Selecting Optimal Trace Clustering Pipelines with Meta-learning

Gabriel Marques Tavares[1(✉)], Sylvio Barbon Junior[2], Ernesto Damiani[3], and Paolo Ceravolo[1]

[1] Università degli Studi di Milano (UNIMI), Milan, Italy
{gabriel.tavares,paolo.ceravolo}@unimi.it
[2] Università degli Studi di Trieste (UniTS), Trieste, Italy
sylvio.barbonjunior@units.it
[3] Khalifa University (KUST), Abu Dhabi, UAE
ernesto.damiani@kustar.ac.ae

Abstract. Trace clustering has been extensively used to discover aspects of the data from event logs. Process Mining techniques guide the identification of sub-logs by grouping traces with similar behaviors, producing more understandable models and improving conformance indicators. Nevertheless, little attention has been posed to the relationship among event log properties, the pipeline of encoding and clustering algorithms, and the quality of the obtained outcome. The present study contributes to the understanding of the aforementioned relationships and provides an automatic selection of a proper combination of algorithms for clustering a given event log. We propose a Meta-Learning framework to recommend the most suitable pipeline for trace clustering, which encompasses the encoding method, clustering algorithm, and its hyperparameters. Our experiments were conducted using a thousand event logs, four encoding techniques, and three clustering methods. Results indicate that our framework sheds light on the trace clustering problem and can assist users in choosing the best pipeline considering their environment.

Keywords: Process mining · Trace clustering · Meta-learning · Recommendation · Pipeline design

1 Introduction

Executing business processes leaves trails of the accomplished activities, performances achieved, and resources consumed. This information is stored in event logs, which embrace the history of the process. Executions generating the same sequence of activities are observed as the same trace by Process Mining (PM) algorithms that can group multiple executions in a single representation. Often, the variability of traces is however remarkable, and traces by themselves do not offer a helpful representation of the process. This variability causes problems for existing PM techniques. For instance, business processes with high trace variability generate spaghetti-like models, i.e., complex models with an enormous number of relations, often unreadable [1]. Neubauer et al. [24] identified two elements that contribute primarily to the inherent complexity of business processes: (i) knowledge-intensive processes where decision-making is human-dependent, and (ii) processes from large organizations with a fast generation rate, and therefore high volume output. Therefore, it is of interest to simplify the analysis representation, thus, allowing an easier interpretation for stakeholders and leveraging efficiency. For instance, consider an event log and its sequences of activities (traces) $L = \{\langle a, b, c, d, e\rangle, \langle a, d, c, e\rangle, \langle a, b, c\rangle, \langle a, d, c\rangle\}$, it is possible to notice two groups of closely related traces, i.e., trace $\langle a, b, c, d, e\rangle$ is similar to $\langle a, b, c\rangle$ whereas trace $\langle a, d, c\rangle$ has a sequence closer to trace $\langle a, d, c, e\rangle$. Grouping these similar traces may improve the accuracy and comprehensibility of process discovery techniques [11], and at the same time, support the identification of deviating or anomalous instances [17]. Moreover, concurrency might also be a problem in some domains. For instance, traces $\langle a, b, c, e\rangle$ and $\langle a, c, b, e\rangle$ may be considered the same from a business perspective if the order of activities b and c do not affect the process outcome. This way, these trace representations should be close when projected into the feature space.

Trace clustering techniques have been adopted to solve these issues by identifying sub-logs grouped by trace similarity. This way, by detecting groups with homogeneous behavior, process discovery techniques can be executed in sub-logs, producing higher quality models, which are instead accessible for stakeholders [14]. Trace clustering has also been studied in the context of explainability for PM [20] and, more recently, adapted to incorporate expert knowledge [19]. However, selecting the appropriate clustering technique is a complex task. Many transformation methods were presented, treating traces as vectors generated from bags of activities [22], edit distance [4] or dependency spaces [12], discriminant rules [15,26] or log footprints [20]. The set of clustering algorithms applied is also ample, e.g., k-means [15], hierarchical clustering [4], spectral clustering [12], constrained clustering [19], among others. Given this large set of options to set up a clustering pipeline, a non-expert user can likely feel overwhelmed.

Considering the challenge of designing pipelines to identify the correct encoding method, clustering algorithms, and hyperparameters to use for a specific log, we propose a framework based on Meta-learning (MtL). Our framework recommends the trace clustering pipeline that best fits a specific event log. MtL is a learning process applied to meta-data representing other learning processes [31] and has been used successfully to emulate experts' recommendations, maximize performance, and improve quality metrics [16].

The problem of simultaneously recommending an algorithm and tuning its hyperparameters to optimize a task is defined as the combined algorithm selection and hyperparameter optimization problem [28]. Alternatively, it is possible to exploit similar recommending tasks, in which algorithms and hyperparameters are represented as discrete spaces, mapping possible inter-correlations between the different hyperparameters as a multi-output machine learning problem. In this work, the meta-data consists of a large set of event log features that are provided as input to the MtL workflow that outputs trace clustering pipelines described by an encoding technique, a clustering algorithm, and hyperparameters modeled using a problem transformation approach. In our scenario, MtL serves as an automated approach as it suppresses the need for expert interaction to work correctly. The relationship between event log features and the quality of PM techniques has been already pointed out in the literature [2,3]. We introduce a general framework for studying this relationship for the trace clustering task using MtL. Moreover, we instantiate this framework to provide an example of its functionality. In particular, in our experiments, we submit the method to a set of 1091 event logs described by 93 log features, four encoding techniques, and three clustering algorithms. Results show that our approach achieves considerable performance for recommending encoding and clustering techniques. We also provide a comparison with two baseline methods, highlighting the improvement supported by the MtL strategy.

The remainder of this paper is organized as follows. Section 2 gives a historical overview of trace clustering solutions, focusing on the employed transformation and clustering methods. Section 3 defines the task and its configuration steps, while Sect. 4 presents our proposed framework to solve the trace clustering recommendation problem. Section 5 presents the material used for experiments, the techniques, and quality metrics adopted. Section 6 shows the results and raises a discussion around them. Section 7 concludes the paper and Sect. 8 lists its broader impact.

2 Related Work

Trace clustering research is deeply connected to the variant analysis problem, that is, detecting groups of similar behavior within a single business process [20]. Clustering traces is partitioning an event log into groups of comparable traces such that each trace is assigned to a unique group [19]. Since its initial adoption, trace clustering has been proposed as an instrument to reduce variability. Discovering process models from clusters, for example, generally improves quality [14]. An early work in the area used a set of n-grams to encode a trace activity sequence, thus, mapping traces to a feature vector space [15]. Song et al. [26] went further by defining multiple encoding procedures, named profiles, to represent traces as vectors. Furthermore, the authors call attention to the modularity between the profiling and clustering steps. Bose and Aalst [4] represent traces as strings and apply edit distance to measure trace similarity. Delias et al. [12] proposed a measure to calculate trace distance based on dependency. However, approaches based on instance-level similarity may be applicable only to particular domains. Thaler et al. [27] highlight that bags of activities may lose critical information regarding the execution order. Delias et al. [12] show that no single optimal similarity metric is applicable for all domains and applications while Zandkarimi et al. [34] stated that trace clustering is a context-specific task. Koninck et al. [20] characterize the complexity of clustering with the assessment of the best event log splitting operations. A well-performing encoding method improves a wide range of posterior analyses without the need to tune them [3]. The authors also showed that there is no best encoding method for every scenario, that is, different event logs are encoded better, considering several quality criteria, by different encoding techniques. A similar conclusion is achieved in [27] when analyzing clustering algorithms applied to PM.

The authors stated that some techniques are suitable for particular scenarios, reinforcing the argument that process characteristics may guide the decision of the appropriate clustering technique. Besides, different from supervised approaches, unsupervised learning performance is severely affected by small changes in hyperparameters, depending heavily on user-domain knowledge [18]. This implies that the solutions proposed today are far from optimal as they are attached to a unique set of encoding and clustering algorithms.

Considering the multiple available profiling and clustering algorithms, we envision two main building blocks regulating the success of clustering techniques. The first regards the encoding method, i.e., converting the trace sequences into feature vectors, and the latter comprises the clustering techniques. The approaches currently available in the literature are strictly attached to a specific combination of these building blocks; hence, they neither offer a means to study the relationship between the different steps that compose a pipeline nor relate process behavior to optimal solutions.

3 Problem Statement

Given the plethora of configuration steps and parametrization, designing the appropriate trace clustering pipeline is a complex issue even for experts. We identified in the literature three configuration steps that highly affect the clustering results: (i) trace encoding, (ii) clustering algorithm, and (iii) hyperparameters regulating the clustering algorithm. The choice of each step is critical since slight changes deeply affect the clustering results.

PM techniques ingest event logs. An *event log* is the set of events executed in a business process. An *event* records the execution of an *activity*. It follows that each event is strictly related to a unique process instance, identified by its *case*. A unique end to end sequence of activities within a case is known as a *trace*. Let Σ be the *event universe*, i.e. the set of all possible event identifiers; Σ^* denotes the set of all sequences over Σ. A *trace* is a non-empty sequence of events $t \in \Sigma^*$ where each event appears only once and time is non-decreasing, i.e., for $1 \leq i < j \leq |t| : t(i) \neq t(j)$. In PM applications, encoding aims at transforming traces into vectors, mapping process instances into a feature space. Therefore, an *encoding* method is a function $E^{()}$ that maps a set of traces into a n-dimensional feature space, projecting the instances' distances according to their trace sequence.

The problem of selecting a trace clustering pipeline is different from the traditional algorithm selection, in which it is expected to recommend a tuple $\langle$*encoding*, *clustering*, *hyperparameters*$\rangle$. It is worth mentioning that the hyperparameters are continuous values with a high dimensional space that might present different inter-correlations between them. Here, they were discretized at frequent intervals to cover a wide range of promising possibilities. Further, we employed a multi-output strategy to take advantage of inter-correlations from the clustering algorithm and its hyperparameters.

We formulate the problem as a set of encoding methods $\mathcal{E} = \{E^{(1)}, ..., E^{(j)}\}$, clustering algorithms $\mathcal{C} = \{C^{(1)}, ..., C^{(l)}\}$ associated with a hyperparameter space $\mathcal{H} = \{H^{(1)}, ..., H^{(l)}\}$ and event log data mapped by meta-features and best pipeline as $\mathcal{D} = \{(x^{(i)}, Y^{(s)} | s = 2)\}$. We have j encoding methods, l combinations of clustering algorithms and hyperparameters, and s is the number of expected pipelines' steps to be recommended. It is important to mention that the clustering algorithm and hyperparameter recommendation were modeled as a single step of the pipeline. For each event log sample $(x^{(i)}, Y^{(i)})$, $x^{(i)} \in \mathcal{X}$ is a d-dimensional meta-feature vector $(x^{(i1)}, x^{(i2)}, ..., x^{(id)})$ and $Y^{(i)} \subseteq \mathcal{Y}$ is the tuple $\langle$*encoding*, *clustering*, *hyperparameters*$\rangle$ associated to $x^{(i)}$. The goal is that for any unseen event log x, the MtL model $h_E()$ recommends $h_E(x) \in \mathcal{E}$ as the proper encoding method for x and $h_{CH}()$ recommends $h_{CH}(x) \in \mathcal{L}$. In the proposed setup, we are facing a *multi-output* problem, where a set of labels $\mathcal{C} \times \mathcal{H} \subseteq L$ is associated with a single instance [29]. Following the taxonomy proposed in [29], we adopt a problem transformation approach, which converts the data into a format that can be used in conjunction with traditional techniques. More specifically, we employed the Binary Relevance (BR) problem transformation approach [33]. BR works by transforming the original data set into q data sets D_{λ_j}, where $j = [1, ..., q]$ contains all instances of the original data that are labeled according to the existence or not of single labels λ_j. Thus, BR learns q binary classifiers, one for each label L. Given a new instance, BR provides the union of the labels λ_j predicted by the q classifiers.

There are several ways to model this problem. In this paper, we followed the supervised machine learning approach to build $h_E()$ and $h_{CH}()$ towards determining a promising pipeline candidate configuration. The problem is, in nature, a multi-output problem. Therefore, we model this through the BR approach to combine outputs from both $h_E()$ and $h_{CH}()$. Alternative optimization-based modeling methods to control the trade-off between exploitation versus exploration of pipeline combinations exists, but as an initial study exploring new meta-features and meta-target selection in a new application on the PM domain, we adopted this modeling strategy for simplicity.

4 MtL-Based Solution for Trace Clustering

Trace clustering solutions must be able to adapt according to domain characteristics. We then propose a framework grounded in MtL capable of delivering suitable recommendations according to different business process behaviors. The main goal of our approach is to recommend a tuple $\langle$*encoding*, *clustering*, *hyperparameters*$\rangle$ that maximizes quality metrics for the trace clustering problem. Figure 1 shows the overview of the framework. First, an event log repository is created to represent different business scenarios. The *Meta-Feature Extraction* step mines features for each event log in the repository, creating *meta-features* according to MtL terminology. The description quality of the *meta-features* is an important constraint bounding the performance of the complete pipeline. Moreover, the *Meta-Target Definition* defines a set of encoding and clustering (coupled with its hyperparameter) techniques that are assessed by quality metrics and ranked according to a ranking function. Then, the *Meta-Database* combines the *meta-features* and *meta-targets* defined in previous steps, creating a data set populated by *meta-instances*. Using the *meta-database*, the *Meta-learning* step induces a *Meta-model* that is, then, used to recommend a pipeline for a given event log considering its *meta-features*. It is worth mentioning that multi-output machine learning modeling for the meta-model can bring important achievements in terms of performance, considering the interrelations between each step of the pipeline. In Fig. 1, green arrows indicate the steps that are used for the creation and training of the framework, while blue arrows represent a production environment where one assesses the meta-model for recommendation.

Given the adaptable setup of our framework, one can implement it using a different set of *meta-features* and *meta-targets*. The automatic aspect of this approach provides the user with recommendations based on event log behavior, considering the possible options among the configurable steps. Moreover, other aspects are adaptable, such as the adopted quality metrics and the ranking function. Nonetheless, we note that the robustness of the approach depends on the MtL structure, which must be maintained when the framework is instantiated in real scenarios.

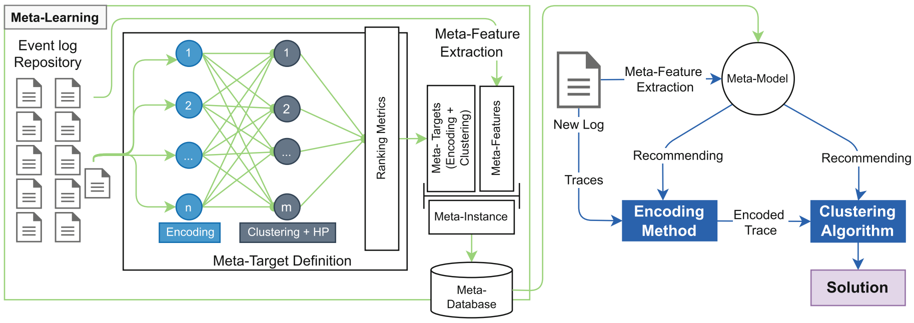


Fig. 1. Overview of MtL proposal for trace clustering.

5 Experimental Setup

In this section, we expose the details of each framework step, as seen in Fig. 1, and reveal the experiments implemented to study a possible instance of our MtL framework. The implementation is available for replication purposes[1].

[1] https://github.com/gbrltv/meta_trace_clustering/.

5.1 Event Logs and Featurization

MtL benefits from using a large set of instances in the meta-database. Hence, we are aiming at a heterogeneous set of business process logs representing different scenarios and behaviors. For that, we rely on the set of logs proposed in [2]. These event logs were grouped to represent a plethora of business behaviors, mapping the relationship between process characteristics and quality metrics. This set contains both real and synthetic event logs. Regarding real-life data, there are six logs from past Business Process Intelligence Challenges (BPIC)[2], the environmental permit[3], helpdesk[4] and sepsis[5] logs. For synthetic data, the authors adopted 192 logs from the Process Discovery Contest (PDC) 2020[6], an annual event organized to evaluate the efficiency of process discovery algorithms. The PDC logs are complex given the nature of employed behaviors, such as dependent tasks, loops, invisible and duplicate tasks, and noise. The next group of synthetic data contains 750 logs proposed in the context of online PM [7]. These logs are built to depict process drifts, i.e., behavior change during the business process execution, containing four drift types, five noise percentages, and three trace lengths. The final group of synthetic event logs was proposed for the evaluation of trace encoding techniques [3]. This set contains 140 logs generated from five process models, six anomaly types, and four frequency percentages.

The performance of the meta-model is directly dependent on the quality of the meta-features. Thus, the meta-features extracted from event logs must capture the process behavior and describe it from complementary perspectives. We adopted the featurization introduced in [2]. The authors presented a group of features that capture several layers of business processes. These features are based on the distribution of trace behavior, considering trace length, activity frequencies, and trace variants. Regarding activity-level features, the group is subdivided into all activities, start activities, and end activities. 12 features are extracted for each group, they are the number of activities, minimum, maximum, mean, median, standard deviation, variance, the 25th and 75th percentile of data, interquartile range, skewness, and kurtosis coefficients. To capture the behavior at the trace level, the authors propose features for trace lengths and trace variants. The former group contains 29 attributes: minimum, maximum, mean, median, mode, standard deviation, variance, the 25th and 75th percentile of data, interquartile range, geometric mean and standard variation, harmonic mean, coefficient of variation, entropy, and a histogram of 10 bins along with its skewness and kurtosis coefficients. Trace variants are captured by 11 descriptors: mean number of traces per variant, standard variation, skewness coefficient, kurtosis coefficient, the ratio of the most common variant to the number of traces, and ratios of the top 1%, 5%, 10%, 20%, 50% and 75% variants to the total number of traces. Log-level behavior is captured by: number of traces, unique

[2] https://www.tf-pm.org/resources/logs.

[3] https://doi.org/10.4121/uuid:26aba40d-8b2d-435b-b5af-6d4bfbd7a270.

[4] https://doi.org/10.17632/39bp3vv62t.1.

[5] https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

[6] https://doi.org/10.4121/14626020.v1.

traces, their ratio, and number of events. To describe log complexity, entropy-based measures have been adopted recently in PM literature [1] aiming at the discretization between logs that are better mined by declarative or imperative algorithms. Hence, such metrics capture the structuredness and variability of the log. The 14 entropy features we adopt are: trace, prefix, k-block difference and ratio (k values of 1, 3 and 5), global block, k-nearest neighbor (k values of 3, 5, and 7), Lempel-Ziv, and Kozachenko-Leonenko. Considering all groups, 93 meta-features were used to extract log behavior covering log structuredness and variability, statistical dispersion, probability distribution shape, and tendency.

5.2 Trace Encoding Techniques

Many PM techniques rely on encoding to transform event log-specific representations to other formats [8,25,30]. The transformation usually applies at the trace-level, that is, converting the sequence of activities respective to a unique trace into a feature vector. In [3], the authors compared ten different encoding techniques through the lens of quality metrics measuring data dispersity, representativeness, and compactedness. They concluded that there is no encoding that excels in all tasks and perspectives concomitantly. For instance, graph embeddings outperform the others in the classification task and representation quality. However, these encoding methods are costly and usually sparse, meaning that there are better encoding techniques considering space and time complexity. The trace clustering literature has already experimented with several types of encoding methods, such as one-hot encoding [15,26], edit distance [4], log footprints [20], activity profiles and n-grams [9]. Nonetheless, no trace similarity measure is general enough to be applicable in all scenarios [10].

In this work, we adopt four encoding techniques that were frequently applied in the context of trace clustering. The first one is one-hot encoding. This technique encodes activities as categorical dimensions, creating a feature vector of binary values for each trace based on the occurrence of activities in a trace. Next, we adopt n-grams, a common technique used in text mining applications. This encoding maps groups of activities of size n into a feature vector, accounting for their occurrence or not. More specifically, we apply bi-gram and tri-gram. Finally, we applied position profiles [6], an approach that relates activity frequency and position. A log profile is created by computing the activity appearances in each trace position and its respective frequency. A trace is encoded considering the frequency of its activities in their positions according to the log profile.

5.3 Trace Clustering Algorithms

We selected three clustering techniques commonly applied in data mining and trace clustering literature. These techniques are grounded in different heuristics, and with this, we aim to evaluate if a particular clustering structure outperforms the others. The choice of parameters was also guided by considering the literature on trace clustering, comprising different trace behaviors and complexities. It is important to note that we selected a range of possible values to support the exploration of the algorithmic space.

First, we adopt the Density-based Spatial Clustering of Applications with Noise (*dbscan*) algorithm [13]. The *dbscan* method guides its clustering based on the density of the feature space, hence, instances in high-density regions form a cluster while instances sitting at low-density regions are regarded as outliers. The main hyperparameter affecting the clustering results is *eps*, which regulates the maximum distance between two points for them to be considered of the same neighborhood. We explore different configurations of the *eps* hyperparameter to evaluate its impact and to recommend the best configuration in the meta-model step. For that, we apply the following *eps* values: 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1. Moreover, we adopt k-means [21], a clustering technique that randomly selects centroids, which are the initial cluster points, and works by iteratively optimizing the centroid positions. The k-means technique requires the expected number of clusters (k) from a given data set as a hyperparameter. We set k to these values: 2, 3, 4, 5, 6, 7, 8, 9, 10. Finally, the last technique is *agglomerative* clustering [32], a type of hierarchical clustering with a bottom-up approach. The algorithm starts by considering each point as a cluster and merges the clusters as the hierarchy moves up, creating a tree-like structure depicting the cluster levels and merges. As with k-means, *agglomerative* clustering requires the number of clusters as input, we then adopted the same range of values for the k parameter.

5.4 Ranking Metrics

To complete the creation of a meta-database, meta-targets must be defined for each meta-instance. This way, a ranking strategy is required to compare the possible trace clustering pipelines. Hence, the technique sitting at the top of the ranking strategy is the one recommended for a meta-instance, i.e., it is defined as the meta-target. As pointed out in the literature [3,10], there is no unique solution for a problem that outperforms the others from all perspectives. Considering this hypothesis, we propose three complementary metrics to evaluate trace clustering solutions, this way, capturing different degrees of performance. Moreover, a user applying a trace clustering solution may expect to evaluate the results from several perspectives. Here, we support such a user by assessing clustering quality from a set of criteria.

Silhouette coefficient (s), the first metric we propose to measure performance, is based on the traditional clustering literature. The Silhouette score is computed at the cluster level to capture its tightness and separation, judging instances that fit their cluster or are in between different clusters. The scores of a group of clusters can be combined to assess the relative quality of the clustering technique.

To complement this evaluation with a PM-inspired metric, we propose to measure the quality of clusters concerning trace variants. This way, by computing the trace variant frequency in each cluster, we can evaluate if the solution provides a clear separation of variants in the feature space. For that, we compute the unique traces in a cluster, and by

$$v = \frac{\sum_{C_i \in C} var(C_i) - 1}{\#traces} \tag{1}$$

a weighted mean, the Variant score (v) is obtained. Consider C the group of all clusters, C_i the cluster of index i, $var(C_i)$ the number of unique traces found in cluster C_i and $\#traces$ the total number of traces in the event log, Eq. 1 depicts the Variant score calculation, 0 is the optimal value. As resource consumption is an important aspect in organizations, we also consider the clustering time (t) as a metric to assess its quality. The lower the t metric for a particular solution, the better it is ranked compared to others.

Table 1. Ranking trace clustering pipelines.

Log	Encoding	Clustering	s	v	t	R_s	R_v	R_t	R
L	E_1	C_1	0.9	0.5	50	1	2	3	2
L	E_2	C_2	0.3	0	10	3	1	1	1.67
L	E_3	C_3	0.8	0.7	15	2	3	2	2.33

Given this set of metrics, i.e. s for silhouette coefficient, v for the variant score, and t for computational time, a *meta-target* $\langle$*encoding*, *clustering*, *hyperparameters*$\rangle$ has to successfully balance between all metrics to be considered good. This way, the approach rewards techniques that excel in the three metrics, such as ignoring one or more may lead to a lack of tightness, improper variant identification, and high resource consumption. Hence, we propose a ranking strategy (R) that combines all dimensions. Table 1 presents an example of the ranking strategy we propose. For each pair of encoding techniques and clustering algorithms, we apply it for a given event log (L) and measure the quality metrics (s, v, t). Following, a positional rank is built for each metric (R_s, R_v, R_t), i.e, comparing the pairs of encodings and clustering in each dimension. Finally, a rank (R) is computed by the average of the metrics ranks. For example, considering the pairs $\langle E_1, C_1\rangle$, $\langle E_2, C_2\rangle$ and $\langle E_3, C_3\rangle$, their respective final ranks are 2, 1.67 and 2.33. The solution chosen as the meta-target is the one that minimizes the R function, which, in this example, is the pair $\langle E_2, C_2\rangle$.

5.5 Meta-model

Regarding the meta-learner, we applied the Random Forest (RF) algorithm [5] due to its robustness, being less prone to overfitting. Moreover, we applied a hyperparameter tuning technique to improve performance in the recommendation task. For that, we adopted a holdout strategy where 80% of the meta-database was used for tuning and 20% as the validation set. After a grid search tuning strategy with 5-fold cross-validation, the best hyperparameters were: (i) *50* as the number of trees composing the forest, (ii) *gini* as the criterion measuring split quality, (iii) *3* as the required minimum number of samples for a node split, (iv) *1* as the minimum number of samples required to be a leaf node, and (v) *log2* as the number of considered features for a split. The results reported in Sect. 6 were extracted when applying the tuned meta-model to validation data.

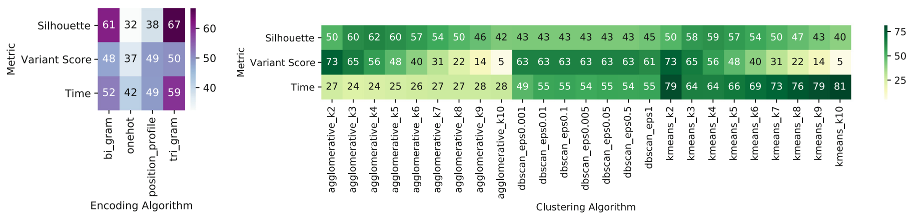
6 Results and Discussion

This section explores the meta-database composition by observing the encoding techniques and clustering algorithms chosen by their performance and balancing.

Next, an overall analysis, including the comparison of the proposed strategy with the baselines, is introduced.

6.1 Meta-learning Exploratory Analysis

The rank results, considering all algorithms for setting the meta-database, including the metrics used for ranking the meta-targets, are presented in Fig. 2. The heatmap plots show the ranking of the metrics s, v, and t for encoding (Fig. 2a) and clustering (Fig. 2b) used to sort and identify promising algorithms as meta-targets. Each ranking varies from 1 to 81, in which 1 is the best-ranked algorithm for a given metric.



(a) Encoding Ranking (b) Clustering Ranking

Fig. 2. Encoding and clustering rankings. Color gradient represents the ranking position variation.

Observing the encoding techniques (Fig. 2a), it is possible to identify a large discrepancy between them when evaluated by s, revealing the superiority of *one-hot* and *position profile* algorithms, whereas v score and t do not present a such prominent variation, leading to closer ranking positions. Note that the results report the average ranking position. In other words, one-hot encoding is the most well-ranked across the set of event logs, although it is not unanimous. However, when observing the clustering algorithms (Fig. 2b), it is possible to note a balance regarding s while v and t reveal discrepancies. The former (v) exposes the importance of hyperparameter definition since *agglomerative* and k-means ranged throughout the rankings when changing their hyperparameter k. Moreover, the t metric delivered an important perspective, in which each clustering algorithm is recognizable regardless of its hyperparameters. In particular, *agglomerative* and *dbscan* were superior to k-means. This superiority led to no usage of k-means as a clustering meta-target.

The meta-database was built using the combination of the top-ranked algorithms for each meta-instance (event logs). This combination leads to an imbalanced multi-output dataset. Combinations such as *one-hot* encoding with *agglomerative* clustering using 10 as k value ($onehot\_agglomerative\_k10$) represented 469 meta-instances. The second most frequent combination (171 meta-instances) was *position profile* with *agglomerative* clustering using 10 as k ($position\_profile\_agglomerative\_k10$). The third was *one-hot* using *dbscan* adopting a *eps* equals 0.001 (*onehot_dbscan_eps*0.001) in 125 meta-instances. These meta-target frequencies show the evident dominance of *one-hot* and *position profile* over the other encoding methods. *Bi-gram* was the best encoding technique for 37 meta-instances while *tri-gram* was the best one, combined with *dbscan*, only with four meta-instances. When evaluating from a clustering perspective, we observe a balance between *dbscan* with a wide range of *eps* and *agglomerative* using k as 10. Different values of k for *agglomerative* did not meet many meta-instances. Conversely, *dbscan* demonstrates the necessity of hyperparameter adjustments since different values of *eps* could match particular meta-instances. The imbalance issue was addressed by removing the minority class combinations, that is, meta-targets that appear less than five times. The final meta-database was composed of 1036 samples, with fifteen different combinations of *one-hot*, *position profile*, and *bi-gram* with *agglomerative* (k in {8, 9, 10}) and *dbscan* (*eps* in {0.001, 0.005, 0.05, 0.01, 0.1, 0.5, 1}).

6.2 Meta-model Performance

Using RF as our meta-model built over the meta-database, we analyzed the performance for both encoding and clustering algorithm recommendations (Fig. 3). It is worth mentioning that the problem was modeled as a multi-output problem using the BR transformation approach, addressing encoding and clustering at once. Since there are no other literature references, we employed majority voting and random selection as baseline approaches for comparison reasons. Majority voting works by always indicating the most common meta-target, i.e., the majority class in the meta-database. In this setup, *one-hot* and *agglomerative_k*10 are the most common encoding technique and clustering algorithm, respectively. Although a simple baseline, majority voting is a suitable comparison in machine learning applications, clearly specifying the minimum performance threshold. The random selection approach randomly chooses one of the possible pipeline combinations (coming from the set of meta-targets). This technique simulates a PM practitioner in a scenario without the availability of experts, a common situation in real environments. This way, we situate our method's performance both in relation to the machine learning and PM landscapes, creating an initial assessment and benchmark for the trace clustering problem.

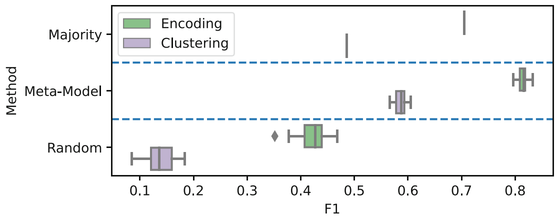


Fig. 3. Performance of the MtL framework to recommend the encoding technique and clustering algorithm in terms of accuracy and F1.

As observed in Fig. 3, our proposal obtained an F1 of 0.81 ($\pm$0.01) when recommending the encoding technique and an F1 of 0.59 ($\pm$0.01) for the recommendation of the clustering algorithm. The majority baseline for encoding obtained an F1 of 0.71 while the random baseline achieved 0.42 ($\pm$0.03). Regarding clustering, the majority obtained 0.49 of F1, and random selection reached 0.14 ($\pm$0.03). Our approach obtained a mean predictive performance of 0.7 for the whole trace clustering pipeline. The results were superior to the majority and random baselines, which averaged 0.59 and 0.28, respectively. Note that the majority voting results are boosted by the imbalanced scenario, for balanced meta-databases, the tendency is to underperform. The superior performance of our proposal confirms our hypothesis, i.e., there is a relationship between event log behavior and optimal pipelines. Since this relationship exists (and is partially captured by our proposed meta-features), our method outperforms the baselines. Given the universe of possibilities (81 combinations) and the limitations imposed by the imbalanced scenario, we consider the F1 performances suitable. Furthermore, this assessment serves as a benchmark for the area to be compared to alternative solutions proposed in the future.

7 Conclusion

This paper proposes an MtL framework to recommend the best pipeline for trace clustering based on a specific event log and its behavior. For that, we extract meta-features to describe event logs and match them with the best clustering pipeline by assessing three complementary metrics. The framework recommends a tuple $\langle$*encoding*, *clustering*, *hyperparameters*$\rangle$, making trace clustering solutions accessible for non-expert users and assisting experts with guided recommendations. Results have shown that the framework outperforms baseline approaches. In future research, we aim to extend the experimental evaluation to gather further insights into the relationship between trace clustering quality and event log behavior. Moreover, we plan to improve the modeling of the multi-output approach by testing different techniques, possibly taking advantage of the inter-correlation between different steps of the recommended pipeline.

8 Limitations and Broader Impact Statement

Our approach could be applied in a wide range of PM tasks, including process discovery, conformance checking, trace clustering, anomaly detection, and several others. This way, our research paves the way for automation in the business process domain, complemented with a supporting data-driven framework to study PM problems. Therefore, there are multiple benefits unlocked by the proposed technology, such as guidance for non-expert users and insights for experienced analysts. However, not enough attention has been paid to the over-application of automation techniques in PM. An important aspect touches the validity of the research and experimental design [23]. More specifically, the underlying behavior distribution of event logs might lead to unexpected results. This way, adopters of this tool should be careful in selecting a representative set of event logs to serve as the basis of the meta-database. Otherwise, the insights or recommendation quality might decrease. Being a technique that abstracts the pipeline for non-experts, the possibility of results misuse rises, thus requiring understanding about the possible domain risks when applying the tool.

References

1. Back, C.O., Debois, S., Slaats, T.: Entropy as a measure of log variability. J. Data Semant. **8**(2), 129–156 (2019)
2. Barbon Jr., S., Ceravolo, P., Damiani, E., Tavares, G.M.: Using meta-learning to recommend process discovery methods (2021)
3. Barbon J., Sylvio, C., Paolo, D., Marques Tavares, G.: Evaluating trace encoding methods in process mining. In: Bowles, J., Broccia, G., Nanni, M. (eds.) DataMod 2020. LNCS, vol. 12611, pp. 174–189. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-70650-0_11
4. Bose, R.P.J.C., van der Aalst, W.M.: Context aware trace clustering: towards improving process mining results. In: Proceedings of the 2009 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics (2009)
5. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
6. Ceravolo, P., Damiani, E., Torabi, M., Barbon, S.: Toward a new generation of log pre-processing methods for process mining. In: Carmona, J., Engels, G., Kumar, A. (eds.) BPM 2017. LNBIP, vol. 297, pp. 55–70. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-65015-9_4
7. Ceravolo, P., M. Tavares, G., Barbon Jr., S., Damiani, E.: Evaluation goals for online process mining: a concept drift perspective. IEEE Trans. Serv. Comput. 1 (2020)
8. De Koninck, P., vanden Broucke, S., De Weerdt, J.: act2vec, trace2vec, log2vec, and model2vec: representation learning for business processes. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 305–321. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_18
9. De Koninck, P., De Weerdt, J.: Scalable mixed-paradigm trace clustering using super-instances. In: International Conference on Process Mining (2019)
10. de Leoni, M., van der Aalst, W.M., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**, 235–257 (2016)
11. De Weerdt, J., vanden Broucke, S., Vanthienen, J., Baesens, B.: Active trace clustering for improved process discovery. IEEE Trans. Knowl. Data Eng. **25**(12), 2708–2720 (2013)
12. Delias, P., Doumpos, M., Grigoroudis, E., Manolitzas, P., Matsatsinis, N.: Supporting healthcare management decisions via robust clustering of event logs. Knowl. Based Syst. **84**, 203–213 (2015)
13. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 226–231. KDD 1996, AAAI Press (1996)
14. Fani Sani, M., Boltenhagen, M., van der Aalst, W.: Prototype selection using clustering and conformance metrics for process discovery. In: Del Río Ortega, A., Leopold, H., Santoro, F.M. (eds.) BPM 2020. LNBIP, vol. 397, pp. 281–294. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-66498-5_21

15. Greco, G., Guzzo, A., Pontieri, L., Sacca, D.: Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng. **18**(8), 1010–1027 (2006)
16. He, X., Zhao, K., Chu, X.: Automl: a survey of the state-of-the-art. Knowl. Based Syst. **212** (2021)
17. Hompes, B., Buijs, J., van der Aalst, W., Dixit, P., Buurman, J.: Discovering deviating cases and process variants using trace clustering. In: 27th Benelux Conference on Artificial Intelligence (2015)
18. Hou, J., Gao, H., Li, X.: Dsets-dbscan: a parameter-free clustering algorithm. IEEE Trans. Image Process. **25**(7), 3182–3193 (2016)
19. Koninck, P.D., Nelissen, K., vanden Broucke, S., Baesens, B., Snoeck, M., Weerdt, J.D.: Expert-driven trace clustering with instance-level constraints. Knowl. Inf. Syst. **63**(5), 1197–1220 (2021)
20. Koninck, P.D., Weerdt, J.D., vanden Broucke, S.K.L.M.: Explaining clusterings of process instances. Data Mining Knowl. Discov. **31**(3), 774–808 (2016)
21. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297. Oakland, CA, USA (1967)
22. de Medeiros, A.K.A.., et al.: Process mining based on clustering: a quest for precision. In: ter Hofstede, A., Benatallah, B., Paik, H.-Y. (eds.) BPM 2007. LNCS, vol. 4928, pp. 17–29. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78238-4_4
23. Mendling, J., Depaire, B., Leopold, H.: Theory and practice of algorithm engineering (2021)
24. Neubauer, T.R., Pamponet Sobrinho, G., Fantinato, M., Peres, S.M.: Visualization for enabling human-in-the-loop in trace clustering-based process mining tasks. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 3548–3556 (2021)
25. Polato, M., Sperduti, A., Burattin, A., Leoni, M.d.: Time and activity sequence prediction of business process instances. Computing **100**(9), 1005–1031 (2018)
26. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00328-8_11
27. Thaler, T., Ternis, S.F., Fettke, P., Loos, P.: A comparative analysis of process instance cluster techniques. Wirtschaftsinformatik **2015**, 423–437 (2015)
28. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 847–855 (2013)
29. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining Multi-label Data, pp. 667–685. Springer, US, Boston, MA (2010)
30. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
31. Vanschoren, J.: Meta-learning: a survey (2018). arxiv.org/abs/1810.03548
32. Ward, J.H.: Hierarchical grouping to optimize an objective function. J. Am. Statist. Assoc. **58**(301), 236–244 (1963)
33. Xu, D., Shi, Y., Tsang, I.W., Ong, Y.S., Gong, C., Shen, X.: Survey on multi-output learning. IEEE Trans. Neural Networks Learn. Syst. **31**(7), 2409–2429 (2020)
34. Zandkarimi, F., Rehse, J.R., Soudmand, P., Hoehle, H.: A generic framework for trace clustering in process mining. In: 2020 2nd International Conference on Process Mining (ICPM), pp. 177–184 (2020)