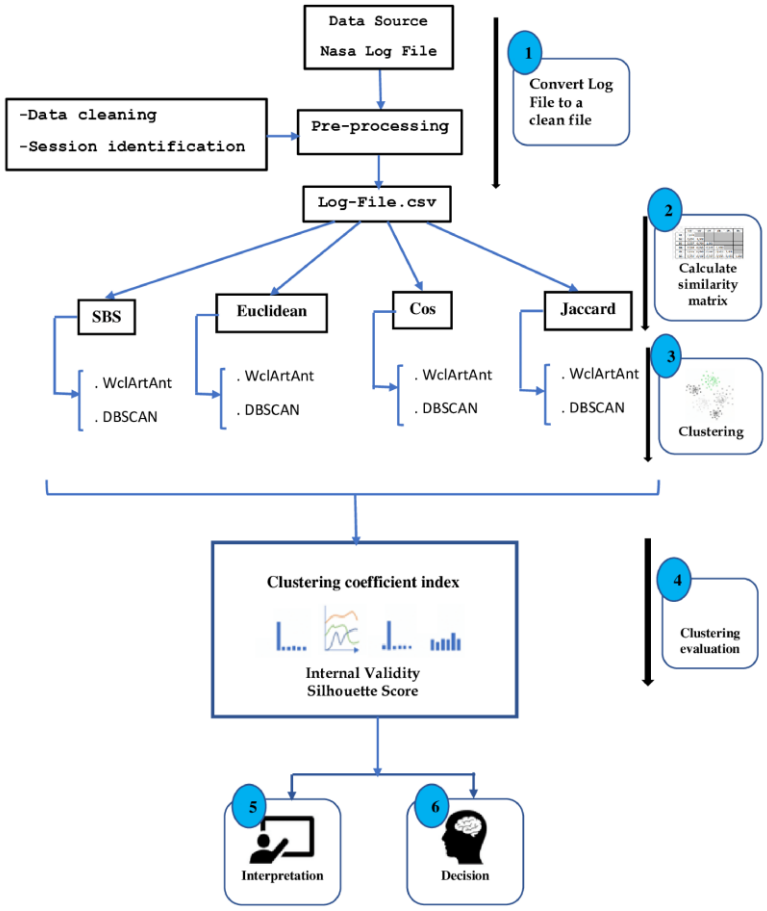


Figure 19. Implementation process.

4. Experiment, Results and Discussion

4.1. Research result

In this section, we present the experimental results of our algorithm WCLArtAnt. The introduced algorithms in the present paper are implemented in Java language, and the programs are run on a personal computer with i3-2328M CPU @ 2.20GHz, and 4 Go, under Windows-7 operating system (64-bit).

In order to identify the similarity measure that best suits the implemented algorithms:

WCLArtAnt and DBSCAN, to cluster WEB sessions of different length. The advantage of these algorithms, is that they do not need input initialization of the number of groups. Thus, we varied the number of sessions from 50 to 832, in order to check out the best combination clustering algorithm/similarity measure algorithm, which provides the best clusters. At this level, we consider the use of silhouette index to measure the quality of each clustering approach.

Table 5. Experimental results of WCLArtAnt, combined with the similarity measures SBS, Euclidean, Jaccard and Cosine.

SeqSz	50	100	200	250	500	832
WCLArtAnt-SBS						
NbClst	14	23	43	59	125	199
Silhouette	0.62	0.44	0.57	0.33	0.32	0.39
WCLArtAnt-Euclidean						

NbClst	16	25	49	60	113	180
Silhouette	0.25	0.23	0.20	0.21	0.19	0.20
			WCLArtAnt-Jaccard			
NbClst	14	32	52	65	120	187
Silhouette	0.28	0.3	0.23	0.24	0.15	0.13
			WCLArtAnt-Cos			
NbClst	16	29	50	63	119	198
Silhouette	0.09	0.11	0.11	0.12	0.12	0.12

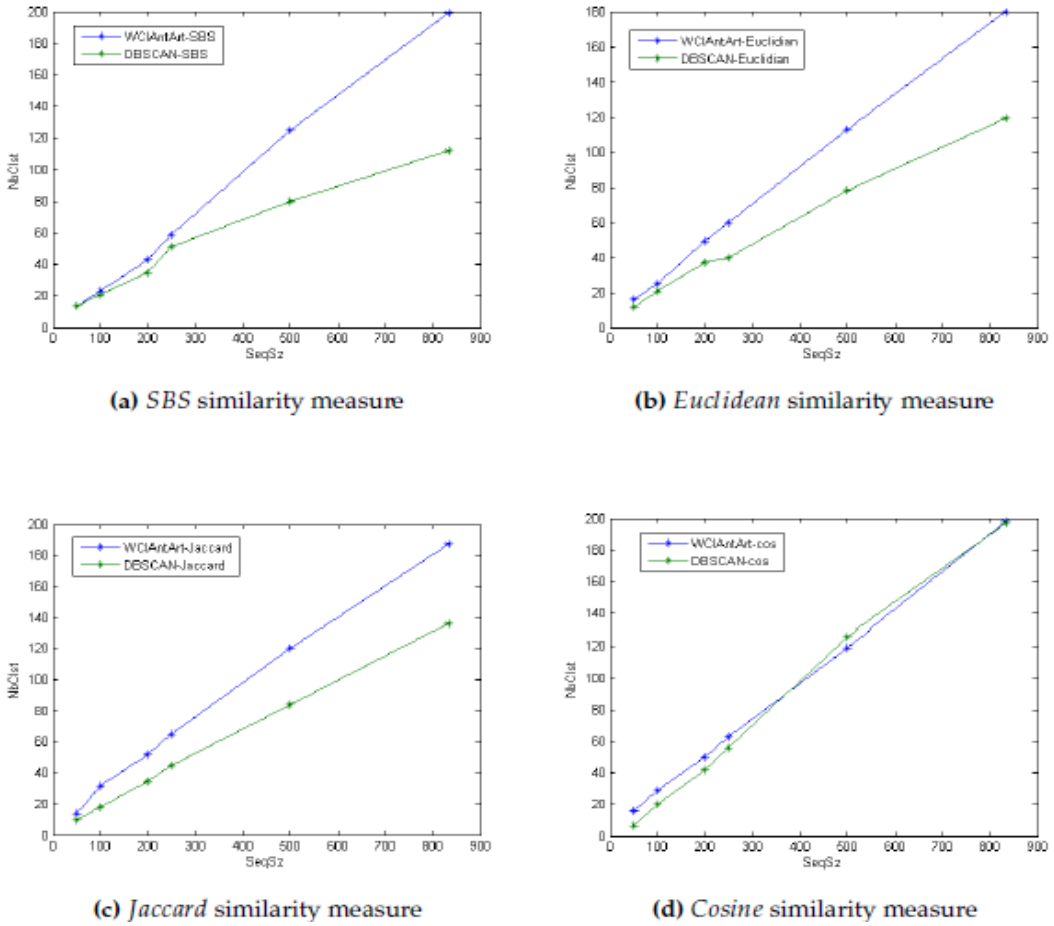
Table 6. Experimental results of DBSCAN applied with the similarity measures SBS, Euclidean, Jaccard and Cosine

Seqsz	50	100	200	250	500	832
	Eps=49	Eps=99	Eps=199	Eps=249	Eps=498	Eps=828
			WCLArtAnt-SBS	(MinPts=2)		
NbClst	14	21	35	51	80	112
Silhouette	0.21	0.16	0.13	0.17	0.13	0.1
			WCLArtAnt-Euclidean	(Eps=1.3, MinPts=2)		
NbClst	12	21	37	40	78	119
Silhouette	0.58	0.69	0.69	0.7	0.68	0.72
			WCLArtAnt-Jaccard	Eps=0.8/0.7/0.6/0.5, MinPts=2		
NbClst	10	18	35	45	84	136
Silhouette	0.09	0.19	0.13	0.1	0.1	0.13
			WCLArtAnt-Cos	Eps=0.6 / 0.3, MinPts=2		
NbClst	16	29	50	63	119	198
Silhouette	0.13	0.22	0.16	0.13	0.07	0.09

Table 5 and table 6 summarize the provided results from the implemented approaches WCLArtAnt and DBScan, respectively. In both tables (table 5 and table 6), we give for each tested log file, the clusters' number NBCLST and the corresponding Silhouette value. As given above, we used different techniques to measure the similarities (SBS, Euclidean, Jaccard and Cosine).

Figure 20 illustrates the comparison between WCLArtAnt and DBScan algorithms using different similarity measures, where X-axis (labeled SeqSz) represents the tackled dataset and Y-axis the clusters' number, labeled NbClst. The corresponding curves to both algorithms show the increase of NBCLST according to the increasing sessions' number SEQSZ.

ISSN 2723-6471



(a) *SBS* similarity measure

(b) *Euclidean* similarity measure

(c) *Jaccard* similarity measure

(d) *Cosine* similarity measure

Figure 20. Comparison of the NbClst between WCLArtAnt and DBScan using different similarity measures (see table 5 and table 6)

In figure 21, the corresponding curves to the Silhouette index obtained through WCLArtAnt and DBScan algorithms using SBS, Euclidean, Jaccard and Cosine similarities, are traced. Thus, X-axis represents SeqSz and Y-axis gives the Silhouette values. We notice a variation of silhouette with the increase in the number of sessions. The sub-figures 21a and 21c, show the corresponding curves to SBS and Jaccard, respectively. According to the silhouette value, the best results are provided by WCLArtAnt compared to the DBSCAN. Through the sub-figures 21b and 21d, we notice the performance of DBSCAN adopted with Euclidean and Cosine. However, we should highlight that WCLArtAnt becomes more efficient, combined with the similarity measure Cosine, where the sessions' number is ≥300.

Consequently, we conclude that our algorithm WCLArtAnt is more efficient, if we adopt SBS or Jaccard as a similarity measure, compared to Euclidean and Cosine. This is because we conceived WCLArtAnt particularly for WEB sessions of different lengths, without taking into consideration the occurrence order of the sessions, which conforms to the basics of SBS and Jaccard.

ISSN 2723-6471

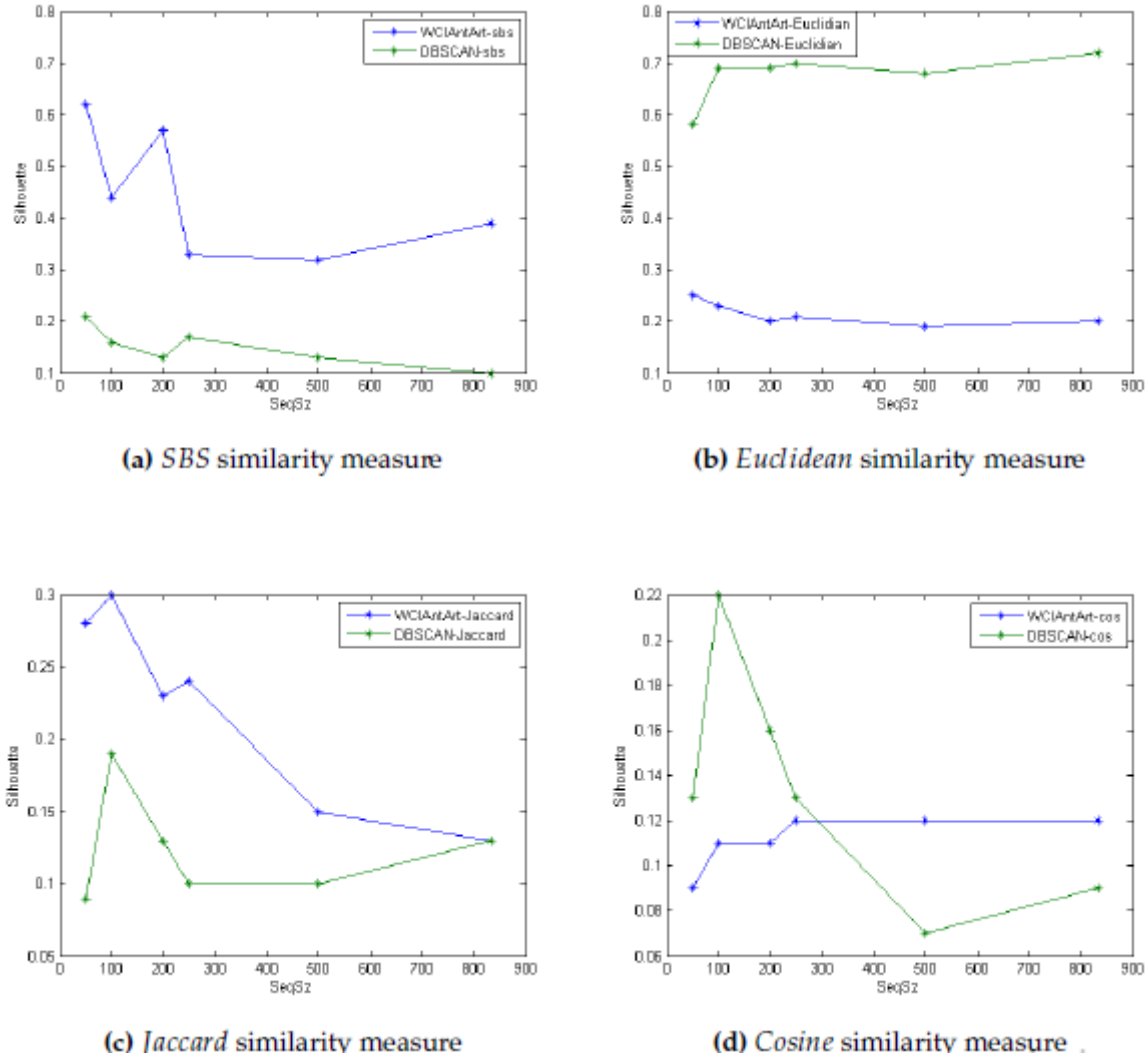


Figure 21. Comparison of the Silhouette index between WCLArtAnt and DBScan using different similarity measures (see table 5 and table 6)

In figure 22, we illustrate the Silhouette value obtained via WCLArtAnt, in terms of the number of sessions SEQSZ. The four curves given in this figure show the clustering quality of our algorithm, using the similarity measures SBS, Euclidean, Jaccard and Cosine. The corresponding curve to SBS, drawn in blue, highlights clearly the efficiency of WCLArtAnt. We notice that the performance varies between 0.62 and 0.39, which is considered as good results, while SBS tackles all the sequence sessions and considers two sequences similar if the inclusion relation is verified between the both.

ISSN 2723-6471

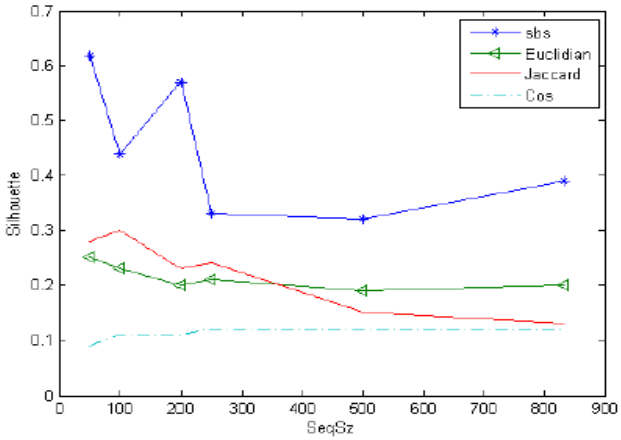


Figure 22. Comparison of Silhouette index between the similarity measures using WCLArtAnt clustering algorithm

4.2. Discussion

This work compared our WCLArtAnt algorithm to the DBSCAN algorithm using several similarity measures (SBS, Jaccard, Euclidean, and Cosine). This comparison aims to identify the most suitable similarity measure for clustering web sessions of different lengths. The graphical result represented in figure 11, shows that the number of clusters increases as the number of sessions grows for both algorithms. This indicates precise segmentation of users, improving the understanding of user behavior and allowing for personalized web content. The result illustrated in figure 20 and figure 21 demonstrates that the WCLArtAnt algorithm is more effective when associated with the SBS or Jaccard similarity measures due to their ability to handle discrete data, accurate evaluation of similar user behaviors, their robustness in handling missing data, enhancing the reliability of the clustering SBS and Jaccard measures is better suited for capturing relationships between sessions without considering the order of appearance.

On the other hand, the effectiveness of Euclidean and Cosine similarity measures is limited in this specific context, even though they are often better at measuring distances between vectors in different fields.

Clustering based on the WCLArtAnt algorithm produces more efficient clusters than DBSCAN. This means that the WCLArtAnt algorithm groups data more densely, which is important for applications such as identifying similar user navigation patterns on a website.

In order to give statistical significance to our results (see table 7), we used Student's t-test [43] to compare the performance of WCLArtAnt and DBSCAN algorithms in terms of silhouette scores, using different similarity measures (SBS, Euclidean, Jaccard and Cosine). These analyses give more clarity on the impact of methodological choices and similarity measures on the quality of clustering in the context of web session analysis.

Table 7. Comparison of Silhouette Scores between WCLArtAnt and DBSCAN

	Score WCLArtAnt	Score DBSCAN	Conf_Interval WCLArtAnt	Conf_Interval DBSCAN	P_values	t-test
SBS	0.45	0.15	[0.313, 0.576]	[0.109, 0.190]	0.000253	5.52
Euclidean	0.19	0.68	[0.189, 0.237]	[0.625, 0.728]	1.4 e-09	-20.96
Jaccard	0.22	0.15	[0.150, 0.294]	[0.085, 0.162]	0.1128	3.10
Cosine	0.11	0.13	[0.099, 0.124]	[0.078, 0.189]	0.3525	-0.97

 ISSN 2723-6471

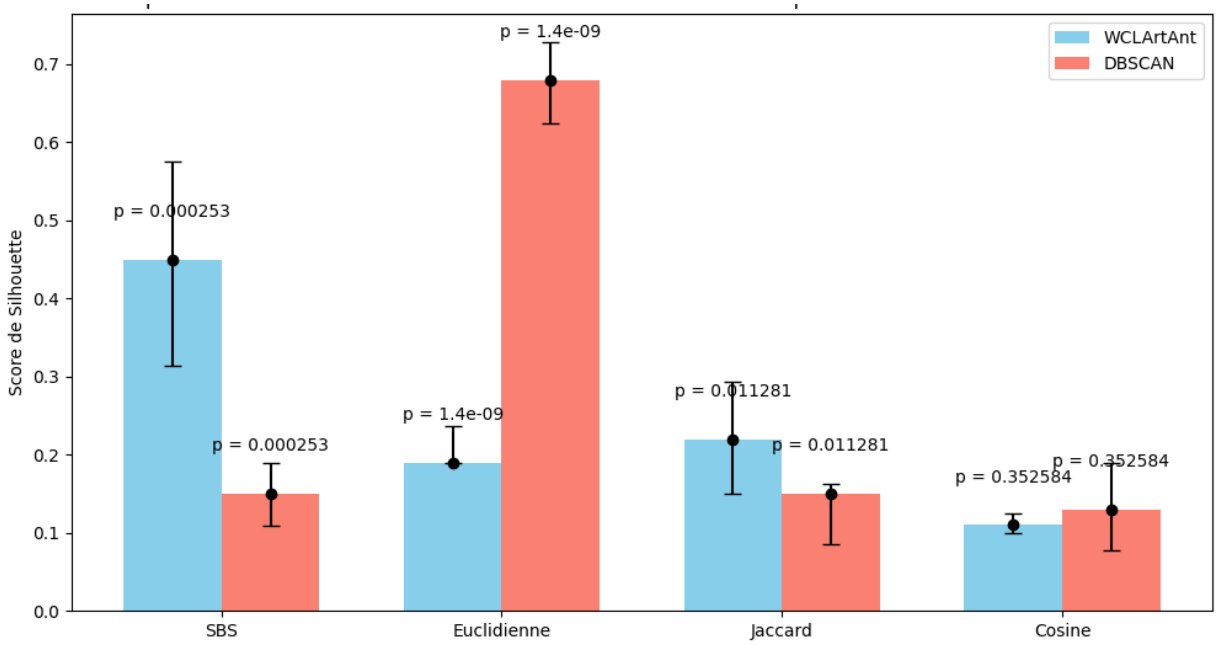


Figure 23. Scores comparison of Silhouette index between WCLArtAnt and DBSCAN for different similarity measures

Figure 23 showsS the average silhouette scores of WCLArtAnt and DBSCAN for each similarity measure (SBS, Euclidean, Jaccard, and Cosine), while including the associated confidence intervals and p-values for each similarity measure. This chat gives a statistical analysis that is conducted to compare the performances of WCLArtAnt and DBSCAN algorithms, combined with different similarity measures, revealing significant differences in their silhouette scores.

We have to specify that during the process of clustering web sessions from log files, the choice of similarity measure is critical to the performance of the algorithm. Factors such as session length (number of pages visited) and recency of visits significantly influence similarity scores. More details about each factor are given as fllows:

Session length: The number of pages visited during a session is a key indicator of user engagement with the site content. Sessions with a high number of page views may reflect deep browsing or a search for specific information. By factoring session length into similarity measures, similar navigation patterns between users can be better identified, which is essential for analyses such as content personalization or site architecture optimization.

Visit recency: The temporality of visits plays a crucial role in session relevance. Recent sessions may carry more weight in the analysis, reflecting current user trends, while older sessions may represent outdated behaviors. Incorporating recency into similarity measures allows us to capture the evolution of user behaviors and adapt strategies accordingly. In our study, we considered the number of pages visited together, recognizing that sessions can vary in length.

However, the order of page visits was not taken into account. Therefore, we chose not to integrate certain factors, such as session length (number of pages visited) and recency of visits, into similarity measures. This decision is based on several considerations: Computational Complexity: Including these factors would have increased the complexity of similarity calculations, making the clustering process more costly in terms of time and resources. Data Variability: Web sessions exhibit high variability in length. Incorporating this variable would have required complex normalizations to ensure a fair comparison between sessions. Study Objective: Although visit recency can provide insights into current user trends, our study aimed to cluster stable browsing behaviors over a given period. Therefore, recency was not considered as a determining factor in our analysis. While these choices simplify the analysis, we recognize that integrating session length and recency could provide a deeper understanding of user behaviors. This perspective will be considered in our future work to enrich the quality of web session clustering.

 ISSN 2723-6471

5. Conclusion and Perspectives

Clustering WEB data is a crucial step in WEB Mining. In this paper, we developed a new clustering algorithm WCLArtAnt founded on research tree basics and the self-assembly of real ants' behavior. We experimented with different approaches on NASA log files, from which the sessions' matrix is deduced to be used as an input of the algorithms. Through the illustrated curves, we succeeded in showing that our algorithm WCLArtAnt is more efficient when the similarity measure SBS is adopted. In addition, to measure the quality of the provided results, we computed Silhouette index of each approach, which high-lighted the promising aspect of WCLArtAnt results, especially when we use the similarity measure SBS. This means that the WCLArtAnt algorithm allows for dense data grouping, using similarity measures (SBS and Jaccard) that are suitable for capturing relationships between sessions without considering the order of appearance, which is important for applications like identifying similar user navigation on a website. On the other side, the DBSCAN algorithm achieves better clustering results by adopting the Euclidean similarity measure, which is the default distance measure used by DBSCAN. The strong aspect of our algorithm is the fact that we tackle the clustering problem without prior knowledge about the clusters' number. To improve our algorithm's efficiency, a meticulous choice of the similarity measure technique is required.

As perspective, it is important to emphasize that the clustering result of the WCLArtAnt algorithm can be integrated into clustering-based recommendation systems. In this case, each cluster allows the identification of the user behavior and better understanding of their needs. Once the clusters are formed, the most popular or representative sessions of each cluster are recommended to the members of this cluster.

This technique will improve the performance and quality of recommendations and provide better personalization of recommendations. For instance, in the context of an academic website. Since each session is represented by the pages visited by each user, the recommendation technique provides to users the suitable events, training, conference pages, or workshops.

The obtained results show that the WCLArtAnt algorithm, when using the SBS similarity measure, produces better clustering results compared to classical measures such as Jaccard and Cosine, as well as to the DBSCAN algorithm, whether with a small or a large amount of data. This can be explained by the ability of the SBS measure to efficiently capture complex relationships between web sessions, even in sparse or non-linear data environments. On the other hand, it is interesting to note that the Euclidean measure, when used with DBSCAN, generates better clustering results on a large volume of data. This is probably due to the reason that DBSCAN relies on a notion of local density, and the Euclidean measure is particularly suited to detecting well-separated clusters in a continuous space. With a large amount of data, this combination excels in identifying global clusters with clear boundaries. These observations highlight the importance of choosing the similarity measure based on the characteristics of the data.

In our future work, we plan to focus on visualizing the session clusters generated by WCLARTANT, so that they can be exploited as input data for various WEB applications, such as web personalization, anomaly detection, and web page recommendation. Furthermore, we intend to apply our algorithm in the field of web content mining and text clustering, integrating natural language processing (NLP) methods, so that it is suitable for the automatic identification of groups of web pages sharing similar content characteristics, to be used in the categorization of web pages and web query categorization.

More precisely, the integration of NLP techniques into our WCLArtAnt algorithm will start with the use of text data as input to the clustering process. To make this data usable and usable by WCLArtAnt, we will represent it as numerical vectors. To this end, we plan to apply adapted NLP techniques, such as tokenization, lemmatization, and vectorization. Among the vectorization methods considered, the TF-IDF model is particularly suited to capture the lexical specificities of texts. These representations will enrich the algorithm's ability to identify similarities based on the textual content of web sessions, making clustering more precise and relevant.

6. Declarations

6.1. Author Contributions

Conceptualization: S.M., K.B., and F.Z.L.; Methodology: S.M.; Software: K.B.; Validation: S.M., K.B., and F.Z.L.; Formal Analysis: S.M., K.B., and F.Z.L.; Investigation: K.B.; Resources: S.M.; Data Curation: S.M.; Writing Original Draft Preparation: S.M., K.B., and F.Z.L.; WritingReview and Editing: S.M., K.B., and F.Z.L.; Visualization: K.B. All authors have read and agreed to the published version of the manuscript.

6.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

6.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

6.4. Institutional Review Board Statement

Not applicable.

6.5. Informed Consent Statement

Not applicable.

6.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] M.-L. Pérez-Delgado, "Revisiting the iterative ant-tree for color quantization algorithm," *J. Vis. Commun. Image Represent.*, vol. 78, no. July, pp. 17, 2021, doi: 10.1016/j.jvcir.2021.103180.

[2] J. Nayak, K. Vakula, P. Dinesh, B. Naik, and M. Mishra, "Ant colony optimization in data mining: critical perspective from 2015 to 2020," in *Innovation in Electrical Power Engineering, Communication, and Computing Technology: Proceedings of IEPCCT 2019*, Springer, 2020, vol. 630, no. February, pp. 361-374, doi: 10.1007/978-981-15-2305-2_29.

[3] M.-L. Pérez-Delgado, "Color quantization with particle swarm optimization and artificial ants," *Soft Comput.*, vol. 24, no. 6, pp. 4545-4573, 2020.

[4] V.-T. Luu, G. Forestier, J. Weber, P. Bourgeois, F. Djelil, and P.-A. Muller, "A review of alignment-based similarity measures for web usage mining," *Artif. Intell. Rev.*, vol. 53, no. 3, pp. 1529-1551, 2020, doi: 10.1007/s10462-019-09712-9.

[5] H. Singh and P. Kaur, "An Effective Clustering-Based Web Page Recommendation Framework for E-Commerce Websites," *SN Comput. Sci.*, vol. 2, no. 4, pp. 339, June 2021, doi: 10.1007/s42979-021-00736-z.

[6] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, no. January, pp. 10031-10061, 2022, doi: 10.1109/ACCESS.2022.3142859.

[7] S. Kumar and S. M. Verma, "Multi-criteria-Based Page Ranking Using Metaheuristic Swarm Optimization," in *Advanced Computational and Communication Paradigms*, Springer Nature Singapore, vol. 2023, no. 1, pp. 31-41, doi: 10.1007/978-981-99-4284-8_3.

[8] E. Twumasi, E. A. Frimpong, N. K. Prah, and D. B. Gyasi, "A novel improvement of particle swarm optimization using an improved velocity update function based on local best murmuration particle," *J. Electr. Syst. Inf. Technol.*, vol. 11, no. 1, pp. 42-54, Oct. 2024, doi: 10.1186/s43067-024-00168-8.

[9] R. Mateless, H. Zlatokrilov, L. Orevi, M. Segal, and R. Moskovitch, "IPvest: Clustering the IP Traffic of Network Entities Hidden Behind a Single IP Address Using Machine Learning," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3647-3661, 2021, doi: 10.1109/TNSM.2021.3062488.

[10] K. Tabianan, S. Velu, and V. Ravi, "K-Means Clustering Approach for Intelligent Customer Segmentation Using Customer Purchase Behavior Data," *Sustainability*, vol. 14, no. 12, pp. 15, 2022, doi: 10.3390/su14127243.

[11] T. N. Win and N. K. Z. Lwin, "Analysis of Customers Interest for Web Log Clustering," in *2024 IEEE Conference on Computer Applications (ICCA)*, 2024, vol. 2024, no. May, pp. 1-6, doi: 10.1109/ICCA62361.2024.10533033.

[12] A. Safdari-Vaighani, P. Salehpour, and M.-R. Feizi-Derakhshi, “Detecting Non-Spherical Clusters Using Modified CURE Algorithm,” in *2021 11th International Conference on Computer Engineering and Knowledge (ICCKE)*, 2021, vol. 2021, no. February, pp. 369-373, 2021, doi: 10.1109/ICCKE54056.2021.9721508.

[13] T. Fan, N. Guo, and Y. Ren, “Consumer clusters detection with geo-tagged social network data using DBSCAN algorithm: a case study of the Pearl River Delta in China,” *GeoJournal*, vol. 86, no. 1, pp. 317-337, Feb. 2021, doi: 10.1007/s10708-019-10072-8.

[14] Z.-Y. Lim, L.-Y. Ong, M.-C. Leow, T.-W. Lee, and Q.-M. Tay, “Understanding User Behaviour with Web Session Clustering and User Engagement Metrics,” in *2023 19th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, 2023, vol. 2023, no. April, pp. 19-24, doi: 10.1109/CSPA57446.2023.10087488.

[15] Y. Qawqzeh, M. T. Alharbi, A. Jaradat, and K. N. A. Sattar, “A review of swarm intelligence algorithms deployment for scheduling and optimization in cloud computing environments,” *PeerJ Comput. Sci.*, vol. 7, no. August, pp. 17, 2021, doi: 10.7717/peerj-cs.696.

[16] J. Tang, G. Liu, and Q. Pan, “A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends,” *IEEE CAA J. Autom. Sin.*, vol. 8, no. 10, pp. 1627-1643, 2021, doi: 10.1109/JAS.2021.1004129.

[17] A. Lakhmani, R. K. Thulasiram, and P. Thulasiraman, “Nature-Inspired Portfolio Diversification Using Ant Brood Clustering,” in *Applications of Evolutionary Computation*, Springer Nature Switzerland, 2024, vol. 14634, no. March, pp. 115-130, doi: 10.1007/978-3-031-56852-7_8.

[18] D. Minarolli, “Distributed Task Allocation in Network of Agents Based on Ant Colony Foraging Behavior,” in *Proceedings of the 24th International Conference on Computer Systems and Technologies (CompSysTech ’23)*, ACM, 2023, vol. 2023, no. September, pp. 59-64, doi: 10.1145/3606305.3606324.

[19] S. Bouarourou, A. Boulaalam, et al., “A bio-inspired adaptive model for search and selection in the Internet of Things environment,” *PeerJ Comput. Sci.*, vol. 7, no. December, pp. e762, 2021, doi: 10.7717/peerj-cs.762.

[20] Q. He, J. Mou, and B. Lin, “A Robust Self-Organizing Tree-Based Routing Protocol for Wireless Sensor Networks,” *Math. Probl. Eng.*, vol. 2021, no. 1, pp. 13, 2021, doi: 10.1155/2021/5932347.

[21] S. Moufok, A. Mouattah, and K. Hachemi, "K-means and DBSCAN for look-alike sound-alike medicines issue," *Int. J. Data Min. Model. Manag.*, vol. 16, no. 1, pp. 49-65, 2024, doi: 10.1504/IJDMMM.2024.136215.

[22] N. M. Nhat, "Applied Density-Based Clustering Techniques for Classifying High-Risk Customers: A Case Study of Commercial Banks in Vietnam," *J. Appl. Data Sci.*, vol. 5, no. 4, pp. 1639-1653, 2024, doi: 10.47738/jads.v5i4.344.

[23] A. S. Paramita and T. Hariguna, "Comparison of K-Means and DBSCAN Algorithms for Customer Segmentation in E-commerce," *J. Digit. Mark. Digit. Curr.*, vol. 1, no. 1, pp. 43-62, 2024, doi: 10.47738/jdmdc.v1i1.3.

[24] L. Benova and L. Hudec, "Comprehensive Analysis and Evaluation of Anomalous User Activity in Web Server Logs," *Sensors*, vol. 24, no. 3, pp. 23, 2024, doi: 10.3390/s24030746.

[25] A. Fawzia Omer et al., "Big Data Mining Using K-Means and DBSCAN Clustering Techniques," in *Big Data Analytics and Computational Intelligence for Cybersecurity*, vol. 2022, no. 1, pp. 231-246, 2022, doi: 10.1007/978-3-031-05752-6_15.

[26] R. Bateja, S. K. Dubey, and A. Bhatt, "Evaluation and Application of Clustering Algorithms in Healthcare Domain Using Cloud Services," in *Sustainable Technologies for Computational Intelligence*, vol. 2022, no. 1, pp. 249-261, 2022, doi: 10.1007/978-981-16-4641-6_21.

[27] V. Zabiniako et al., "Analysis of Algorithms for Detecting Users’ Behavioral Models based on Sessions Data," *Complex Syst. Inform. Model. Q.*, vol. 2024, no. 41, pp. 55-79, 2024, doi: 10.7250/csimq.2024-41.04.

[28] R. Geetharamani and P. Revathy, "Grouping Users Through Pair Wise Sequence Alignment and Graph Traversal Based on Web Page Navigation Behaviour," in *ICDSMLA 2019*, vol. 2020, no. 1, pp. 1770-1791, 2020, doi: 10.1007/978-981-15-1420-3_182.

[29] D. Ai et al., "Identifying local associations in biological time series: algorithms, statistical significance, and applications," *Brief. Bioinform.*, vol. 24, no. 6, pp. 13, Nov. 2023, doi: 10.1093/bib/bbad390.

[30] R. Geetharamani and P. Revathy, "Web User Grouping Based on Navigation Patterns Through Pair Wise Sequence Alignment and Breadth First Search," in *ICDSMLA 2019*, vol. 2020, no. 1, pp. 1743-1758, doi: 10.1007/978-981-15-1420-3_180.

[31] M. Abbasi and A. Shokrollahi, "Enhancing the performance of decision tree-based packet classification algorithms using CPU cluster," *Clust. Comput.*, vol. 23, no. 4, pp. 3203-3219, Dec. 2020, doi: 10.1007/s10586-020-03081-7.

[32] A. Rashelbach, O. Rottenstreich, and M. Silberstein, "A Computational Approach to Packet Classification," *SIGCOMM '20*, vol. 2020, no. 1, pp. 542-556, doi: 10.1145/3387514.3405886.

[33] B. Merikhi and M. R. Soleymani, "Automatic Data Clustering Framework Using Nature-Inspired Binary Optimization Algorithms," *IEEE Access*, vol. 9, no. 1, pp. 93703-93722, 2021, doi: 10.1109/ACCESS.2021.3091397.

[34] A. Law and A. Ghosh, "Multi-Label Classification Using Binary Tree of Classifiers," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 6, no. 3, pp. 677-689, 2022, doi: 10.1109/TETCI.2021.3075717.

[35] P. Bose et al., "Competitive Online Search Trees on Trees," *ACM Trans Algorithms*, vol. 19, no. 3, pp. 20-32, June 2023, doi: 10.1145/3595180.

[36] V. Sonai et al., "CTLA: Compressed Table Look up Algorithm for Open Flow Switch," *IEEE Open J. Comput. Soc.*, vol. 5, no. 1, pp. 73-82, 2024, doi: 10.1109/OJCS.2024.3361710.

[37] P. Svec et al., "Web Usage Mining: Data Pre-processing Impact on Found Knowledge in Predictive Modelling," *Procedia Comput. Sci.*, vol. 171, no. 2, pp. 168-178, 2020, doi: 10.1016/j.procs.2020.04.018.

[38] P. Verma and N. Kesswani, "FEDUS: A comprehensive algorithm for web usage mining," *J. Inf. Optim. Sci.*, vol. 41, no. 3, pp. 835-854, 2020, doi: 10.1080/02522667.2019.1616912.

[39] M. Srivastava et al., "Performance evaluation of the mapreduce-based parallel data preprocessing algorithm in web usage mining with robot detection approaches," *IETE Tech. Rev.*, vol. 39, no. 4, pp. 865-879, 2022, doi: 10.1080/02564602.2021.1918584.

[40] M. Shutaywi and N. N. Kachouie, "Silhouette Analysis for Performance Evaluation in Machine Learning with Applications to Clustering," *Entropy*, vol. 23, no. 6, 2021, doi: 10.3390/e23060759.

[41] A. Dudek, "Silhouette Index as Clustering Evaluation Tool," in *Classification and Data Analysis*, vol. 2020, no. 1, pp. 19-33, doi: 10.1007/978-3-030-52348-0_2.

[42] F. Malik et al., "A Novel Hybrid Clustering Approach Based on Black Hole Algorithm for Document Clustering," *IEEE Access*, vol. 10, no. 1, pp. 97310-97326, 2022, doi: 10.1109/ACCESS.2022.3202017.

[43] P. Maciej and M. Tadeusz, "A study on using data clustering for feature extraction to improve the quality of classification," *Knowl. Inf. Syst.*, vol. 63, no. 7, pp. 1771-1805, 2021.