



Using Trace Clustering to Group Learning Scenarios: An Adaptation of FSS-Encoding to Moodle Logs Use Case

Noura Joudieh, Marwa Trabelsi, Ronan Champagnat, Mourad Rabah, Nikleia Eteokleous

► To cite this version:

Noura Joudieh, Marwa Trabelsi, Ronan Champagnat, Mourad Rabah, Nikleia Eteokleous. Using Trace Clustering to Group Learning Scenarios: An Adaptation of FSS-Encoding to Moodle Logs Use Case. 16th International Conference on Computer Supported Education, May 2024, Angers, France. pp.247-254, 10.5220/0012636400003693 . hal-04635560

HAL Id: hal-04635560

<https://hal.science/hal-04635560v1>

Submitted on 6 Apr 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Using Trace Clustering to Group Learning Scenarios: An Adaptation of FSS-Encoding to Moodle Logs Use Case

Noura Joudieh¹ ^a, Marwa Trabelsi¹ ^b, Ronan Champagnat¹ ^c, Mourad Rabah¹ ^d
and Nikleia Eteokleous² ^e

¹*L3i Laboratory, La Rochelle University, La Rochelle, France*

²*Frederick University, Cyprus*

Keywords: Learning Process, Trace Clustering, Process Mining, Learning Scenarios, Learning Paths, Quality of Education.

Abstract: Learners adopt various learning patterns and behaviors while learning, rendering their experience a valuable asset for recommending learning paths for other learners. Process Mining is useful in this case to discover models that reveal learners' taken learning paths in an educational platform. Nonetheless, due to the heterogeneity of behavior and the volume of data, trace clustering is crucial to reveal various groups of learners and discover relevant process models rather than 'spaghetti' ones. In this paper, we address the limits of and improve on a feature-based trace clustering approach known as FSS-encoding, ideal for unstructured processes to extract diverse learning patterns adopted by students, to be later employed in a learning path recommendation. Our enhancements include a refined pattern selection, preserving the uniqueness of less frequent events and increasing the overall effectiveness of the trace clustering process. Our method was applied to Moodle logs acquired from 2018 to 2022, comprising 471 students in the Computer Science and Engineering Department of Frederick University in Cyprus. The results show three clusters with a 25% improvement in silhouette coefficient. Their consequent discovered process models depict the various learning scenarios adopted, including activities like studying, solving exercises, undergoing assessments, applying, and others.

1 INTRODUCTION

Learning is no longer confined to classrooms or conventional teacher-student scenarios, thanks to advancements in technology that improved the educational sector. On the other hand, e-learning is becoming more prevalent, as learning material and resources are accessible to everyone, at any time, and anywhere. However, this accessibility comes at a cost: learners may become overwhelmed when attempting to meet a learning goal, which could ultimately result in a drop in motivation and learning efficiency. In this direction, Recommender Systems (RS) (Aggarwal, 2016) in e-learning aim to personalize the learning experience by intelligently filtering online content based on individual learner preferences, actions, and needs, de-

viating from one-size-fits-all models. On the other hand, users in information systems leave their traces recorded by logging system in the form of event logs. Process Mining (PM), a discipline that combines data mining, machine learning and process modeling uses these event logs to discover process models that reveal the behavior followed by users in a system (Van der Aalst, 2016). In educational platforms and systems, the former has paved the way for a rich line of research to discover the behaviors of students while performing different learning activities like attending a course, or taking a quiz (Cenka and Anggun, 2022).

In a previous work (Joudieh et al., 2023), we have proposed a framework to recommend a personalized adaptive learning path for a learner seeking a learning objective, while employing past learning experience extracted via Process Mining. The framework constitutes 3 main modules: Recommender Module, Data Module and Process Mining Module. The PM module is the focal point of this article. The former is in charge of using event logs from the Moodle learning management system to discover a model that

^a  <https://orcid.org/0000-0003-3142-2962>

^b  <https://orcid.org/0009-0001-2040-063X>

^c  <https://orcid.org/0000-0001-5256-5706>

^d  <https://orcid.org/0000-0001-8136-5949>

^e  <https://orcid.org/0000-0003-4364-9558>

shows the learning paths taken by students while taking courses. It enables the generation of insights into students' learning journeys, forming the foundation for personalized and effective learning path recommendations. However, a huge amount of logs might generate incomprehensible "spaghetti" models, thus the need of trace clustering (Song et al., 2008) as a primary step to capture the different learning patterns and discover more understandable models.

Driven by this objective, we adopt a recent trace clustering approach (Trabelsi et al., 2021) based on encoding the traces in terms of the frequent subsequent patterns within, called **FSS Encoding**. Originally applied to digital library users, this method was able to discover three different profiles of users. In this paper, we address the limitations of this method and propose an improved version that enhances the clustering results and discovered models. The improved approach is applied on collected Moodle Logs of 471 students taking courses in the Department of Computer Science and Engineering in Frederick University in Cyprus for the period of 2018-2022.

The paper provides an overview of trace clustering approaches in Section 2, followed by an in-depth discussion of FSS encoding, its limitations, and proposed improvements in Section 3. The enhanced FSS approach is demonstrated using Moodle logs, with data collection and treatment detailed in Section 4 and results presented in Section 5. The paper concludes in Section 6 with discussions on future perspectives.

2 RELATED WORKS

Process Mining is a field of science bridging the gap between data-oriented analysis and process-oriented analysis, which aims to extract knowledge from event logs. The application of process mining techniques extends across various domains, including hospitals, banks and municipalities (Van der Aalst, 2016; Lu et al., 2019; Trabelsi et al., 2021).

Process mining techniques utilize event logs as input to generate, enhance, or validate process models. An event log comprises a set of execution traces, each representing a specific process instance. For example, the workflow of a student on an e-learning platform. Starting with the action of viewing a course (Course viewed), the student may navigate to explore specific elements within the course (Course module viewed). These elements could include lectures, videos, assignments, or quizzes. Upon selecting a quiz element, the student proceeds to submit his/her answers (Quiz submitted). Each of these activities constitutes a unique trace within the main process.

Table 1: Sample of students event logs.

CaseId	Timestamp	Activity label
1	2018-01-12T10:34:25	Course viewed
2	2018-01-12T10:36:25	Course viewed
1	2018-01-12T10:34:26	Course module viewed
1	2016-01-12T10:34:28	Submission viewed
3	2018-01-12T10:36:26	Course viewed
3	2018-01-12T10:36:27	Submission form viewed

For instance, in the Moodle platform, every student can be considered as a case following a learning process. The series of events associated with a specific case is referred to as a trace. Each row in Table 1 represents an executed event, including details such as the event identifier (CaseId), the activity label, the timestamp (day, hour, minute, and second), and additional attributes relevant to the event. Formally, an **event log** $L = \{t_1, t_2, \dots, t_k\}$ is a set of k **traces** where each trace t_i ($1 \leq i \leq k$) is a set of n_i consecutive events $t_i = \langle e_{i1}, e_{i2}, \dots, e_{in_i} \rangle$ made by the same CaseId.

Many process discovery methods have been proposed in the literature in order to automatically generate process models. Process discovery algorithms aim to discover process models based on event logs. The discovered process models are intended to represent the whole event logs. Several models can be used for this purpose such as Petri nets or Fuzzy models (Van der Aalst, 2016).

Nevertheless, numerous process mining studies have demonstrated that creating a single process model for an entire log is not ideal, particularly for very large datasets containing unstructured processes. An unstructured process is one that is driven by a user rather than a software. It results in many possible paths, but only few are relevant. Process mining techniques often result in complex and/or overfitted models, such as the widely recognized spaghetti model or flower model (Van der Aalst, 2016). To overcome these issues, existing works proposed trace clustering methods prior to modeling (Diamantini et al., 2016).

The literature offers numerous approaches to trace clustering, which can be categorized into three types of clustering techniques based on how the traces are presented before clustering (Song et al., 2008; Zandkarimi et al., 2020). Additionally, there is a hybrid-based clustering category that integrates various techniques from the aforementioned methods (De Koninck and De Weerd, 2019).

The first category, referred to as **trace-based clustering**, groups traces based on syntax similarity, as explained in (Bose and Van der Aalst, 2009) and (Chatain et al., 2017). This approach draws inspiration to the Levenshtein distance metric, which measures the dissimilarity between two strings. In this context, a trace can be transformed into another

through edit operations like substitution, addition, or removal of events. The edit distance between two traces is then calculated as the minimum number of these edit operations required to convert one trace into the other. A lower edit distance indicates a higher level of similarity between the traces. Following this, distance-based clustering algorithms are applied to group the traces into distinct clusters. In the field of education, both (Laksitowening et al., 2023) and (Zhang et al., 2022) focus on students' logs to capture various characteristics and learning patterns. They both employ hierarchical clustering as a clustering algorithm to group students' traces.

The second category is the **model-based clustering**. It assumes that accurate models are discovered from homogeneous sub-logs (Cadez et al., 2003; Ferreira et al., 2007). The process model is considered as input for the clustering in order to structure traces. These traces are used back to mine process models. The obtained clusters strongly depend on the quality metrics used for evaluating the accuracy of discovered process models (De Weerd et al., 2013).

The third category, called **feature-based clustering**, involves the conversion of each trace into a vector of features based on predefined characteristics. The similarity between two traces is then determined by the similarity between their corresponding vectors. Existing methods within this category often rely on metrics such as the frequency of events or the frequency of direct succession relations between events to transform traces into vectors (Song et al., 2008). For instance, (Song et al., 2008) analyzed traces from healthcare information systems, converting them into features such as the count of individual event occurrences or the count of pairs of events in immediate succession. (Bose and van der Aalst, 2009) employed a similar technique on longer sub-parts of traces, evaluating the occurrence of more complex motifs like repeats, defined as n-grams observed at different points in the trace. Subsequently, distance-based clustering algorithms are applied to group the traces into distinct clusters (Zandkarimi et al., 2020).

Our work in this article aligns with the feature-based approach, as we enhance a feature-based method called **FSS encoding** (Trabelsi et al., 2021). The foundational FSS encoding method originated in the context of digital libraries. Its primary concept involves extracting features from traces through the identification of frequent sub-sequences and their corresponding encoding. The encoding of frequent sub-sequences considers multiple parameters to effectively distinguish traces of digital libraries users. A clustering algorithm was then applied on the converted traces to finally assign each trace to the appro-

priate cluster. In the case of digital libraries, FSS encoding results showed the effectiveness of the method to cluster users and to model their journeys. These results were validated on a real-world data provided by the national library of France¹.

3 [F]REQUENT [S]UB-[S]EQUENCES ENCODING OF LEARNERS INTERACTIONS

As mentioned in Section 1, we extended the FSS encoding method (henceforth baseline) proposed by (Trabelsi et al., 2021). The improvement preserves more information about the traces and the uniqueness of each trace, and allows better patterns to be discovered from the traces.

3.1 Baseline

In this section, we briefly describe the baseline, the FSS encoding proposed in (Trabelsi et al., 2021). The fundamental strategy of this method operates under the assumption that a trace containing a frequent sub-sequence (FSS) is the most significant trace (Lu et al., 2019). This strategy involves grouping traces based on frequent sub-sequences. An FSS, denoted as $\langle e_1, \dots, e_n \rangle$, comprises a finite set of events of length n ($n > 1$), where the events are executed in order at least two times.

Traces are converted using a specific **FSS encoding**. In this encoding, each identified FSS in a trace is replaced by its corresponding encoding. Events that don't belong to any FSS are deemed irrelevant, and only their positions contribute to the clustering. Thus, such events in the traces are replaced by 1.

The baseline itself aims to effectively distinguish traces within different clusters by considering factors such as the [1-length] and [2-frequency] of the FSS, the [3-frequency of events] within the FSS, and the [4-frequency of direct succession relations between events] in the FSS. This encoding strategy enhances the vector representation of traces, emphasizing the significance of longer FSSs, higher frequencies, and the occurrence of specific events and relations within the sequences.

All the extracted FSS are encoded in each trace as follows:

¹Bibliothèque Nationale de France: <https://www.bnf.fr/>

$$Encoding(FSS) = \frac{1}{f_{FSS} \sum_{i=1}^{n-1} f_{e_i} f_{e_{i+1}} f_{r_{i,i+1}}} \quad (1)$$

Where, f_{FSS} is the frequency of the extracted FSS, n is its length (number of events), f_{e_i} is the frequency of the event e_i in the event logs and $f_{r_{i,i+1}}$ is the frequency of the direct relation between all consecutive events of the FSS in the event logs. The resulting encoding value falls between 0 and 1, where a value closer to 0 signifies the significance of the FSS within the entire event logs.

This encoding method allows the distinction between traces that share the same FSS but not in the same position. Furthermore, by replacing all events not part of an FSS with 1, the information about the position of the FSS was retained, the gaps between different FSS, as well as the size of the trace. Following the FSS encoding, all the traces in the event logs are converted into sparse vectors. These vectors are then clustered based on the similarity between them.

3.2 Our Approach

While the baseline approach replaces all events not part of a Frequent Sub-sequence with 1, it leads to a loss of information regarding the uniqueness of individual activities that do not participate in a pattern. These activities might carry significance even though they occur less frequently. The baseline's simplification may overlook the diversity and importance of such singular activities, potentially impacting the comprehensiveness of the analysis. For instance, two traces, $t_1 = \langle e_1, e_2, e_3, e_4, e_5 \rangle$ and $t_2 = \langle e_0, e_1, e_2, e_3 \rangle$, are converted into vectors $[E_{FSS_1}, 1, 1]$ and $[1, E_{FSS_1}]$, respectively, where E_{FSS_1} represents the encoding of $\langle e_1, e_2, e_3 \rangle$. Additionally, the events e_0 , e_4 and e_5 are not considered in the clustering.

On the other hand, our proposed FSS encoding method takes into account the frequency and relationships of both the FSS and individual events, offering a more nuanced representation that preserves the distinctive characteristics of each activity. For example in Moodle, a learner is more likely to view a course several times before taking a quiz once. Thus, it is important to preserve the position and identity of these less frequent activities as they might hold significant information to understand the undertaken learning process. Driven by this, using our approach traces t_1 and t_2 in the previous example, are converted respectively into vectors $[E_{FSS_1}, f(e_4), f(e_5)]$ and $[f(e_0), E_{FSS_1}]$. $f(a)$ represents the frequency of occurrence of activity a in all the traces, thus preserving its identity in means of its frequency that eventually reflects its significance.

Algorithm 1 depicts our proposed approach. As outlined, the initial step involves the transformation of the original event logs R (refer to Table 1) into a set of traces L . This set organizes the sequence of events chronologically based on the unique identifier *CaseId*. For example, based on the Table 1, the corresponding trace of CaseId 1 is $\langle \text{Course viewed}, \text{Course module viewed}, \text{Submission viewed} \rangle$.

Algorithm 1: Improved FSS Encoding algorithm.

Data: Original log file R , Minimum pattern support percentage $minSup$, Minimum pattern Length $minLen$, Number of Clusters $n_clusters$

Result: Log Files F corresponding to resulting clusters

begin

 Convert R to a set of traces L ;

 From L , extract frequent sub-sequences FSS with length $\geq minLen$ and minimum support $\geq minSup$;

 From FSS , remove $x \in FSS$ if x does not exist as is in L ;

 For $x \in FSS$, compute $Encoding(x)$;

 Sort FSS in descending order of pattern lengths ;

 For each trace in L , replace any existing FSS by their encoding;

 Remove traces from L where no FSS is found;

 For remaining traces in L , replace remaining activities with their frequency ;

 Scale the values in the traces using MinMax Scaler, to have a range of $[0, 1]$;

 Add padding of -1 for the traces to have same lengths ;

 Cluster the traces in L into $n_clusters$;

 Generate Log Files F for resulting clusters;

 Return F ;

end

Subsequently, we employ the PrefixSpan algorithm (Pel et al., 2001) to extract sequential patterns $FSSs$ from the modified event logs L . This algorithm efficiently identifies recurring patterns within the traces, aiding in the discovery of significant sequences of activities over time.

In the baseline, the frequent subsequences selection criteria using PrefixSpan is the top-k patterns, that extracts the top k frequent patterns extracted by PrefixSpan leading to a loss of control over the relevance of patterns based on their support. This could result in situations where all top-k patterns have support percentages above 90%, or conversely, some patterns have support percentages below 50%. Also, using top-k patterns can be computationally expensive because it requires generating and checking many potential patterns. In our approach, we refine the pattern selection criteria by incorporating two thresholds,

(i) a minimum support percentage ($minSup$) and (ii) a pattern length ($minLen$). In this refinement, the dataset is scanned once to determine the support of candidate patterns and then filters out those below the specified threshold. The support of a pattern X is the ratio of traces in which X appears to the total number of traces, and the length of a pattern is the number of activities it includes. Also, as PrefixSpan can extract patterns that do not exist as they are in the traces, the extracted patterns are filtered from such cases.

Following, the encoding of each FSS is calculated using equation 1 and the FSSs are sorted in descending order of their length (the longer the pattern, the more important it is). This defines the priority of replacement in the next step. For each trace, where found, the FSS is replaced by its encoding. If 2 FSSs are found in the same trace, the longer FSS is first replaced and then the rest of the trace is searched for other FSSs. After the replacement, traces with no found FSSs are removed as they count as unrepresentative ones. For remaining traces, individual activities that did not belong to a pattern are replaced by their frequency as previously explained. Finally, the encoded values are scaled to a range of $[0,1]$ and clustering is performed on the resulting traces converted to numerical vectors. Finally, the log files corresponding to each cluster are generated and returned as an output. These files are later used to discover a process model describing the general behavior of learners in each cluster.

In a nutshell, our method enhances trace encoding in two key ways. First, it refines the selection criteria for patterns by replacing the top-k approach with a combination of minimum support percentage and pattern length. Second, in the conversion of traces, individual activities within traces possessing FSSs are substituted with their respective frequencies instead of a uniform value of 1. This alteration preserves the identity and positional significance of less frequent events. These enhancements have a direct impact on the representation of traces, consequently elevating the quality of trace clustering. As a result, this improvement facilitates a more detailed understanding and analysis of learning scenarios within each cluster.

4 DATA TREATMENT

As the context of this work lies within the previously proposed framework, as mentioned in Section 1, the data we are dealing with is Moodle Logs. This section is dedicated to elaborate on the data collection and treatment steps of the former.

4.1 Data Collection and Preprocessing

Moodle, a well-known learning management system that is extensively used in universities and educational institutions, contains a logging system that captures any user's activity in the system at any point in time. In the current study, Moodle event logs of 471 students enrolled in courses at Frederick University's Department of Computer Science and Engineering in Cyprus from 2018 to 2022 were collected.

The collected event logs were cleaned to keep only actions made by students on Moodle while studying, such as taking courses, tests, and assignments, as the initial logs included the actions taken by all users in the system (students, instructors, assisting instructors, manager, etc.). Additionally, a unique identifier for each student was created to be used in the trace clustering later on.

The structure of a single log file is illustrated by Table 2. The "Regnum" is the registration number, used as a unique identifier to track the path of a student throughout different courses and different years, i.e. it is used as "CaseId". The "Timestamp" records the exact time of each event taken by the students, used to order the events. While the "Event Name" is used as the activity and the "Event Context" gives information about the concerned learning resource (file, assignment, folder, etc.) affected by the event. Finally, the "Description" explains the event in a more detailed manner.

Table 2: The structure of a single event log.

Regnum	Timestamp	Event Context	Event Name	Description
--------	-----------	---------------	------------	-------------

The initial number of event names was 65, including events related to course actions, quiz taking, assignments submissions, chats and discussions, profile viewing, and others. Only 14 events are kept, as shown in Table 3. These events were chosen as they particularly show actions like completion of an assignment or a learning resource, assessment, taking feedback, studying, and exploring. The log is filtered given the chosen events to finally end with 471 students with a total of 3942 traces for the period of 2018 until 2022.

Table 3: The chosen event names from Moodle logs.

Event Name	
"A submission has been submitted"	"Quiz attempt submitted"
"Course activity completion updated"	"Course module viewed"
"Zip archive of folder downloaded"	"Content page viewed"
"Clicked join meeting button"	"Course summary viewed"
"Course module instance list viewed"	"Sessions viewed"
"Lesson started"	"Lesson resumed"
"Feedback viewed"	"Course viewed"

Table 4: The generated semantic activity.

Semantic Activity			
Study_P	Study_A	Revise	Expand
Exercise_P	Exercise_A	View	Interact
Assess_P	Assess_A	Feedback	Apply

Table 5: The structure of the trace file.

CaseId	Trace
1.course1	<View, Exercise_P, Assess_A >
1.course2	<View, View, Study_P, Study_A >
2.course1	<Interact >

4.2 Enrichment /Transformation

As the collected logs express the interaction between students and Moodle of different courses, a transformation step followed to create a new activity that gives more information on the pedagogical action done by the student. We refer to this activity as ‘Semantic Activity’. It is rule-based created using the ‘Event Context’, ‘Event Name’ and ‘Description’ from the original logs. The details of this transformation step are outside the scope of this paper. This activity can have one of the 12 values presented in Table 4. We differentiate between two types of actions: passive and active (denoted by _P and _A respectively). When a student downloads a lecture material to study it, it is considered a ‘passive’ action as there is no guarantee of completion. When a student submits an assignment, we presume they completed exercises, referred to as ‘active’ actions.

4.3 Traces Preparation

This part explains the initial step of Algorithm 1. To perform clustering, the traces should be extracted from the event logs. As each event log corresponds to activities done in one course, we define a trace as an ordered sequence of events taken by a student in one course. Thus, the trace file used to perform clustering looks as illustrated in Table 5, where the CaseId is the unique value of a student behaving in one course and the trace t_i ($1 \leq i \leq k$) is an ordered sequence of n_i ‘Semantic Activity’ events $t_i = \langle se_{i1}, se_{i2}, \dots, se_{in_i} \rangle$ made by the same CaseId.

5 IMPLEMENTATION AND RESULTS

In what follows, we illustrate the improvement implied by our approach at the level of the extracted pat-

Table 6: Comparison among extracted patterns.

Parameters	Baseline	Our Approach
# of Extracted Patterns	K = 100	(MinSup = 80%, MinLen = 2)
# of Patterns Existing in Traces	100	1412
	32	248
Among Patterns that exist as they are in the Traces		
[Min - Max] Pattern Length	[1 - 6]	[2 - 9]
[Min - Max] Pattern Support %	[86% - 100%]	[80% - 95%]
On Trace Level		
[Min - Max] Original Trace Length	[2 - 5599]	[2 - 5599]
[Min - Max] Encoded Trace Length	[1 - 2484]	[1 - 1618]
# of Traces with no FSS	49	45

terns and their encoding, the level of clusters and the discovered models.

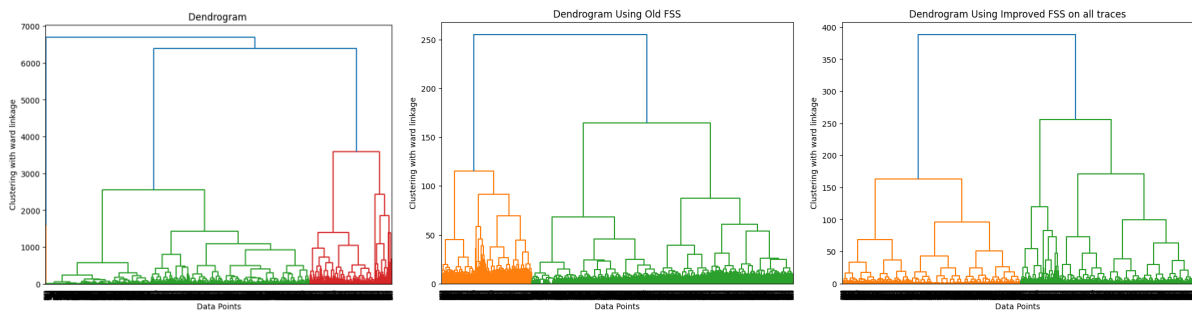
5.1 Patterns Extraction and Encoding

Table 6 compares the baseline and our approach in pattern extraction and trace encoding. Our refined selection criteria yield better control over minimum support and pattern length, resulting in more and longer patterns with high support. The encoded traces in our approach are shorter, emphasizing longer and more significant patterns, outperforming the baseline in both extraction and encoding levels, as reflected in subsequent trace clustering results.

5.2 Trace Clustering

In the final section of Algorithm 1, FSS-encoded traces are clustered using Hierarchical Agglomerative Clustering (HAC) with ward linkage, known for merging similar clusters in a bottom-up approach. Our approach is demonstrated as efficient through a comparison with baseline and ‘Activity Profile’ (Song et al., 2008), where the latter transforms traces into binary vectors (0 and 1) using one-hot encoding. The vector length equals the number of unique activities, providing a binary representation of activity presence in each trace. As for the clustering, the optimal number of clusters, determined through dendrogram analysis, is found to be 3 for all approaches as shown in Figure 1. Evaluation metrics, including Silhouette coefficient (-1 to 1, higher is better) and Davies Bouldin index (lower is better), reveal the quality of resulting clusters, as presented in Table 7.

Despite the initial impression that the Silhouette is better without FSS, detailed analysis in Table 7 reveals potential misinterpretations of numeric values. The Activity Profile yields clusters with almost all elements in one, rendering its results meaningless. In contrast, the baseline divides traces into different clusters, but they lack clear separation, as reflected in their Silhouette, while our approach combines well-divided traces with acceptable Silhouettes for each cluster.



(a) On activity profile encoded traces.

(b) On baseline encoded traces.

(c) On improved FSS encoded traces.

Figure 1: The dendrogram of HAC using ward linkage.

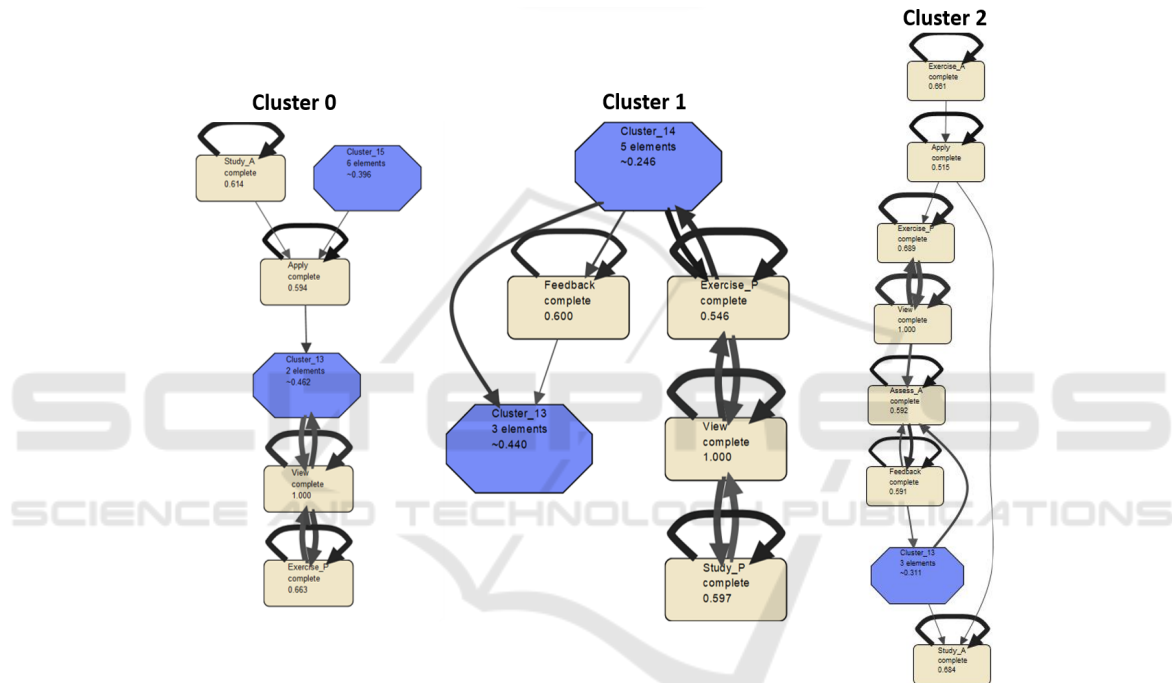


Figure 2: Fuzzy models of the resulting clusters.

Table 7: Cluster and Silhouette Analysis with 3 Clusters.

		Activity Profile	Baseline	Our Approach
Final # of Traces		3942	3893	3897
Silhouette Coefficient		0.546	0.117	0.360
Davies Bouldin Index		0.720	2.43	1.15
<i>Detailed Silhouette Analysis</i>				
Cluster 0	# of Traces	710	1000	1496
	Silhouette	0.156	-0.065	0.260
Cluster 1	# of Traces	3	1482	1961
	Silhouette	0.520	0.038	0.470
Cluster 2	# of Traces	3229	1411	440
	Silhouette	0.632	0.408	0.250

5.3 Process Models

The resulting clusters are used with the Fuzzy Miner algorithm in ProM tool to discover process models, chosen for its capacity to generate simplified models with a focus on significant nodes and well-correlated

edges (Van der Aalst, 2016). With Fuzzy Miner, less significant but highly correlated nodes are aggregated, i.e. hidden in clusters within the simplified model. Figure 2 displays the models of clusters 0, 1, and 2, simplified using node significance, leading to the aggregation of some nodes into clusters. Analyzing the learning scenarios through model readings, Cluster 1, representing the majority of learners, primarily engages in routine activities such as viewing, studying, and solving exercises. In contrast, Cluster 0, the second-largest cluster, encompasses learners who engage in routine tasks but also actively ‘Apply’ knowledge, often through project submissions. Lastly, Cluster 2, with the smallest learner count, consists of individuals who show a preference for quizzes and tests as part of their learning paths.

6 CONCLUSION

This study revolves around a previous work presenting a framework aimed at providing personalized adaptive learning paths, taking into account a learner's objective and leveraging the learning experience of previous learners. Employing Process Mining, we extract past learning experiences through the discovery of learning scenarios. However, dealing with the unstructured and voluminous Moodle data, that holds specific learning characteristics, poses a challenge, making trace clustering crucial. Thus, our approach enhances a feature-based Frequent-Subsequence (FSS) trace clustering method by refining pattern selection, particularly preserving the uniqueness of less frequent events. Applied to Moodle logs, our method demonstrates significant improvements, generating more and longer patterns, influencing encoding results, and leading to better clusters reflected by the silhouette coefficient. The identified clusters reveal three distinct learning scenarios: one characterized by a focus on studying and solving exercises, another by the application of acquired knowledge through projects, and a third by a preference for undergoing more assessments. These scenarios provide valuable insights for tailoring personalized recommendations. Future work involves integrating these findings into the recommendation framework, leveraging past learning experiences for more effective guidance. It's noteworthy that extensive testing of clustering algorithms and linkage criteria preceded the selection of the best-performing approach presented in this work.

REFERENCES

- Aggarwal, C. C. (2016). *Recommender Systems*. Springer Int. Publishing, Cham.
- Bose, R. J. C. and Van der Aalst, W. M. (2009). Context aware trace clustering: Towards improving process mining results. In *Proceedings of the Int. Conf. on Data Mining*, pages 401–412. SIAM.
- Bose, R. J. C. and van der Aalst, W. M. (2009). Trace clustering based on conserved patterns: Towards achieving better process models. In *Int. Conf. on Business Process Management*, pages 170–181. Springer.
- Cadez, I., Heckerman, D., Meek, C., Smyth, P., and White, S. (2003). Model-based clustering and visualization of navigation patterns on a web site. *Data mining and knowledge discovery*, 7(4):399–424.
- Cenka, N. and Anggun, B. (2022). Analysing student behaviour in a learning management system using a process mining approach. *Knowledge Management & E-Learning: An Int. Journal*, 14(1):62–80.
- Chatain, T., Carmona, J., and Van Dongen, B. (2017). Alignment-based trace clustering. In *Int. Conf. on Conceptual Modeling*, pages 295–308. Springer.
- De Koninck, P. and De Weerd, J. (2019). Scalable mixed-paradigm trace clustering using super-instances. In *2019 Int. Conf. on Process Mining*, pages 17–24. IEEE.
- De Weerd, J., Vanden Broucke, S., Vanthienen, J., and Baeens, B. (2013). Active trace clustering for improved process discovery. *IEEE Transactions on Knowledge and Data Engineering*, 25(12):2708–2720.
- Diamantini, C., Genga, L., and Potena, D. (2016). Behavioral process mining for unstructured processes. *Journal of Intelligent Information Systems*, 47(1):5–32.
- Ferreira, D., Zacarias, M., Malheiros, M., and Ferreira, P. (2007). Approaching process mining with sequence clustering: Experiments and findings. In *Int. Conf. on business process management*, pages 360–374. Springer.
- Joudieh, N., Eteokleous, N., Champagnat, R., Rabah, M., and Nowakowski, S. (2023). Employing a process mining approach to recommend personalized adaptive learning paths in blended-learning environments. In *12th Int. Conf. in Open and Distance Learning, Athens, Greece*.
- Laksitowening, K. A., Prasetya, M. D., Suwawi, D. D. J., Herdiani, A., et al. (2023). Capturing students' dynamic learning pattern based on activity logs using hierarchical clustering. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 7(1):34–40.
- Lu, X., Tabatabaei, S. A., Hoogendoorn, M., and Reijers, H. A. (2019). Trace clustering on very large event data in healthcare using frequent sequence patterns. In *Int. Conf. on Business Process Management*, pages 198–215. Springer.
- Pel, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proc. 17th IEEE Int. Conf. on Data Engineering. Heidelberg, Germany*, pages 215–224.
- Song, M., Gunther, C. W., and Van der Aalst, W. M. (2008). Trace clustering in process mining. In *Int. Conf. on Business Process Management*, pages 109–120. Springer.
- Trabelsi, M., Suire, C., Morcos, J., and Champagnat, R. (2021). A new methodology to bring out typical users interactions in digital libraries. In *2021 ACM/IEEE Joint Conf. on Digital Libraries (JCDL)*, pages 11–20.
- Van der Aalst, W. (2016). *Process mining: data science in action*. Springer.
- Zandkarimi, F., Rehse, J.-R., Soudmand, P., and Hoehle, H. (2020). A generic framework for trace clustering in process mining. In *2020 2nd Int. Conf. on Process Mining*, pages 177–184. IEEE.
- Zhang, T., Taub, M., and Chen, Z. (2022). A multi-level trace clustering analysis scheme for measuring students' self-regulated learning behavior in a mastery-based online learning environment. In *LAK22: 12th Int. Learning Analytics and Knowledge Conf.*, pages 197–207.