6-18-2022

Trace Clustering for User Behavior Mining

Luka Abb
University of Mannheim, labb@mail.uni-mannheim.de

Carsten Bormann
SAP SE, carsten.bormann@sap.com

Han van der Aa
University of Mannheim, han@informatik.uni-mannheim.de

Jana R. Rehse
University of Mannheim, rehse@uni-mannheim.de

TRACE CLUSTERING FOR USER BEHAVIOR MINING

Research Paper

Luka Abb, University of Mannheim, Mannheim, Germany, luka.abb@uni-mannheim.de

Carsten Bormann, SAP SE, Walldorf, Germany, carsten.bormann@sap.com

Han van der Aa, University of Mannheim, Mannheim, Germany, han@informatik.uni-mannheim.de

Jana-Rebecca Rehse, University of Mannheim, Mannheim, Germany, rehse@uni-mannheim.de

Abstract

Business information systems support a large variety of business processes and tasks, yet organizations rarely understand how users interact with these systems. User Behavior Mining aims to address this by applying process mining techniques to UI logs, i.e., detailed records of interactions with a system's user interface. Insights gained from this type of data hold great potential for usability engineering and task automation, but the complexity of UI logs can make them challenging to analyze. In this paper, we explore trace clustering as a means to structure UI logs and reduce this complexity. In particular, we apply different trace clustering approaches to a real-life UI log and show that the cluster-level process models reveal useful information about user behavior. At the same time, we find configurations in which trace clustering fails to generate satisfactory partitions. Our results also demonstrate that recently proposed representation learning techniques for process traces can be effectively employed in a realistic setting.

Keywords: Process Mining, User Behavior Mining, Trace Clustering, Representation Learning

1 Introduction

Business information systems, such as Enterprise Resource Planning (ERP) and Customer Relations Management (CRM) systems, support a wide range of business operations and allow users to perform a broad set of tasks in an integrated and flexible manner (Beynon-Davies, 2019). Although such flexibility is highly beneficial from a user perspective, it causes organizations and software vendors to lose track of how users actually employ these software applications to conduct their work. Having such insights on user behavior could be highly valuable for purposes such as usability engineering (Nielsen, 1994), which studies how software is used and might be improved, and task automation (van der Aalst, Bichler, and Heinzl, 2018), which aims to recognize and automate repetitive user operations.

These insights can be obtained by analyzing records of how users of a software application interact with its user interface (UI). In this paper, we will refer to such data-driven analysis as *User Behavior Mining* (UBM). The basis of UBM are so-called *UI logs*, which record sequences of events as performed by a user. Each event in these sequences, referred to as *traces*, corresponds to a low-level user action in a software system, such as clicking a button or entering a string into a text field. Techniques from the field of process mining can be used to analyze UI logs, for instance by applying *process discovery* to generate a visual representation of the recorded user behavior in form of a process model (Rubin et al., 2014). However, the complexity of UI logs, in terms of their size and behavioral variance, impedes the ability to directly obtain concrete insights. Process models discovered from UI logs tend to be "spaghetti" models, i.e., highly complex models that, due to hundreds of nodes and crossing edges, are too difficult for humans to understand and interpret (van der Aalst, 2011; Veiga and Ferreira, 2010). Therefore, they typically fail to provide useful insights about a process.

One way to address this problem is to preprocess a log by means of *trace clustering*, which partitions it into smaller groups of similar traces. Ideally, applying process discovery to those more cohesive sub-logs results in a set of less complex and better readable process models, which jointly provide a better overview of the full log (Zandkarimi et al., 2020). Given this potential, the objective of our research is to explore the efficacy and applicability of trace clustering in the context of UBM. In particular, we want to determine whether trace clustering can be used on UI logs to obtain meaningful groups of execution traces and useful visualizations of the recorded user behavior. This results in the following two research questions:

RQ1: Which trace clustering techniques can best partition a UI log so that traces that represent similar user behavior end up in the same cluster?
RQ2: How useful and comprehensible are the process models discovered from the cluster-level sub-logs?

To answer these questions, we conduct an experimental evaluation on a real-life event log, which contains user behavior recorded from interactions with an ERP system. We compare the performance of multiple clustering techniques to identify those that are best suited for highly complex UI logs like the one at hand. Because most established trace clustering techniques were developed for applications in conventional process mining (Zandkarimi et al., 2020), they do not scale well to large event logs with a high number of activities and trace variants, such as UI logs. Therefore, we approach the trace clustering task by first embedding traces into a lower-dimensional vector space using so-called representation learning techniques, which are particularly suited to deal well with high-dimensional event data (De Koninck, Vanden Broucke, and De Weerdt, 2018). In our evaluation, we combine three different representation learning techniques with three clustering algorithms and compare the results of these nine approaches with regard to our research questions.
The remainder of this paper is structured as follows: Section 2 elaborates on the opportunities and challenges of UBM. Section 3 discusses related work on user behavior analysis, trace clustering, and representation learning. In Section 4, we describe the setup of the conducted evaluation. Its results are presented in Section 5 and discussed in Section 6, before the paper is concluded in Section 7.

2 Opportunities & Challenges of User Behavior Mining

The goal of UBM is to analyze how users interact with a software application, which is achieved by applying concepts and techniques from process mining to UI logs. Although it can be seen as an application of process mining, UBM has a different focus: Whereas process mining is mainly interested in determining *what* was done, UBM also provides insights into *how* a certain task was performed by examining the precise sequences of clicks and/or keyboard inputs that were executed. It hence offers numerous opportunities to gain novel insights into user behavior. However, it also poses a number of challenges, which make it difficult to apply existing process mining techniques to UI logs.

2.1 Opportunities

Until recently, the analysis of user behavior has primarily been a research topic in the context of web-based applications (Ding, Li, and Chatterjee, 2015; Ho, Bodoff, and Tam, 2010; Raphaeli, Goldstein, and Fink, 2017; Srivastava et al., 2000; Wang et al., 2017). On the web, records of user activities are readily available through server logs, which enable the analysis of clickstreams, i.e., paths that users take when navigating through and across websites (Facca and Lanzi, 2005). However, clickstream analysis and web analytics tools such as Google Analytics typically do not abstract from user actions to actual behavioral patterns (Poggi et al., 2013) and do not connect these actions with the business processes underlying a website's interface. They thus provide a simplified and incomplete view of user behavior. The application of these tools is also naturally limited to browser-based software systems.
In this paper, we focus on UBM in ERP systems, where it holds great potential for usability engineering (Maalej et al., 2016) and task automation (Leno, Polyvyanyy, et al., 2021). In the context of *usability engineering*, potential use cases for UBM include: (1) identifying common usability issues or missing

features, (2) pinpointing system areas where user behavior deviates from system design, (3) identifying groups of users with similar usage habits and adapting the UI to their preferences, and (4) developing new UI components, such as a virtual assistant that suggests the next activity in a task based on how it is typically performed by other users. With respect to *task automation*, UBM stands at the core of Robotic Process Automation (RPA). This field aims to automate tasks and end-to-end processes by recording how human users interact with a system and then making bots emulate this behavior (van der Aalst, Bichler, and Heinzl, 2018). In RPA, UBM techniques have two main use cases: (1) they can be used to find repetitive tasks with high automation potential (Leno, Polyvyanyy, et al., 2021) and (2) they are needed to derive automation scripts from records of user activities.

Despite these opportunities, data-driven analysis through UBM has so far received little attention. Instead, many software vendors gauge the usability of their product by collecting explicit user feedback, e.g., in the form of surveys or feedback forms, but these feedback channels require additional effort on the part of users (Hoffmann et al., 2019). Explicit feedback also primarily captures user attitudes, rather than actual system usage, and it can only reveal usability issues that users actively perceive (Davis, 1989; Hoffmann et al., 2019; Parks, 2012). In contrast, UBM constitutes a data-driven, non-intrusive alternative that provides a holistic and detailed view of software usage. It can also monitor user behavior continuously and in real time, thus enabling context-specific UI adaptations.

2.2 Challenges

The essential idea behind UBM, i.e., the analysis of event logs, is the same as in standard process mining. However, various characteristics of UI logs make them particularly challenging to analyze (Dev and Liu, 2017; Leno, Polyvyanyy, et al., 2021; Weinzierl et al., 2020):

Event types and perspectives. Standard process mining typically treats all events equally. In UI logs, however, it is important to distinguish different types of events (e.g., clicks, string inputs, copy/paste, scrolling). One high-level distinction in an ERP context is between *input events* that make changes to a business object and *navigation events* that only serve to navigate through the user interface. Depending on the objective of the analysis, one can focus only on input events, i.e., take an *outcome perspective* on the user behavior, or consider both input and navigation events, i.e., take an *interaction perspective*.

Number of activities. In UBM, data is captured on the level of UI elements, so each interactive element in an application corresponds to one activity in the UI log. Because modern software systems cover a wide range of functionalities, and therefore contain many UI elements, the number of activities in a UI log can easily become an order of magnitude higher than the number activities typically found in traditional event logs. This means dealing with hundreds rather than dozens of distinct activities (Augusto et al., 2018).

Flexible control-flow. Software UIs often provide a high degree of flexibility, allowing users to perform activities in an arbitrary order (e.g., when adding multiple items to a sales order) and to revisit previously executed steps (e.g., to adjust the quantities of ordered items). Although this can be highly beneficial for the user, it also means that UBM cannot rely on the presence of clearly defined control-flow relations among the activities in a log. Along with the aforementioned high number of activities, this lack of a clear control-flow also leads to a high number of unique trace variants.

Event log size. Because UI logs capture event data at a high resolution, their size quickly exceeds that of conventional event logs, as exemplified by the publicly available dataset of the BPI Challenge 2016 (Dees and van Dongen, 2016), as well as the event log used in our evaluation.

To effectively analyze UI logs, UBM techniques therefore need to be able to handle complex, high-volume, high-cardinality data. Traditional process mining techniques commonly lack these capabilities, since they are designed for the analysis of conventional end-to-end processes. One way to reduce event log complexity and to address the aforementioned challenges is to abstract from low-level activities to higher-level ones (van Zelst et al., 2021). However, reducing the granularity of a UI log through abstraction also reduces the level of detail that the user behavior can be analyzed on. In particular, discovering a process model from an abstracted UI log can give a clear view on which high-level business activities are executed (de Leoni and Dündar, 2020), but it provides no insights into how users interact with a system's

UI. Trace clustering is an alternative approach to reduce event log complexity while retaining the level of detail. However, existing trace clustering approaches are also tailored towards relatively small and structured event logs. It is therefore not clear how well such techniques are able to deal with UI logs and, consequently, how suitable they are in the context of UBM. This paper therefore investigates if trace clustering techniques are able to handle the challenges of UI logs and which techniques are best equipped to realize the promising opportunities that UBM offers.

3 Background and Related Work

This section briefly discusses related research streams on the analysis of user behavior in ERP systems (Section 3.1), trace clustering (Section 3.2), and representation learning (Section 3.3).

3.1 Analyzing User Behavior in ERP Systems

User behavior in ERP systems is traditionally studied with qualitative empirical methods, such as interviews and questionnaires (Amoako-Gyampah, 2007; Lambeck et al., 2014; Scholtz, Cilliers, and Calitz, 2010; Topi, Lucas, and Babaian, 2005). These methods capture user attitudes and intentions, but do not allow for the analysis of actual system usage. Another approach to user behavior research is to conduct experiments, in which users perform specific tasks in an ERP system in a controlled environment (e.g., Burton-Jones and Straub, 2006; Parks, 2012). Although these experiments can provide insights into system usage, they are time-consuming and expensive, and therefore typically limited in scope.

Despite these shortcomings of empirical methods, there has been little research on data-driven approaches for the analysis of ERP user behavior. The potentials and challenges of analyzing low-level user actions in general software systems are discussed by J. Song, Luo, and Chen (2008) and Rubin et al. (2014), but they do not discuss specific applications in ERP systems. There are two case studies that apply UBM in an ERP setting to improve usability: Maruster et al. (2008) analyze user navigation patterns in a decision support system for cultivar selection. They succeed in identifying areas of the UI where a redesign would be beneficial, and also derive potentials for user-group-specific personalization. Astromskis, Janes, and Mairegger (2015) mine UI logs from an industrial ERP system and show that simplifying the system's UI can be used to increase employee efficiency.

ERP usage data has also been analyzed in the context of RPA, which is referred to as Robotic Process Mining (Leno, Polyvyanyy, et al., 2021) or Desktop Activity Mining (Linn, Zimmermann, and Werth, 2018). The goal of these techniques is to identify ERP routines that are suitable for automation (Jimenez-Ramirez et al., 2019; Leno, Augusto, et al., 2020; Linn, Zimmermann, and Werth, 2018); they therefore take a relatively narrow perspective that only covers one aspect of user behavior. They have also so far only been applied in environments with isolated processes and not in real-life ERP systems.

A more theoretical lens for studying user behavior in ERP systems is provided by organizational routines, i.e., repetitive, recognizable patterns of interdependent actions, carried out by multiple actors (Pentland, Vaast, and Wolf, 2021). Routines are enacted to fulfill some purpose within or for an organization, often to accomplish some task in an ERP system, such as hiring or budgeting (Feldman, 2016). A current stream of research propagates the data-driven analysis of organizational routines in organizations by means of process mining, which provides a detailed view into routine enactment (Grisold et al., 2020; Pentland, Vaast, and Wolf, 2021). However, so far, organizational routines have mainly been studied through regular event logs instead of UI logs (Wurm et al., 2021).

3.2 Trace Clustering

Clustering is an unsupervised technique that partitions a dataset so that the data points in one cluster are more similar to each other than to points in other clusters (Jain and Dubes, 1988). The high number of different clustering algorithms that exist today can be broadly categorized into partitional, hierarchical, and density-based ones (Xu and Tian, 2015). They can also be distinguished by whether they assign each point to exactly one cluster (hard clustering) or compute a cluster membership probability (fuzzy clustering). Finally, some clustering algorithms require the number of clusters to be specified as a hyperparameter, whereas others can determine it automatically. For clustering to be successful, it is essential to choose an adequate metric to measure the (dis-)similarity between data points.

In the context of process mining, *trace clustering* refers to the task of grouping together similar traces in an event log. The two clustering algorithms that are commonly used for this purpose are k-means clustering and agglomerative hierarchical clustering. Common distance measures are the Euclidian, Hamming, Jaccard, cosine, and edit distance (Thaler et al., 2015). Most trace clustering approaches revert to those basic algorithms; they mainly differ with regard to the trace representation, i.e., which features are used and how they are encoded. In the following, we therefore focus on this aspect of trace clustering. For a comprehensive overview of the topic, we refer the reader to Zandkarimi et al. (2020).

Most trace clustering techniques represent traces as feature vectors. Greco et al. (2006) use frequent sub-sequences of events as features and are the first to show that it is possible to find meaningful clusters in real and synthetic process logs. M. Song, Günther, and van der Aalst (2009) introduce the concept of profiles, which are basic sets of features that can be used to describe a trace from a particular perspective. For example, the activity profile and transition profile encompass attributes such as event labels and timestamps that capture the control-flow of the trace. Bose and van der Aalst (2010) propose additional sets of features (maximal repeats and (near-)super-maximal repeats) that are derived from conserved patterns, i.e., sub-sequences of events that are conserved across multiple traces.

Other approaches to trace clustering include Ferreira et al. (2007) and Veiga and Ferreira (2010), who do not use a vector representation but instead cluster activity sequences by representing clusters as first-order Markov chains. They succeed in clustering a real event log, but note that their approach requires the event log to be carefully preprocessed. Bose and van der Aalst (2009) cluster event sequences by means of generic edit distance, which is computed by counting the number of insertion, deletion, and substitution operations required to transform one sequence into another. They propose an algorithm to derive separate costs for each operation on each individual event based on the context of that event within a sequence. ActiTraC (De Weerdt et al., 2013) takes a domain-specific approach to trace clustering. Traces are grouped not because they exhibit similar behavior, but because they fit well into a cluster's process model. The algorithm iteratively adds traces to a cluster as long as this operation does not cause the accuracy of the cluster's process model to fall below a specified threshold. Both edit distance computation and the ActiTraC algorithm have quadratic-time complexity with respect to the number of activities, and therefore do not scale well to typical UBM logs. Noteworthy is also the work of Lu et al. (2019), who cluster a hospital event log with a large number of activities. However, their semi-supervised clustering approach assumes that domain experts can provide a small labeled set of cases for each cluster to be found, which is not realistic in UBM. In this paper, we particularly focus on trace clustering that uses *representation learning*, due to its suitability to deal with the complex characteristics of UI logs, as described next.

3.3 Representation Learning

The goal of representation learning is to embed high-dimensional data in a lower-dimensional vector space so that the lower-dimensional representation (called an *embedding*) retains as much of the original structure as possible. Approaches range from simple matrix decomposition algorithms such as Principal Component Analysis to non-linear manifold learning techniques, such as Multi-dimensional Scaling (Kruskal, 1964) and Isomap (Tenenbaum, Silva, and Langford, 2000), and neural networks. Representation learning can address an important challenge in feature-vector-based trace clustering: event labels and many context attributes are categorical, but clustering algorithms require numeric features as input. Many trace clustering techniques encode events by assigning bag-of-activities or bag-of-n-grams vectors to traces, but this leads to a very high-dimensional and sparse feature space when the event log contains a large number of distinct activities. Embedding techniques reduce the dimensionality of this feature space

and thereby make it more likely that a clustering algorithm can find a good partition (De Koninck, Vanden Broucke, and De Weerdt, 2018).
Representation learning in trace clustering was first proposed by M. Song, Yang, et al. (2013), who reduce the dimensionality of the feature space by applying matrix decomposition techniques and random projection, but find that this approach does not lead to better clusters. More recently, however, techniques that learn trace representations with neural networks have shown promising results. De Koninck, Vanden Broucke, and De Weerdt (2018) create trace embeddings by adapting the popular Word2Vec representation learning technique from natural language processing (Mikolov, Corrado, et al., 2013; Mikolov, Sutskever, et al., 2013). They show that neural-network-generated embeddings can accurately represent process traces, and that clustering the trace vectors produces better results than previous trace clustering techniques. One downside of the Word2Vec approach is that it only encodes control-flow information, but multiple authors have since proposed extensions or other network architectures that also incorporate context attributes (Bui et al., 2020; Guzzo et al., 2021; Luettgen et al., 2021; Seeliger et al., 2021).
In our evaluation experiments, described next, we compare the performance of three such representation learning techniques: Isomap, Autoencoder, and Word2Vec.

4 Evaluation Setup

To answer our research questions, we conduct an experimental evaluation on a large real-world UI log. We assess nine clustering methods in terms of the cluster quality they achieve and the usefulness and comprehensibility of the subsequently discovered process models. We decided against an evaluation on synthetic data because we want to evaluate trace clustering in a real UBM setting. The details on the employed dataset, clustering methods and settings, and evaluation measures are described below. Because our research was conducted with a partner company, we are unfortunately unable to make the dataset or source code available.

4.1 Dataset

We conduct our experiments based on a UI log that contains data recorded over a period of ten months in ERP systems across multiple administrative departments of a multinational company. The log entries reflect a large variety of tasks performed by employees as part of their daily work. As shown in the excerpt in Table 1, each event corresponds to a single low-level user action and refers to the identifier of the UI element that the user interacted with, which we use as the activity label. The events are divided into two event types: *input events* correspond to actions that result in changes to the underlying business object, whereas *navigation events* correspond to actions that serve to navigate the system's UI.
Each case is made up of all actions that are executed on a single instance of a business object as part of the same higher-level task, for example, filling out a sales order or creating an invoice. Within a case, the events are ordered and have timestamps. Each case has three attributes: a *business object type (BOT)* and *document type (DT)*, which together specify the type of form processed, and a *department code (DC)*. It should be noted that, due to a bug in an earlier version of the logging application, these case-level attributes are incorrectly assigned in a small number of cases in our dataset.
We prepare the data for analysis by filtering out all activities that are either very rare, i.e., do not occur in more than 0.1% of traces, or do not relate to the execution of a specific task or process, e.g., saving, scrolling, or zooming. The filtered UI log contains $46,000$ traces with $564,000$ events, of which about 40% are input events, and 500 distinct activities.

4.2 Evaluation Settings

Table 2 shows an overview of the configurations used in our evaluation. We employ four different data selections (combining the two perspectives with each of the two feature sets) and nine clustering approaches (combining each of the three embedding techniques with the three clustering algorithms).

Case ID	Timestamp	Activity (UI element)	Event type	BOT	DT	DC
11276	10:23:54	new_contract	input	SalesOrder	GU	G0504
11276	10:23:56	general_notes	navigation	SalesOrder	GU	G0504
11276	10:24:02	find	navigation	SalesOrder	GU	G0504
11276	10:24:12	new_contract_number	input	SalesOrder	GU	G0504
11276	10:24:15	table_dates	navigation	SalesOrder	GU	G0504
11276	10:24:20	table_dates	input	SalesOrder	GU	G0504
15432	10:30:01	user	navigation	Billing	GC	G0504
15432	10:30:04	user	input	Billing	GC	G0504
15432	10:30:10	password	navigation	Billing	GC	G0504
15432	10:30:11	password	input	Billing	GC	G0504
13002	12:50:41	display_billing_list	navigation	Billing	EEV	G0504
13002	12:50:44	vb2	navigation	Billing	EEV	G0504
13002	12:50:48	next_item	navigation	Billing	EEV	G0504
13002	12:50:59	exit	navigation	Billing	EEV	G0504
13002	12:52:10	display_billing_list	navigation	Billing	EEV	G0504
13002	12:54:52	confirm	input	Billing	EEV	G0504

Table 1. UI log excerpt (last two columns anonymized)

Data selections. We apply trace clustering to two different perspectives on the UI log: one that includes all events (*interaction perspective*) and one from which the navigation events are omitted (*outcome perspective*). We combine each perspective with two feature sets: one that only captures the activities per trace and one that also incorporates the three case attributes as additional features. Overall, this results in four different data selections.

Data selections	
Perspectives:	*Outcome* (only input events), *interaction* (input and navigation events)
Feature sets:	*Activities only*, *Activities + case attributes*
Clustering approaches	
Embedding techniques:	*Isomap*, *Autoencoder*, *Word2Vec*
Clustering algorithms:	*k-means*, *agglomerative hierarchical clustering (AHC)*, *DBSCAN*

Table 2. Data selections and clustering approaches employed in the evaluation.

Clustering approaches. We encode traces as feature vectors and use representation learning techniques to reduce the dimensionality of the feature matrix, as discussed in Section 3.3. Since our primary objective is not to compare state-of-the-art algorithms, but rather to determine how well these techniques generally work in a UBM setting, we choose to use three standard, well-established representation learning techniques to obtain embeddings: *Isomap* (Tenenbaum, Silva, and Langford, 2000), *Autoencoder* (Hinton and Salakhutdinov, 2006), and *Word2Vec* (Mikolov, Corrado, et al., 2013; Mikolov, Sutskever, et al., 2013). As mentioned earlier, UI interactions often do not have a rigid control-flow. For Isomap and the Autoencoder, we therefore initially assign each trace a binary bag-of-activities vector and disregard event frequency and semantics. We create the Isomap embedding by first computing a neighborhood graph from the trace vectors with Jaccard distance as the dissimilarity measure, then estimating geodesic distances through a shortest path search, and finally reducing the dimensionality of the geodesic distance matrix with Kernel-PCA (H. Choi and S. Choi, 2007). We build the Autoencoder as a standard feedforward neural network with a single hidden, fully-connected embedding layer and train it to reconstruct the input vectors with a cross-entropy loss function.
In contrast to the previous two methods, Word2Vec is explicitly designed to encode semantic information in the embedding, so we can use it to examine whether or not considering semantics leads to different and

perhaps better clusters in our use case. For this, we generate activity skip-grams and train the network to predict context activities in a small window around the target activity, then mean-aggregate the activity vectors to create trace vectors. Note that Word2Vec is not designed to encode additional attributes and is therefore only applied to the data selections with the *activities only* feature set.

We apply three clustering algorithms in combination with each embedding technique. These include two standard algorithms, *k-means clustering* and *agglomerative hierarchical clustering* (AHC), as well as the density-based *DBSCAN*. The latter has seen many successful applications in data mining, but has only recently been used for the purpose of trace clustering (Barbon et al., 2021; de Leoni and Dündar, 2020). For each algorithm, we compute dissimilarity as Euclidean distance between the input embedding vectors.

Parameters. We select parameters for the embedding techniques so that the respective reconstruction error is minimized. An initial analysis indicates that the "true" number of clusters in our dataset is in the hundreds. However, there is a trade-off between the number of clusters found by an algorithm and the usefulness of the corresponding cluster-level process models: models mined from few large clusters are too difficult to understand, whereas too many clusters result in simpler but also less expressive process models, which are not useful to users. With respect to our second research question, our goal is to find a middle ground where models are comprehensible but not simplified so much that they become trivial. For k-means clustering and AHC, we therefore choose a relatively low number of 80 clusters, which we deemed most appropriate after manually inspecting the trace allocation for different values of the number of clusters parameter. DBSCAN cannot be explicitly restricted to a specific number of clusters, so we instead use it in an exploratory manner, i.e., to discover types of user actions in the UI log. We choose DBSCAN parameters so that (1) no cluster contains more than one third of all traces, (2) the number of outliers does not become excessive, and (3) the silhouette score of the partition is maximized. These criteria result in partitions with 500 to 1000 clusters, depending on the data selection.

Implementation and environment. We implemented our clustering approaches in Python 3.8, using the scikit-learn library (Pedregosa et al., 2011) for Isomap and the three clustering algorithms, and tensorflow (Abadi et al., 2015) for the neural networks. All experiments are performed on a Lenovo ThinkPad with four Intel i5 cores and 16 GB RAM.

4.3 Evaluation Measures

We assess the clusters obtained in the various configurations with respect to our two research questions:

RQ1 (Cluster quality). We assess the quality of obtained clusters using the widely-employed *silhouette coefficient* (Rousseeuw, 1987), which measures the intra-cluster cohesion and inter-cluster separation of grouped traces. Silhouette scores close to 1 indicate dense, well-separated clusters, whereas scores close to 0 or negative ones indicate poorly separated or overlapping clusters.

Additionally, to get an intuition of how the dataset is partitioned, we visualize our clusters in two-dimensional *t-SNE projections* (van der Maaten and Hinton, 2008). We also compare the results of different combinations of representation learning techniques and clustering algorithms with the *Adjusted Rand Index* (Hubert and Arabie, 1985; Rand, 1971) to see how the different configurations relate to each other. The Adjusted Rand Index is a measure of clustering similarity that computes the ratio of allocation agreements between two partitions, adjusted for chance. An Adjusted Rand Index close to 1 indicates near-identical cluster allocation, whereas a value close to 0 indicates that the partitions are dissimilar.

RQ2 (Model usefulness & comprehensibility). There are multiple ways to evaluate the usefulness and comprehensibility of the discovered cluster-level process models. One approach would be to quantify these concepts through measures for process model complexity (Buijs, van Dongen, and van der Aalst, 2014). However, these measures can only provide a very abstract assessment of the value of a process model visualization, and they do not directly measure usefulness and comprehensibility. For instance, a model with a low complexity measure may still not allow a human observer to gain any insights if the depicted traces follow no discernable pattern. Therefore, we instead choose to qualitatively assess the visualization potential by inspecting directly-follows-graphs. To evaluate usefulness, we compare the activity sequences in the graphs with the processes and tasks supported by the ERP system and determine to which degree they match. For comprehensibility, we look at the number of nodes and edges, and assess whether a human observer can clearly discern the main execution variants.

5 Evaluation Results

This section presents the evaluation results obtained for our two research questions.

5.1 RQ1: Cluster Quality

We applied nine clustering approaches on four data selections, resulting in 36 clustering configurations. For each one, computation took less than one hour. The neural-network-based embedding techniques (Autoencoder and Word2Vec) were considerably faster than Isomap, which took upwards of 45 minutes per task. Similarly, clustering was considerably faster with k-means than with AHC and DBSCAN.

	Outcome perspective		**Interaction perspective**	
Clustering approach	**Activities only**	**Activities + attributes**	**Activities only**	**Activities + attributes**
Isomap				
K-means	0.28	0.20	0.20	0.16
Hierarchical	0.27	0.20	0.21	0.15
DBSCAN	**0.60**	**0.42**	0.26	0.09
Autoencoder				
K-means	0.37	0.26	0.25	**0.19**
Hierarchical	0.34	0.23	0.22	0.17
DBSCAN	**0.60**	0.40	**0.29**	0.10
Word2Vec				
K-means	0.29		0.21	
Hierarchical	0.30		0.20	
DBSCAN	0.18		-0.05	

Table 3. Silhouette scores

Table 3 shows the silhouette scores for each configuration. The values are quite low, which is likely due to the complexity of the UI log; only Isomap and Autoencoder embeddings with DBSCAN clustering have a score higher than 0.5. The embedding technique with the highest overall scores is the Autoencoder. As it is not restricted to a low number of clusters, DBSCAN expectedly finds the best partition when applied to Isomap and Autoencoder embeddings. Word2Vec, however, appears to only work well with k-means clustering and AHC, but not with DBSCAN, given the much lower scores for this combination.
The silhouette scores are substantially higher in the outcome perspective, where navigation events are filtered out. They are also higher when case attributes are not considered. Looking at the partitions in detail, we observe that the clusters in the interaction perspective contain traces with the same set of core activities, but with a large number of execution variants due to the navigation events. Consequently, the clustering is successful to an extent, but clusters are less cohesive and less well separated than they are in the outcome perspective. When including case attributes, we find that all algorithms separate traces strongly by those attributes and neglect the control-flow. This results in the grouping of traces that are completely unrelated from a control-flow perspective and hence produce impractical process models.
Figure 1 shows t-SNE projections of the outcome perspective, activities only configuration, in which our methods achieved the best results. Clusters found in Isomap and Autoencoder embeddings appear fairly cohesive, although some overlap or have a lot of outliers. Word2Vec has a few well separated clusters, but a lot of overlap in the others. For all embeddings, the largest clusters found by k-means and AHC are similar, whereas the ones in DBSCAN partitions are smaller and more cohesive.

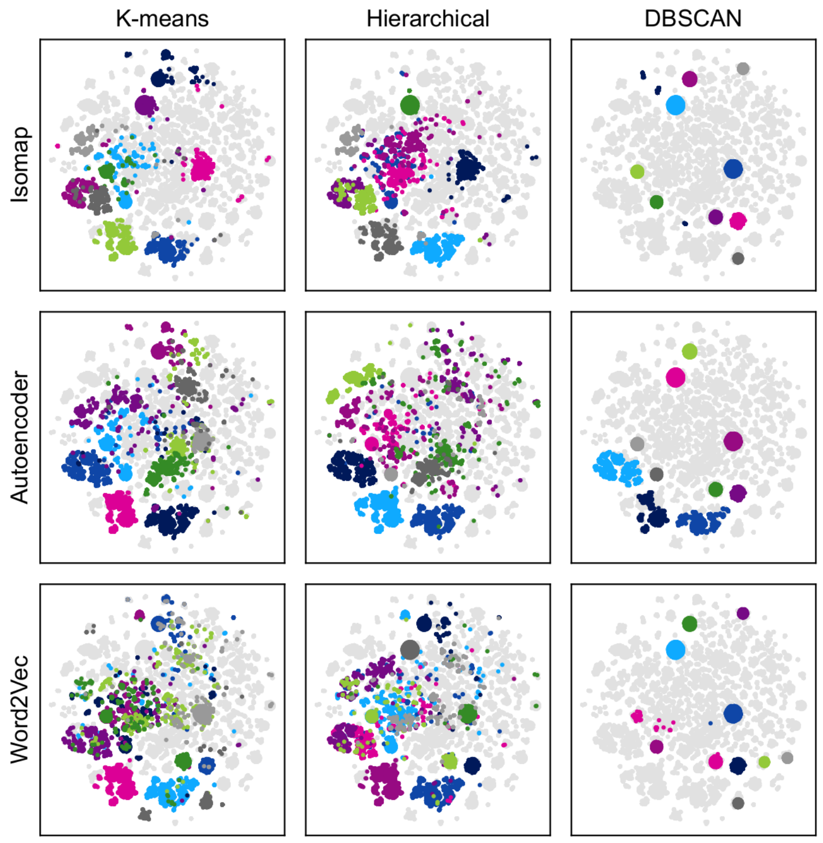

Figure 1. t-SNE projection, outcome perspective and activities only, with the ten largest cluster in each partition highlighted

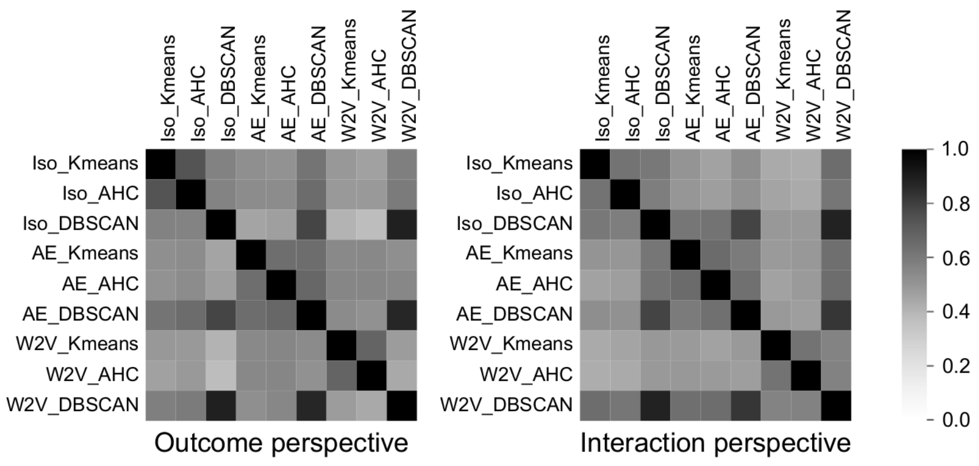

Figure 2. Partition similarity matrices (Adjusted Rand Index)

Figure 2 depicts the cluster similarity per Adjusted Rand Index between all partitions in the outcome and interaction perspectives (activities only). Agreement between methods is overall fairly high, which suggests that there is distinct user behavior in the UI log that all combinations of embedding techniques and clustering algorithms identify. The results are particularly similar between clustering algorithms applied to the same embedding. In the interaction perspective, clustering the Word2Vec embedding with k-means clustering and AHC appears to lead to unique partitions that have low agreement with the results of other methods. Although some agreement values for DBSCAN clustering appear very high, they are inflated by the large number of clusters in these partitions and should be interpreted with caution.

5.2 RQ2: Model usefulness & comprehensibility

Overall, we can say that the application of trace clustering to our UI log has succeeded in grouping related traces that together form useful process models. An example of a model produced by k-means clustering is shown in Figure 3. The traces in the cluster capture varying ways in which users cancel contracts in the ERP system. The core process flow and its main variations are clearly visible, and the process model is well comprehensible. However, it also shows several paths that only occur in a handful of cases, which indicates that there may be some potential to streamline the application's UI.

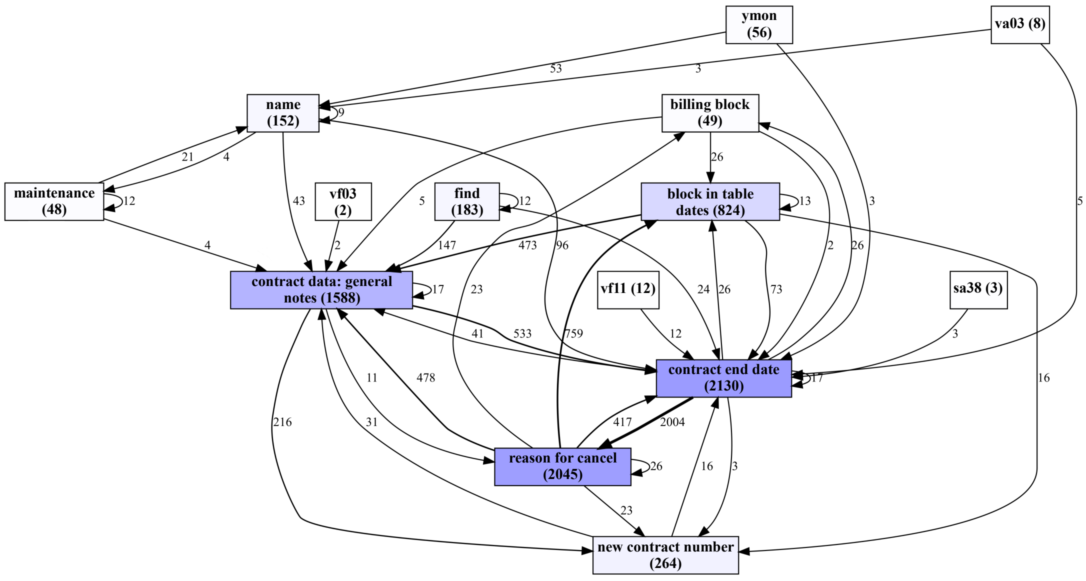

Figure 3. *Large cluster capturing the cancellation of contracts (outcome perspective, activities only, Autoencoder and k-means clustering). Labels like "vf03" refer to UI elements that access particular types of transactions in the ERP system.*

By contrast, DBSCAN clusters typically contain only a handful of activities and significantly less variation, as we did not restrict the algorithm to a certain number of clusters. Rather than showing how different users execute a process in the underlying ERP system, DBSCAN clusters reveal lower-level tasks, as, e.g., seen for the three clusters in Figure 4, which each correspond to a specific step in a process.
However, each obtained partitioning of the UI log also contains at least a few clusters that either group unrelated traces together or are too complex to be visualized in a meaningful way. This is particularly prominent for the interaction perspective, where it happens for the majority of clusters. When including case attributes in the data selection, the resulting cluster-level process models can be visualized, but are often not practically useful. For example, Figure 5 shows a single cluster that consists of four disconnected parts, which have no particular relation to each other from a behavioral perspective. Rather, the traces were clustered together because they were performed in the same department.

6 Discussion

Our results show that it is possible to find meaningful clusters of user behavior in a UI log. The larger clusters found by k-means clustering and AHC mostly contain one core (sub-)process or task with

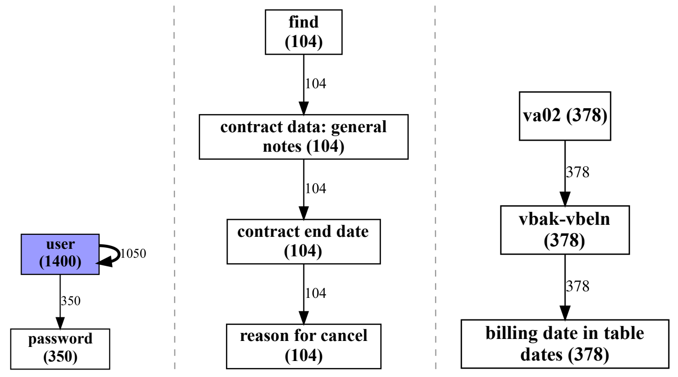

Figure 4. *Three examples of clusters found by DBSCAN (outcome perspective, activities only)*

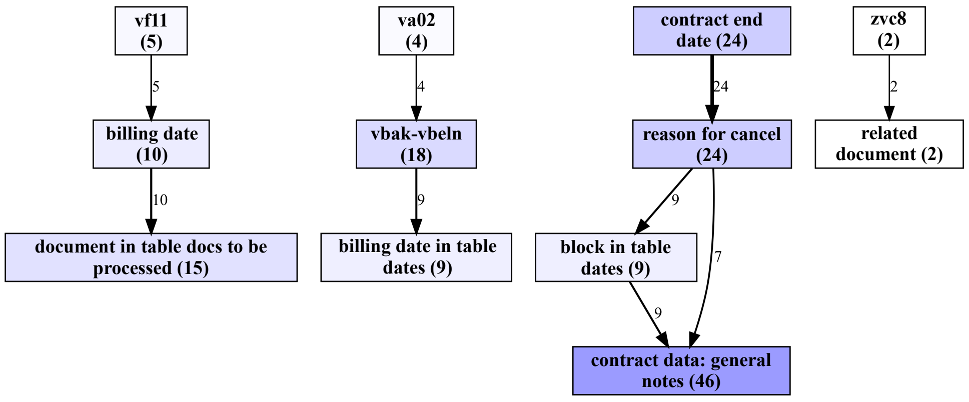

Figure 5. *One cluster that contains four distinct types of control-flow (outcome perspective, activities and case attributes; for visual clarity, only 20% of variants are displayed)*

a number of execution variants that correspond to the different ways that employees interact with the company's ERP system. Traces in the smaller DBSCAN clusters represent more fine-grained user activities that typically belong to a specific step in a larger process. This indicates that trace clustering can not only be used to improve the quality of discovered process models, but also to identify types of UI interactions in an unstructured UI log, which we expect to be a common challenge in UBM. Our clustering also significantly improves the visual clarity of the discovered process models. If we take an outcome perspective on the UI log, the majority of clusters can be visualized in an understandable way, and the process models reveal useful information about core activities and variations in user behavior. However, in the interaction perspective, the clustering results are unsatisfactory, and almost all cluster process models are incomprehensible, even when further simplified with variant hiding and filters. This indicates that user navigation paths introduce a large amount of variation into the process execution that is difficult for trace clustering techniques to handle. Although only considering input events may be sufficient for some use cases, the aspiration of UBM is to analyze the full extent of user behavior. Clustering a UI log that captures every kind of UI interaction therefore remains a challenging and largely unsolved problem.
We have also observed that incorporating context information is detrimental in our use case because the case attributes have a much stronger influence on the cluster allocation than the control-flow. This suggests that considering case-level context in trace clustering, although generally desirable, can actually have an adverse effect on the results if there is a lot of control-flow variation. It should be noted that some of the case attributes are assigned incorrectly in our dataset, which may also be a contributing factor.
Aside from the limitations discussed above, almost all combinations of embeddings and clustering algorithms generate meaningful results, and representation learning is clearly a good approach to handle

the complexity of UI logs. Isomap and the Autoencoder create embeddings from the same input vectors. Because the Autoencoder performs slightly better and is more computationally efficient, we conclude that it is the better option to learn trace representations. Judging by silhouette scores and t-SNE visualizations, Word2Vec appears to perform slightly worse than the alternatives. However, both silhouette scores and the t-SNE embedding are calculated from an indicator matrix with Jaccard distance. This means that they do not consider trace semantics, which causes a discrepancy between the objective function of the algorithm and the evaluation methods. We therefore only conclude that a representation learning technique that encodes the semantics of traces is also suitable to embed traces in a UI log, and that it leads to slightly different clustering results. As evident in the Rand similarity matrix for the interaction perspective, the difference becomes larger if navigation events are included, likely because they introduce significantly more control-flow variation. We expect that the choice of an adequate embedding technique for a UBM application will depend on the specific use case.

K-means clustering and AHC score similarly in all quality measures and have high allocation agreement for all three embeddings, indicating that the choice between these two algorithms has little influence on the results, even when the data is complex and many clusters exist. Because we used them to a slightly different avail, we will not compare these two algorithms to DBSCAN. However, we do observe that DBSCAN can be an effective tool for exploratory trace clustering in UI logs if it is applied to a trace representation without semantics, but fails to find good clusters if semantics are considered.

There are several factors that may affect the clustering results, including the types of tasks recorded, the case- and event-level attributes considered, and the design of the user interface and the underlying information system. Because we have only applied trace clustering techniques to one dataset, we cannot be certain that our findings can be generalized to other UI logs. However, our data consists of real executions of various processes and tasks from a standard, widely used ERP system, so we would expect that clustering traces in other UI logs that record typical user interactions in ERP systems would yield similar results. Further research would be required in order to determine if our findings can be generalized beyond that, for example to a web context.

7 Conclusion

In this paper, we have explored the applicability of process trace clustering in User Behavior Mining (UBM) through an experimental evaluation on an event log that records low-level UI interactions in an ERP system. We find that trace clustering can be used to identify meaningful groups of user input activities in a UI log, but that it produces unsatisfactory results when also incorporating user navigation paths. Consequently, the process models mined from clusters are only well comprehensible in some cases. Our results also indicate that recently proposed neural-network-based representation learning techniques can effectively encode complex process traces in real-life event logs.

With more organizations looking into data-driven methods to optimize software UIs and to streamline and automate processes, UBM is likely to gain relevance in the near future. UI logs are typically more complex than conventional event logs, and enhancing them with some structure is a prerequisite for effectively analyzing user behavior. Our research demonstrates that it is feasible to use trace clustering for this purpose, but it also shows that even some of the most powerful techniques fall short of finding good partitions in every case. We have encountered two specific challenges that should be addressed by further research: clustering a complete UI log that includes user navigation paths, and incorporating case-level context while still separating traces by control-flow behavior. It remains to be seen if these can be solved by modifying existing techniques, or if UBM requires an entirely different approach. It may also be interesting to explore the effect of incorporating other (event-level) attributes, such as timestamps. Furthermore, it may be possible to find better partitions with more comprehensible process models by considering other clustering objectives, such as minimizing a model complexity measure. Finally, clustering other UI logs and comparing the results to our study would shed light on the generalizability of our findings and lead to a more comprehensive understanding of how trace clustering can benefit UBM.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*.

Amoako-Gyampah, K. (2007). "Perceived usefulness, user involvement and behavioral intention: an empirical study of ERP implementation." *Computers in Human Behavior* 23 (3), 1232–1248.

Astromskis, S., Janes, A., and Mairegger, M. (2015). "A process mining approach to measure how users interact with software: An industrial case study." In: *Proceedings of the 2015 International Conference on Software and System Process*. New York: Association for Computing Machinery, pp. 137–141.

Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F. M., Marrella, A., Mecella, M., and Soo, A. (2018). "Automated discovery of process models from event logs: Review and benchmark." *IEEE Transactions on Knowledge and Data Engineering* 31 (4), 686–705.

Barbon, S., Ceravolo, P., Damiani, E., and Tavares, G. (2021). *Selecting Optimal Trace Clustering Pipelines with AutoML*. arXiv preprint: `2109.00635`.

Beynon-Davies, P. (2019). *Business information systems*. London: Red Globe Press.

Bose, R. P. C. and van der Aalst, W. (2010). "Trace alignment in process mining: Opportunities for process diagnostics." In: *Business Process Management*. Cham: Springer, pp. 227–242.

Bose, R. P. C. and van der Aalst, W. (2009). "Context aware trace clustering: Towards improving process mining results." *Proceedings of the SIAM International Conference on Data Mining*, 401–412.

Bui, H.-N., Vu, T.-S., Nguyen, H.-H., Nguyen, T.-T., and Ha, Q.-T. (2020). "Exploiting CBOW and LSTM models to generate trace representation for process mining." In: *Intelligent Information and Database Systems*. Cham: Springer, pp. 35–46.

Buijs, J., van Dongen, B., and van der Aalst, W. (2014). "Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity." *International Journal of Cooperative Information Systems* 23 (1).

Burton-Jones, A. and Straub, D. (2006). "Reconceptualizing system usage: An approach and empirical test." *Information Systems Research* 17 (3), 228–246.

Choi, H. and Choi, S. (2007). "Robust kernel Isomap." *Pattern Recognition* 40 (3), 853–862.

Davis, F. D. (1989). "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology." *MIS Quarterly* 13 (3), 319–340.

De Koninck, P., Vanden Broucke, S., and De Weerdt, J. (2018). "act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes." In: *Business Process Management*. Cham: Springer, pp. 305–321.

de Leoni, M. and Dündar, S. (2020). "Event-Log Abstraction Using Batch Session Identification and Clustering." In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. New York: Association for Computing Machinery, pp. 36–44.

De Weerdt, J., Vanden Broucke, S., Vanthienen, J., and Baesens, B. (2013). "Active trace clustering for improved process discovery." *IEEE Transactions on Knowledge and Data Engineering* 25 (12), 2708–2720.

Dees, M. and van Dongen, B. (2016). *BPI Challenge 2016*. URL: `https://data.4tu.nl/articles/dataset/BPI_Challenge_2016_Clicks_Logged_In/12674816/1`.

Dev, H. and Liu, Z. (2017). "Identifying Frequent User Tasks from Application Logs." In: *International Conference on Intelligent User Interfaces*. New York: Association for Computing Machinery, pp. 263–273.

Ding, A. W., Li, S., and Chatterjee, P. (2015). "Learning User Real-Time Intent for Optimal Dynamic Web Page Transformation." *Information Systems Research* 26 (2), 339–359.

Facca, F. M. and Lanzi, P. L. (2005). "Mining interesting knowledge from weblogs: a survey." *Data & Knowledge Engineering* 53 (3), 225–241.

Feldman, M. (2016). "Routines as processes: Past, Present, and future." In: *Organizational Routines: How They Are Created, Maintained, and Changed*. Ed. by J. Howard-Grenville, C. Rerup, A. Langly, and H. Tsoukas. Routledge Oxford, pp. 23–46.

Ferreira, D., Zacarias, M., Malheiros, M., and Ferreira, P. (2007). "Approaching process mining with sequence clustering: Experiments and findings." In: *Business Process Management*. Cham: Springer, pp. 360–374.

Greco, G., Guzzo, A., Pontieri, L., and Sacca, D. (2006). "Discovering expressive process models by clustering log traces." *IEEE Transactions on Knowledge and Data Engineering* 18 (8), 1010–1027.

Grisold, T., Wurm, B., Mendling, J., and Vom Brocke, J. (2020). "Using process mining to support theorizing about change in organizations." In: *Hawaii International Conference on System Sciences*.

Guzzo, A., Joaristi, M., Rullo, A., and Serra, E. (2021). "A multi-perspective approach for the analysis of complex business processes behavior." *Expert Systems with Applications* 177.

Hinton, G. and Salakhutdinov, R. R. (2006). "Reducing the Dimensionality of Data with Neural Networks." *Science* 313 (5786), 504–507.

Ho, S., Bodoff, D., and Tam, K. (2010). "Timing of Adaptive Web Personalization and Its Effects on Online Consumer Behavior." *Information Systems Research* 22 (3), 660–679.

Hoffmann, P., Mateja, D., Spohrer, K., and Heinzl, A. (2019). "Bridging the Vendor-User Gap in Enterprise Cloud Software Development through Data-Driven Requirements Engineering." In: *International Conference on Information Systems*. Atlanta: Association for Information Systems.

Hubert, L. and Arabie, P. (1985). "Comparing partitions." *Journal of Classification* 2 (1), 193–218.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs: Prentice-Hall.

Jimenez-Ramirez, A., Reijers, H. A., Barba, I., and Del Valle, C. (2019). "A method to improve the early stages of the robotic process automation lifecycle." In: *Advanced Information Systems Engineering*. Cham: Springer, pp. 446–461.

Kruskal, J. B. (1964). "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis." *Psychometrika* 29, 1–27.

Lambeck, C., Müller, R., Fohrholz, C., and Leyh, C. (2014). "(Re-)Evaluating user interface aspects in ERP systems – an empirical user study." In: *47th Hawaii International Conference on System Sciences*, pp. 396–405.

Leno, V., Augusto, A., Dumas, M., La Rosa, M., Maggi, F. M., and Polyvyanyy, A. (2020). "Identifying candidate routines for robotic process automation from unsegmented UI logs." In: *Proceedings of the 2nd International Conference on Process Mining*, pp. 153–160.

Leno, V., Polyvyanyy, A., Dumas, M., Rosa, M. L., and Maggi, F. M. (2021). "Robotic process mining: Vision and challenges." *Business & Information Systems Engineering* 63, 301–314.

Linn, C., Zimmermann, P., and Werth, D. (2018). "Desktop Activity Mining - A new level of detail in mining business processes." In: *Workshops der INFORMATIK 2018*. Bonn: Köllen Druck+Verlag, pp. 245–258.

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: *Business Process Management*. Cham: Springer, pp. 198–215.

Luettgen, S., Seeliger, A., Nolle, T., and Mühlhäuser, M. (2021). "Case2vec: Advances in Representation Learning for Business Processes." In: *Process Mining Workshops, ICPM 2020*. Cham: Springer, pp. 162–174.

Maalej, W., Nayebi, M., Johann, T., and Ruhe, G. (2016). "Toward data-driven requirements engineering." *IEEE Software* 33 (1), 48–54.

Maruster, L., Faber, N., Jorna, R., and van Haren, R. (2008). "A process mining approach to analyse user behaviour." In: *Proceedings of the 4th International Conference on Web Information Systems and Technologies*. Vol. 2, pp. 208–214.

Mikolov, T., Corrado, G., Chen, K., and Dean, J. (2013). “Efficient estimation of word representations in vector space.” In: *Proceedings of the 2013 International Conference on Learning Representations*, pp. 1–12.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). “Distributed representations of words and phrases and their compositionality.” In: *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Vol. 2, pp. 3111–3119.

Nielsen, J. (1994). *Usability engineering*. Burlington: Morgan Kaufmann.

Parks, N. E. (2012). “Testing & quantifying ERP usability.” In: *Proceedings of the 1st Annual Conference on Research in Information Technology*. Calgary: Association for Computing Machinery, pp. 31–36.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12, 2825–2830.

Pentland, B., Vaast, E., and Wolf, J. (2021). “Theorizing process dynamics with directed graphs: A diachronic analysis of digital trace data.” *MIS Quarterly* 45 (2).

Poggi, N., Muthusamy, V., Carrera, D., and Khalaf, R. (2013). “Business process mining from e-commerce web logs.” In: *Business Process Management*. Cham: Springer, pp. 65–80.

Rand, W. M. (1971). “Objective criteria for the evaluation of clustering methods.” *Journal of the American Statistical Association* 66 (336), 846–850.

Raphaeli, O., Goldstein, A., and Fink, L. (2017). “Analyzing online consumer behavior in mobile and PC devices: A novel web usage mining approach.” *Electronic Commerce Research and Applications* 26, 1–12.

Rousseeuw, P. J. (1987). “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.” *Journal of Computational and Applied Mathematics* 20, 53–65.

Rubin, V. A., Mitsyuk, A. A., Lomazova, I. A., and van der Aalst, W. (2014). “Process mining can be applied to software too!” In: *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. New York: Association for Computing Machinery.

Scholtz, B., Cilliers, C., and Calitz, A. (2010). “Qualitative techniques for evaluating enterprise resource planning (ERP) user interfaces.” In: *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*. Bela Bela: Association for Computing Machinery, pp. 284–293.

Seeliger, A., Luettgen, S., Nolle, T., and Mühlhäuser, M. (2021). “Learning of process representations using recurrent neural networks.” In: *Advanced Information Systems Engineering*. Cham: Springer, pp. 109–124.

Song, J., Luo, T., and Chen, S. (2008). “Behavior pattern mining: Apply process mining technology to common event logs of information systems.” In: *IEEE International Conference on Networking, Sensing and Control*, pp. 1800–1805.

Song, M., Günther, C. W., and van der Aalst, W. (2009). “Trace Clustering in Process Mining".” In: *Business Process Management Workshops, BPM 2009*. Cham: Springer, pp. 109–120.

Song, M., Yang, H., Siadat, S. H., and Pechenizkiy, M. (2013). “A comparative study of dimensionality reduction techniques to enhance trace clustering performances.” *Expert Systems with Applications* 40 (9), 3722–3737.

Srivastava, J., Cooley, R., Deshpande, M., and Tan, P.-N. (2000). “Web usage mining: Discovery and applications of usage patterns from web data.” *SIGKDD Explorations* 1, 12–23.

Tenenbaum, J. B., Silva, V. d., and Langford, J. C. (2000). “A global geometric framework for nonlinear dimensionality reduction.” *Science* 290 (5500), 2319–2323.

Thaler, T., Ternis, S. F., Fettke, P., and Loos, P. (2015). “A comparative analysis of process instance cluster techniques.” In: *Proceedings of the 12th International Conference on Wirtschaftsinformatik*. Osnabrück: Universität Osnabrück.

Topi, H., Lucas, W., and Babaian, T. (2005). “Identifying usability issues with an ERP implementation.” In: *Proceedings of the 2005 International Conference on Enterprise Information Systems*, pp. 128–133.

van der Aalst, W. (2011). “Process mining: Discovering and improving spaghetti and lasagna processes.” In: *2011 IEEE Symposium on Computational Intelligence and Data Mining*, pp. 1–7.

van der Aalst, W., Bichler, M., and Heinzl, A. (2018). “Robotic process automation.” *Business & Information Systems Engineering* 60 (4), 269–272.

van der Maaten, L. and Hinton, G. (2008). “Visualizing data using t-SNE.” *Journal of Machine Learning Research* 9 (86), 2579–2605.

van Zelst, S., Mannhardt, F., de Leoni, M., and Koschmider, A. (2021). “Event Abstraction in Process Mining -Literature Review and Taxonomy.” *Granular Computing* 6, 719–736.

Veiga, G. M. and Ferreira, D. (2010). “Understanding spaghetti models with sequence clustering for ProM.” In: *Business Process Management Workshops, BPM 2009*. Cham: Springer, pp. 92–103.

Wang, G., Zhang, X., Tang, S., Wilson, C., Zheng, H., and Zhao, B. Y. (2017). “Clickstream user behavior models.” *ACM Transactions on the Web* 11 (4), 1–37.

Weinzierl, S., Stierle, M., Zilker, S., and Matzner, M. (2020). “A Next Click Recommender System for Web-based Service Analytics with Context-aware LSTMs.” In: *Proceedings of the 53rd Hawaii International Conference on System Sciences*. AISeL.

Wurm, B., Grisold, T., Mendling, J., and Vom Brocke, J. (2021). “Measuring Fluctuations of Complexity in Organizational Routines.” In: *Academy of Management Proceedings*. Vol. 2021. 1.

Xu, D. and Tian, Y. (2015). “A comprehensive survey of clustering algorithms.” *Annals of Data Science* 2 (2), 165–193.

Zandkarimi, F., Rehse, J. R., Soudmand, P., and Hoehle, H. (2020). “A generic framework for trace clustering in process mining.” *Proceedings of the 2nd International Conference on Process Mining*, 177–184.