Cut to the Trace! Process-Aware Partitioning of Long-Running Cases in Customer Journey Logs

Gaël Bernard(✉), Arik Senderovich, and Periklis Andritsos

Faculty of Information, University of Toronto, Toronto, Canada
{gael.bernard,arik.senderovich,periklis.andritsos}@utoronto.ca

Abstract. Customer journeys are recordings of customer interactions with organizational information systems. These interactions are often recorded into so-called customer journey logs. Customer journeys often correspond to long-running and flexible traces that may temper the use of process analytics techniques such as process mining. A common method to make long-running traces suitable for process mining algorithms is to partition them at the largest temporal differences between consecutive events. However, these techniques ignores process context that journeys are often influenced by. In this work, we propose a probabilistic framework that generalizes previous techniques and introduces two novel process-aware partitioning approaches. The first method is inspired by the directly-follows relation, a predominant abstraction in process discovery. The second approach leverages LSTMs, a type of Neural Networks that learn long-term dependencies in sequences. We show that both approaches outperform existing time partitioning methods on both synthetic and real-world customer journey data.

Keywords: Customer journey analysis · Process mining · Trace partitioning

1 Introduction

A *customer journey* is an abstract representation of all interactions between a customer and an organization. In the banking industry, a journey involves interactions with tellers, website logins, money withdrawals, etc. Describing and understanding these customer pathways is referred to as customer journey analytics [20]. A recent study reveals that almost half of information technologies and business leaders consider customer journey analysis to be their top priority [11]. Better understanding of customers and deeper insights into their behavior implies improvement in quality-of-service. These insights are expected to increase customer satisfaction which, ultimately, is positively linked with revenue [20]. For these reasons, customer journey management is an essential topic in information systems management.

M. La Rosa et al. (Eds.): CAiSE 2021, LNCS 12751, pp. 519–535, 2021.
https://doi.org/10.1007/978-3-030-79382-1_31

Customer journey logs are sequential databases that contain transactions of customer interactions (aka 'touchpoints') with information systems (e.g., Customer Relationship Management systems). Due to the similarity between customer journey logs and event logs, process mining was shown to be a promising discipline for performing customer journey analysis [4]. By viewing a customer journey as a trace of events, one can immediately apply process mining techniques and gain insights into customer journeys.

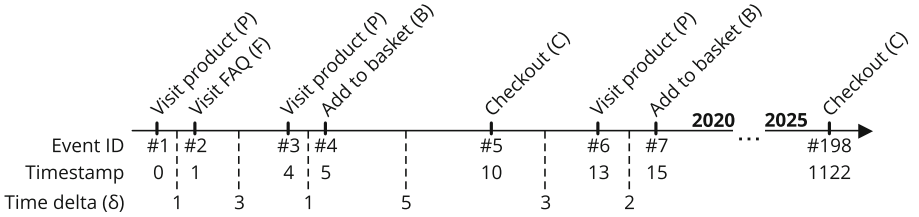

Fig. 1. Illustration of a customer journey composed of 198 events

Figure 1 illustrates a sample of a customer journey produced by a single customer browsing an online retail store. Clearly, after several years of customer interactions, one would expect to have a large number of events per customer (in our case, 198 events in 5 years). On the one hand, the flexibility offered to customers to consume a company's services accommodates their needs. On the other hand, these unique aspects of customer journeys, namely long-running and flexible cases, may result in several challenges for traditional process mining techniques.

Firstly, long-running cases cause process discovery algorithms to produce overly complex (spaghetti-like) process models, since they consider too many process variants [9]. Secondly, process discovery algorithms such as the inductive miner, [19], may overgeneralize and produce models that allow for events to happen in any order (flower models) [26]. In both cases, the resulting process models do not provide useful insights [9,26].

An approach that led its authors to win the 2011 BPI challenge, [13], consists of partitioning long traces into shorter ones by cutting at the largest time differences between events [8]. We shall refer to this approach as time-aware partitioning (TAP). The main limitation of TAP is that it relies solely on the time gap between events and disregards any process context. In our running example, the activity 'Checkout' shortly follows the activity 'Add to Basket'. Occasionally, an unexpected delay may occur between the two activities, e.g., due to customer characteristics. In such cases, TAP would assign the two closely related events to two distinct cases, causing a trace to start with the activity 'Checkout', which does not make sense from a business process standpoint.

In this paper, we use contextual information when cutting long-running traces. Specifically, we propose two novel *process-aware partitioning* approaches

that build upon a probabilistic view of the partitioning problem, while taking process context into account. The first method, namely *local context process-aware partitioning* (LCPAP) is based on local dependencies between event labels. Similarly to the directly-follows relation that often stars in process mining algorithms (e.g., [28]), LCPAP employs the fact that pairs of directly following activities will present (on average) shorter temporal gaps. Our second approach, *global-context process-aware partitioning* (GCPAP), leverages the strength of long-short term memory (LSTM) Neural Networks to better account for global context when partitioning customer journeys. The main contribution of the paper is threefold:

1. We formalize the long-running trace partitioning problem.
2. We provide a general probabilistic framework for temporal trace partitioning and show that our framework subsumes existing approaches.
3. We propose two novel process-aware approaches and demonstrate their effectiveness by conducting an extensive empirical evaluation.

The rest of the paper is organized as follows. Section 2 defines the problem of trace partitioning. Subsequently, Sect. 3 formalizes a probabilistic framework for customer journey partitioning and instantiates the framework using two novel approaches. Section 4 evaluates the framework using synthetic journey logs and show the relevance of the approach by demonstrating its application on real-life customer journey data. In Sect. 5, we discuss the limitations of our approach, while Sect. 6 and Sect. 7 present related work and conclude the paper.

2 The Problem of Customer Journey Partitioning

In this section, we pose the problem of customer journey partitioning (CJP). We refer to long-running traces as customer journeys, however, our problem and approaches for solving it are applicable to general event logs. We start by providing a model for customer journey data, proceed by motivating CJP using our running example and conclude the section with the CJP problem statement.

2.1 Customer Journey Data

Let $\mathcal{E}$, $\mathcal{I}$, $\mathcal{A}$, and $\mathcal{T}$ be the universes of customer events, customer identifiers, activities, and timestamps, respectively. We assume that a customer event $e \in \mathcal{E}$ comprises three elements $e = (i, a, t)$ such that,

- $i \in \mathcal{I}$ is a unique customer identifier,
- $a \in \mathcal{A}$ is a customer activity associated with the event,
- $t \in \mathcal{T}$ is a timestamp associated with the event.

A customer journey $\mathcal{J}_i$ is a sequence of events sorted by time and associated with customer i. Formally, let $\mathcal{E}^*$ be the set of all finite sequences over $\mathcal{E}$. Then, a journey $\mathcal{J}_i \in \mathcal{E}^*$ of length n can be written as a sequence of triplets,

$$\mathcal{J}_i = \langle (i, a_1, t_1), \ldots, (i, a_n, t_n) : t_1 \leq, \ldots, \leq t_n \rangle.$$

For convenience, given an event e we let $i(e)$, $a(e)$, and $t(e)$ be the identifier, the activity, and the timestamp associated with event e. A customer journey log $\mathcal{L}_{\mathcal{J}}$ is a set of customer journeys.

2.2 Customer Journey Partitioning

Motivation. We return to the journey log in our running example (Fig. 1). One searches for a way to cut the 198 events to form meaningful cases. Without cutting the log, applying process mining techniques to $\mathcal{L}_{\mathcal{J}}$ would result in a model that represents all customer interactions rather than customer journeys. We demonstrate this issue by running a discovery algorithm on a real-life customer journey log describing customers' interaction with a service desk (dataset used in the evaluation in Sect. 4.3). We observe that the resulting model, depicted in Fig. 2 is a 'spaghetti' model, which is far from useful if we wish to perform cycle time prediction or discover a variant that customers go through when instantiating the process. Motivated by the need for useful models, we arrive at the problem of customer journey partitioning.

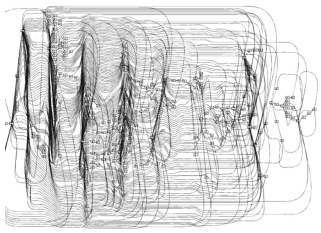

Fig. 2. Heuristic net discovered on real customer journey data.

Problem Formulation. As our first step towards formulating the problem, we define a journey partitioning function, $\chi : \mathcal{E}^* \rightarrow \mathbb{N}^{+*}$ to be a mapping that takes customer journeys and returns finite tuples of indices $\chi(\mathcal{J}_i)$ with each index corresponding to the end event of a case in journey i. The order between the elements of the tuple respects the temporal order within the journey, i.e., given a partition $\chi(\mathcal{J}_i) = (i_1, \ldots, i_k)$, we get that $t_{i_k-1} \leq t_{i_k}$. Thus, given a journey $\mathcal{J}_i = \langle (i, a_1, t_1), \ldots, (i, a_n, t_n) : t_1 \leq, \ldots, \leq t_n \rangle$ and a partition $\chi(\mathcal{J}_i) = (i_1, \ldots, i_k)$ we can derive k cases,

$$\begin{aligned} \sigma_1 &= \langle (i, a_1, t_1), \ldots, (i, a_{i_1}, t_{i_1}) \rangle \\ \sigma_2 &= \langle (i, a_{i_1+1}, t_{i_1+1}), \ldots, (i, a_{i_2}, t_{i_2}) \rangle, \\ &\ldots \\ \sigma_k &= \langle (i, a_{i_{k-1}+1}, t_{i_{k-1}+1}), \ldots, (i, a_{i_k}, t_{i_k}) \rangle. \end{aligned} \tag{1}$$

For example, the partition $\chi(\mathcal{J}_1) = (4, 198)$ transforms the journey from Fig. 1, $\mathcal{J}_1 = [(P, 0), (F, 1), (P, 4), (B, 5), (C, 10), ...]$, into two cases: $\sigma_1 = [(P, 0), (F, 1), (P, 4), (B, 5)]$ and $\sigma_2 = [(C, 10), ...]$. We denote $\mathcal{L}_{\chi(\mathcal{J}_i)}$ the set of cases $\{\sigma_1, \ldots, \sigma_k\}$ that results from applying χ to $\mathcal{J}_i$. We assume the existence of a *true* partitioning of the journey log, which we are aiming to reconstruct. Different instances of χ can be viewed as approximations of the true partitioning.

Next, we let $\Pi(\mathcal{L}_{\chi(\mathcal{J}_i)})$ denote the loss associated with wrong case reconstruction from customer journey i. The customer journey partitioning (CJP) problem is to minimize the total loss given χ and a journey log $\mathcal{L}_{\mathcal{J}}$ defined as,

$$\Pi_{\chi}(\mathcal{L}_{\mathcal{J}}) = \sum_{i=1}^{n} \Pi(\mathcal{L}_{\chi(\mathcal{J}_i)}). \tag{2}$$

There are various options for measuring the loss between partitions. We shall provide some of those distance functions when instantiating our approach in Sect. 4.1.

3 Probabilistic Customer Journey Partitioning

Solving CJP directly by minimizing $\Pi_{\chi}(\mathcal{L}_{\mathcal{J}})$ would require case labeling of the journey events, which is unrealistic to obtain in many practical scenarios. In this paper, we circumvent this challenge by introducing a probabilistic framework for CJP that would allow us to approximate the solution to the partitioning problem. The key in our solution is the ability to accurately identify that a case ends after a given event.

Formally, given an event $e \in \mathcal{E}$, let $E(e)$ be an indicator that equals 1 if e is a case-ending event, and 0 otherwise. Thus, we define $p(e) = P(E(e) = 1)$ as the probability that e is a case ending event. The foundation of our approach lies in the ordering of the probabilities for events to be case ending events and choosing the top K events[1] to set the cuts. Below, we provide three approaches that instantiate the above.

3.1 Time-Aware Partitioning (TAP)

Time-aware partitioning (TAP) assumes that journey cuts correspond to longer inter-event durations. TAP is used as the standard state-of-the-art way to partition long traces when preprocessing event logs (see [8]). A time partitioning algorithm cuts journeys at the longest time differences between events. To quote [8]:

> *"One can use a parameter, say δ days, to demarcate the boundaries between process instances. Two events or event sequences with a time period between them greater than δ fall under two process instances."*

We will formalize the above using our probabilistic framework by defining the dependence between $E(e)$ and observed inter-event times.

For a given journey of customer i, $\mathcal{J}_i$, let $\lambda_i = (\lambda_{i,1}, \ldots, \lambda_{i,n_i})$ be the sequence of inter-event times with n_i being the number of events in the journey, i.e., $\lambda_{i,1} = t_2 - t_1, \lambda_{i,2} = t_3 - t_2, \ldots$ and so on. For example, if our journey log has one journey, $\mathcal{J}_1 = [(P, 0), (F, 1), (P, 4), (B, 5), (C, 10), ...]$, the sequence λ_i is $[1, 3, 1, 5, ...]$. When we refer to some event e in the log, we denote λ_e the inter-event time between e and the next observed event in the same journey. If e is

[1] K is a hyper-parameter that controls for the number of cases in the journey log.

the last event in the journey, we set $\lambda_e = \infty$ to denote that the next event will not be occurring (as e is last); note that we only consider completed journeys.

Further, we assume that the inter-event time Λ_e (which is realized by λ_e) is a random variable that comprises a baseline inter-event time Λ_0 and a random gap Δ that exists only if the case ends after e,

$$\Lambda_e = \begin{cases} \Lambda_0, & \text{for } E(e) = 0 \\ \Lambda_0 + \Delta, & \text{for } E(e) = 1 \end{cases}$$

Let δ be the expected value of Δ, i.e., $\mathbb{E}\Delta = \delta$ and let $p(e) = P(E(e) = 1)$. From the law of total expectation we get that

$$\begin{aligned} \mathbb{E}\Lambda_e &= p(e)(\mathbb{E}(\Lambda_0 + \Delta)) + (1 - p(e))\mathbb{E}\Lambda_0 \\ &= p(e)(\mathbb{E}\Lambda_0 + \mathbb{E}\Delta) + (1 - p(e))\mathbb{E}\Lambda_0 = \mathbb{E}\Lambda_0 + p(e)\delta. \end{aligned} \tag{3}$$

Therefore, we get that

$$p(e) = \frac{\mathbb{E}\Lambda_e - \lambda_0}{\delta}, \tag{4}$$

with $\lambda_0 = \mathbb{E}\Lambda_0 > 0$ being a positive constant. This result allows us to compare two observed events e, e' and decide which one has the higher likelihood to end a case.

Consider two events in the journey log, e and e', such that the gap of the former is greater than the gap of the latter, namely $\lambda_e > \lambda'_e$. Since every event e corresponds to a single random variable Λ_e and a single observation λ_e, our best estimate of $\mathbb{E}\Lambda_e$ is $\widehat{\mathbb{E}\Lambda_e} = \lambda_e$. Therefore, we get that

$$\frac{\widehat{p(e)}}{\widehat{p(e')}} = \frac{\lambda_e - \lambda_0}{\lambda_{e'} - \lambda_0} > 1, \tag{5}$$

since $\lambda_0 > 0$ and $\lambda_e > \lambda_{e'} > 0$. Therefore, ordering the events according to their corresponding inter-event times (including ∞ for journey ending events) and cutting after the top K events on the list, guarantees that we are selecting the events with highest (estimated) chances of being case ending events.

One of the main drawbacks of TAP is that it does not take into account any contextual information. Returning to our example, although the activity 'Checkout' cannot happen before the activity 'Add to basket', TAP might insert a cut. In fact, TAP is sensitive to extreme values drawn from Λ_0, e.g., if the time between two consecutive events that belong to the same case is unexpectedly long, the two events might end up in two separate cases according to TAP.

In Fig. 3, we provide an illustrative customer journey that we use to apply the partitioning techniques. Taking this figure, TAP would partition at: ∞ (#10, journey ending events), 10 (#4), 5 (#7), and 4 (#5); respectively producing the cuts TAP4, TAP1, TAP3, and TAP2. Figure 3 illustrates that we cannot retrieve any of the ground truth by cutting based on the inter-event times. In the next part, we provide two approaches that take contextual information into account when cutting long-running traces.

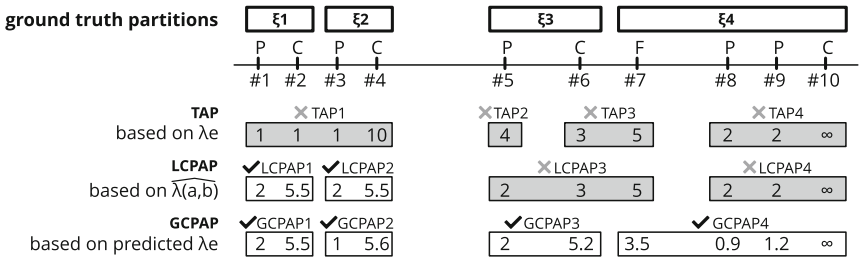

Fig. 3. Illustration of the three CJP techniques.

3.2 Local Context Process-Aware Partitioning (LCPAP)

Unlike TAP that only observes temporal differences, our second solution, namely LCPAP, employs the directly-follows relation between pairs of activities, and their inter-event times to decide on the cut. For LCPAP, we refine the definition of a case ending event by letting it depend on the pair of activities that corresponds to an event e and to its direct follower e' (both events are part of the same journey).

Let $d(e) \in \mathcal{A} \times (\mathcal{A} \cup \{\perp\})$ be the directly follows relation such that $d(e) = (a, b)$ with $a = a(e)$ and $b = a(e')$ and e' directly follows e. Note that $b = \perp$ implies that e is a journey ending event. We assume that the inter-event time, Λ_e, depends not on the event itself, but on the pair (a, b). Formally, we write it as,

$$\Lambda_e^{(a,b)} = \begin{cases} \Lambda_0^{(a,b)}, & \text{for } E(e) = 0, d(e) = (a, b) \\ \Lambda_0^{(a,b)} + \Delta, & \text{for } E(e) = 1, d(e) = (a, b) \end{cases} \tag{6}$$

This implies that the end of a case is dictated by Δ, as before, but its standard duration Λ_0 is a function of the activity pair, (a, b). Repeating a similar derivation as in Sect. 3.1, we get that

$$p_{a,b}(e) = P(E(e) = 1, d(e) = (a, b)) = \frac{\mathbb{E}\Lambda_e^{(a,b)} - \lambda_0^{a,b}}{\delta}. \tag{7}$$

Note that in the equation above, we added contextual information regarding the directly follows relation associated with event e, which was not used by TAP.

We estimate $p_{a,b}(e)$ by assuming that all baseline durations $\lambda_0^{(a,b)}$ are identical and equal to some λ_0. Therefore, it is enough to estimate $\mathbb{E}\Lambda_e^{(a,b)}$ by simply averaging the inter-event times, namely $\{\lambda_{i,j} \mid a_j = a, a_{j+1} = b\}$. We denote this average by $\bar{\lambda}_{(a,b)}$ and for any pair of events e, e' we write,

$$\frac{\widehat{p_{a,b}(e)}}{\widehat{p_{a,b}(e')}} = \frac{\bar{\lambda}_{(a,b)} - \lambda_0}{\bar{\lambda}_{(c,d)} - \lambda_0}, \tag{8}$$

with (a, b) and (c, d) being arbitrary pairs of activity labels of e and e', respectively. Hence, given a pair of events, it is enough to observe the average durations

of their activity labels $\bar{\lambda}_{(a,b)}$ to determine their ordering. Note that for $b = \perp$, we define $\bar{\lambda}_{(a,b)} = \infty$ and thus we shall always have a cut after a journey ending event.

Subsequently, we perform a cut after the top K observed average durations of the pairs that correspond to the events. For instance, in Fig. 3, we get the $\lambda_{P,C} = [1, 1, 4, 2]$ (event IDs #1, #3, #5, and #9). Thus, we get that $\bar{\lambda}_{(P,C)} = 2$. The four largest values are: ∞ (#10, journey ending events), 5.5 (#2, $\bar{\lambda}_{(C,P)}$), 5.5 (#4, $\bar{\lambda}_{(C,P)}$), and 5 (#5, $\bar{\lambda}_{(F,P)}$); respectively producing the cuts LCPAP4, LCPAP1, LCPAP2, and LCPAP3 visible in Fig. 3. Two out of the four cuts are correct with respect to ground truth.

3.3 Global Context Process Aware Partitioning (GCPAP)

Local context PAP considers the directly follows relation between a pair of events. As the name suggests, GCPAP generalizes the notion of context by adding: (1) w activities preceding the current event e, (2) w time inter-event times preceding event e, (3) activity $a(e)$ that is associated with e, (4) w activities following e, and (5) w inter-event times following e. If there are less than w events that precede or follow e, we take the actual number of preceding or following events. Formally, let $\kappa(e, w)$ be the concatenation of the five context elements of event e, hyper-parametrized by $w > 0$. Then, similarly to the definition of Λ_e in Eq. (9), we write,

$$\Lambda_e^{\kappa(e,w)} = \begin{cases} \Lambda_0^{\kappa(e,w)}, & \text{for } E(e) = 0, \kappa(e, w) \\ \Lambda_0^{\kappa(e,w)} + \Delta, & \text{for } E(e) = 1.\kappa(e, w) \end{cases} \tag{9}$$

Let $p_\kappa(e) = P(E(e) = 1), \kappa(e, w))$ be the probability of event e being a case ending event. By similar assumptions as in LCPAP derive that for any pair of events e, e':

$$\frac{\widehat{p_\kappa(e)}}{\widehat{p_\kappa(e')}} = \frac{\hat{\lambda}_{\kappa(e,w)} - \lambda_0}{\hat{\lambda}_{\kappa(e',w)} - \lambda_0}. \tag{10}$$

In this case, $\hat{\lambda}_{\kappa(e,w)}$ is the mean value of the random variable $\Lambda_e^{\kappa(e,w)}$. However, since it comprises a complex context conditioned on the 5 elements mentioned above, we do not simply estimate it using a simple mean over the different context levels, but regress the inter-event time on the context elements (treating them as features). Therefore, estimating $\hat{\lambda}_{\kappa(e,w)}$ (and $\hat{\lambda}_{\kappa(e',w)}$) turns into a regression problem that we solve using deep-learning.

Concretely, we propose the Neural Network (NN) architecture presented in Fig. 4. It comprise LSTM, which is a special type of Recurrent Neural Network (RNN) introduced in [17]. LSTM possesses an advanced memory cell that gives it powerful modeling capabilities for long-term dependencies [27]. The advantage of LSTM is that it can work with sequence tensor possessing more than two dimensions, which is unconventional for machine-learning algorithms [29]. Ultimately, it means that we can use the sequence of activities "as-is" [29] without retreating to simplifications such as the directly-follows relation. The five inputs visible in Fig. 4 reflect the 5 aforementioned pieces of contextual data. The NN is composed of two separate LSTM layers made, followed by a Dense layer. It is a design choice since various alternative architectures are possible. The output is the expected time gap before the next event that we will use to cut the traces. Figure 3 provides an example of how this approach may outperform TAP and LCPAP. In fact, the four largest predicted time gaps are: ∞ (#10, journey ending events), 5.6 (#4), 5.5 (#2), and 5.2 (#6); respectively producing the cuts GCPAP4, GCPAP2, GCPAP1, and GCPAP3 visible in Fig. 3. We conveniently set the predicted values to time gaps that allow GCPAP to find the true partitions. It was done for demonstration purposes as more than 10 events are typically needed to get predicted time gaps that allows to retrieve ground truth cuts.

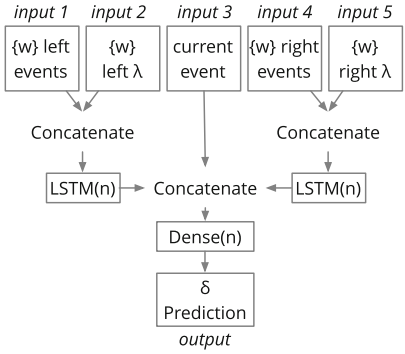

Fig. 4. NN architecture for the GCPAP.

4 Evaluation

The goal of the experimental evaluation is to benchmark the three methods for customer journey partitioning presented in Sect. 3. Our experiments show that:

1. The best performing CJP is GCPAP, followed by LCPAP for both synthetic and real customer journey data, while TAP comes last.
2. The synthetic dataset highlights the value of LCPAP, especially when the customer journey data is complex; i.e., when the time gaps are not very informative.
3. We show in the real dataset the existence of outlier time gaps that TAP cannot handle well.

The implementation of the three methods, as well as the results of the experiments and the instructions to reconstruct the experiments are available online[2].

[2] https://github.com/gaelbernard/cjp.

For the GCPAP, we set w to 10, the number of unit to 64 (LSTM and dense layers) and the number of epochs to 20.

In the remainder of the section, we first introduce the loss function, Π, used for the evaluation. Then, we present the first set of experiments performed on a synthetic customer journey dataset. We report a more extensive experiment containing 10 customer journey logs, and a scalability analysis in a separate technical report [7].

Lastly, we demonstrate the applicability of our approach by presenting a second set of experiments performed on a real-world dataset, which contains customer interactions with a service desk of a large municipality. Note that the real-world data experiment does not aim at an exhaustive application of the approach to numerous datasets, but rather exemplifies how one would use the method in similar realistic settings.

4.1 The Loss Function

Customer journey partitioning can be viewed as a series of binary classification problems: we either correctly identify the true case, or make an error identifying that a sequence corresponds to a case. Our classification may hence fall into one of the following categories: (1) True Positive (TP): Partition correctly found, (2) False Positive (FP): Case wrongly cut, (3) True Negative (TN): Case correctly not cut, and (4) False Negative (FN): Partition not found. The True Positive Rate (TPR), calculated as $\frac{TP}{TP+FN}$, provides the fraction of partitions we are able to retrieve. Conversely, the False Positive Rate (FPR), $\frac{FP}{FP+TN}$, measures the fraction of cases that are wrongly cut. The number of cases, K, is not known in advance and hence we can reach a perfect TPR by classifying all the events as cuts ($K = |\mathcal{L}_{\mathcal{J}}|$). The other extreme would be to get a perfect FPR by not partitioning the event logs ($K = 0$).

ROC curve analysis offers a suitable tool to evaluate the trade-off between TPR and FPR for various values of K. The diagonal represents a random guess (dashed lines in Fig. 7), while the coordinate (0,1) is a perfect classifier. Moreover, we use the *area under the curve* (AUC) as a loss function, since it allows summarizing the ROC curve using a single metric [10].

4.2 Evaluation Using Synthetic Customer Journey Data

The Dataset. We define a process model depicted in Fig. 5. Then, we produce 1,000 traces from this process model using the plugin "Simple simulation of a (stochastic) Petri net" in ProM 6.7. Next, we randomly assign each of the traces to 10 customers in order to form 10 customer journeys. Eventually, this means that these 10 customer journeys are composed, on average, of 100 stacked traces

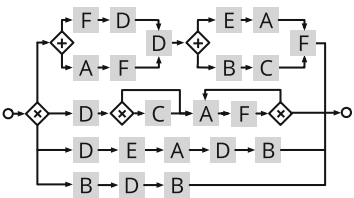

Fig. 5. Process Model used for the experiment.

that we wish to retrieve by leveraging one of the three partitioning techniques. The inter-event time is a key parameter to control for the complexity of the experiment. To simulate the time difference, we assume that the inter-event time corresponds to an exponential distribution, which implies that events arrive according to a Poisson process. Formally, we set

$$timeDiff(\lambda, r) = \frac{-ln(r)}{(1/\lambda)} \begin{cases} +d, & \text{if new variant} \\ +0 & \text{otherwise} \end{cases}$$

where:

λ is the rate of the Poisson Process that we set to 1
r random value uniformly distributed between 0 and 1
d is a constant (delay) which is added if the next event belongs to a new journey

By adding a delay (d) when the next event belongs to a new case, we control the difficulty to retrieve the optimal partitions. The larger the d is, the easier it is to retrieve the optimal partitions. Note that the delay is constant, but the inter-case time is not since we sum the delay with a random value.

Results. Figure 6 shows the AUC for various delays. As can be seen, GCPAP outperforms the two other approaches, in particular, for low delays. Even though LCPAP loses accuracy compared to GCPAP, this approach considers a limited context (i.e., pairs of events) and is hence very efficient to compute (see the technical report, [7]) and still largely outperforms TAP.

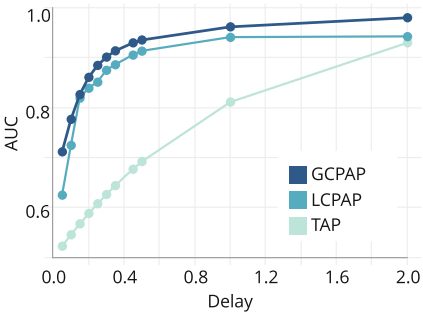

Fig. 6. AUC for various delays.

Figure 7 shows the ROC curve for the delays: 0.05, 0.10, 0.30, and 0.50. We observe that TAP curves have an interesting shape: the TPR is saturating before the FPR. For instance, for a delay of 0.5, we can see that the TPR reaches 100% when the FPR is approximately 60%. We interpret the results as follows: at TPR saturation point, we already retrieved all the partitions and augmenting K is only making the FPR worse, i.e., we are trying to cut the trace at values that are lower than the delay. The LCPAP curves also have an interesting informal characteristic: they show a 'stair-like' shape. We believe that this is due to the lower number of unique values at which we can apply a cut.

In [7], we extended the experiment with 10 new process models and conducted a scalability analysis.

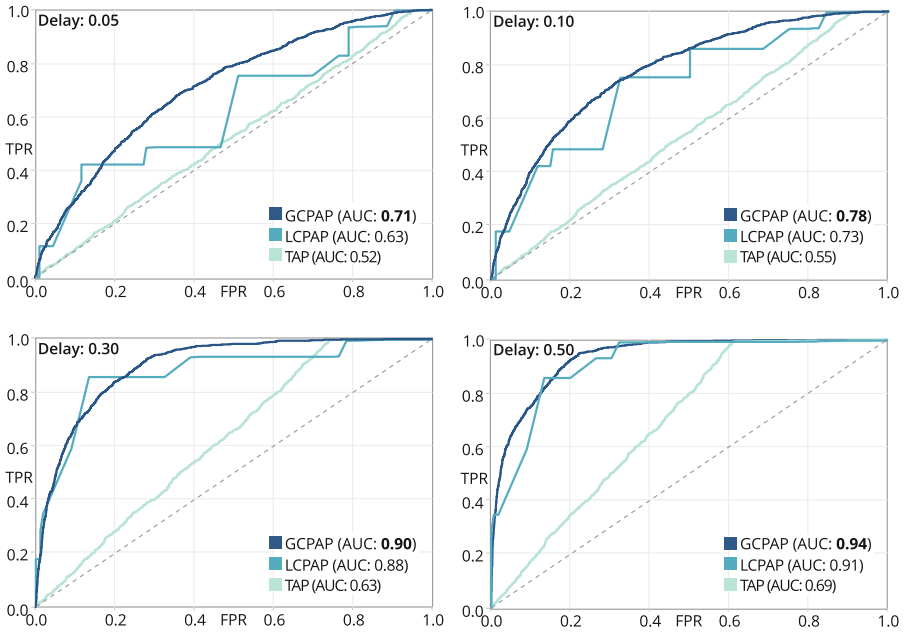

Fig. 7. ROC curves for the experiment with the synthetic datasets for the delays 0.05, 0.10, 0.30 and 0.50.

4.3 Experiment with Real Customer Journey Data

Dataset. We used an event log that contains tickets handled by a service desk of a municipality over a time span of one year. Since tickets have well defined start and end timestamps, it does not meet our definition of long-running customer journeys. However, we use this to our advantage by considering that ticket identifiers are the true cuts that we wish to retrieve using our three techniques. Hence, we keep the ticket identifier for the evaluation as our ground truth and consider customer journeys to be interactions between a customer and the service desk. We only keep the top 1% of customers that generated most of the tickets throughout the year to focus the experiment on difficult cases where many tickets are generated within a short period of time. From this procedure, we obtained a customer journey event log with the following characteristics: 27 customer journeys, 7.9K tickets, and 100.9K 'touchpoints' of 290 types. The median time between events belonging to the same case was 34 s, while the median time between events belonging to distinct cases within a customer journey was 30 min. Despite this massive mean time differences, retrieving the journeys is not straightforward due to the existence of extreme values.

Results. Figure 8 shows the ROC curves and the AUCs of the three CJP methods. Similar to the results for the synthetic experiments, GCPAP outperforms the two other approaches. It is interesting to note that the TPR for the TAP reaches a plateau around 90% and jump to 100% only when $K = |\mathcal{L}_{\mathcal{J}}|$. In fact, sometimes, two events from two distinct tickets closely follow each other, causing the TAP to wrongfully assign them to the same case. It demonstrates that the extreme values mentioned in Sect. 3 do exist in real customer journey data. We may further observe that GCPAP and LCPAP solve this issue as they can exceed the value of 90% TPR much sooner compared to TAP. Lastly, we note that the LCPAP 'stair-like' shape highlighted with the synthetic dataset is not as strong in Fig. 7 because of the large number of unique events of the real dataset (i.e. 290).

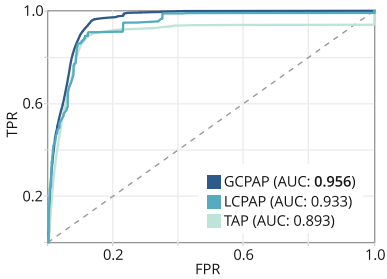

Fig. 8. ROC curves for real customer journey data

5 Discussion and Limitations

In this part, we discuss the limitations of our approach. Our solution for the CJP problem can be easily extended and generalized; furthermore, it does not rely on any knowledge of the underlying process models. However, we make several assumptions about the data that may limit the applicability of our method.

First, we assume that traces within a journey cannot overlap. Specifically, our technique cannot recompose correctly multiple journeys from a single customer that runs simultaneously. Second, we assume that the touchpoints happening along the journey on various channels are available on a uniform granularity level and exhibit reasonable data quality. One may require some preprocessing to level event abstractions, which could be performed using one of the event abstraction techniques studied in [30]. Lastly, being able to extract events logs from heterogeneous information systems is a well-recognized dilemma within the process mining community [1]. We claim that the customer journey's complex nature even exacerbates this challenge. The fact that customer journeys are complex is also what makes it appealing for process mining analysis in the first place.

By offering a flexible framework for partitioning customer journeys into traces, we hope to ease the analysis of customer journeys with process mining. Having said that, we see this work as a stepping stone for improving the proposed methods, which would relax some of the aforementioned assumptions.

6 Related Work

Clickstream analysis requires a preprocessing step similar to the partitioning of customer journeys to extract website usage insights. This step consists of

grouping web interactions into user sessions [24]. The predominant technique to partition clickstream data is based on time and is equivalent to TAP, e.g., [2, 18,24]. Other approaches leverage clickstream-specific information not available in a customer journey context, such as the 'referer' [3], or logging events in a service-oriented architecture [14].

In the human-computer interaction community, Leno et al. propose a workers' routines partitioning algorithm of user interaction logs using a graph-based approach [21]. The underlying assumption is that a worker will perform the same task with some variance and noise, and the goal is to partition them. A limitation of this approach is that multiple routines cannot share a single activity label, limiting its applicability to the less-structured customer journey context. Another approach in the same community is presented in [12]. It consists of a custom-made frequent itemset mining that encompasses user behavior-specific metrics. Interestingly, one assumption is that the order of the tasks within a routine is not important. Both these works do not take into consideration the time between events.

Event log abstraction is a recent area of studies within process mining that aims to transform low-level events (e.g., a series of clicks on a website) into meaningful activities to process stakeholders (e.g., canceling an order). One can refer to [30] for an extensive literature review about abstraction techniques. Some of these approaches require business knowledge inputs that can take the form of ontologies [22], annotated event logs [25], interactions with business experts [5], or composition rules [16]. In [23], Mannhardt and Tax propose to rely on local process models (LPMs) introduced in [26] to by-pass the needs to input business knowledge. What makes this approach appealing is that LPMs allows dealing with long and chaotic traces and, hence, with customer journey logs. Most of these abstraction approaches disregard the time and consider only the activity context – the exact opposite of TAP. One interesting exception is the work from [22] where a threshold is required to abstract the activities. Overall, abstracting and partitioning traces fulfill a distinct goal but complement each other to analyze overly complex event logs.

In the process mining community, LSTM based methods have been shown to work well for predictive process monitoring tasks such as predicting the next activity [15,27], timestamp prediction [27], or trace truncation [6]. However, to the best of our knowledge, LSTMs have never been used as a preprocessing step for event logs.

7 Conclusion

In this work, we defined and solved the problem of customer journey partitioning. Our approach generalizes and extends existing process mining methods for finding cuts in traces. We found that the state-of-the-art method for cutting traces, namely TAP, was overrun by our newly proposed process-aware partitioning (PAP) methods that are able to take into account local (LCPAP) and global (GCPAP) contextual information. Both LCPAP and GCPAP outperformed TAP on both synthetic and real customer journey data. Compared to

TAP and using LCPAP and GCPAP, we improved the partitioning of the real customer journey data by 4.0 and 6.3% points, respectively.

In future work, we would like to further employ the flexibility offered by Neural Network architectures, to potentially improve the partitioning of long-running by taking additional contextual information into consideration (e.g., resources, departments, day of the week). In addition, we aim at exploring inter-case dependencies between journeys, since customer interactions are often performed in groups (e.g., family members and co-workers often interact together).

References

1. Aalst, W.: Data science in action. In: Process Mining, pp. 3–23. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4_1
2. Benevenuto, F., Rodrigues, T., Cha, M., Almeida, V.: Characterizing user behavior in online social networks. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, pp. 49–62 (2009)
3. Berendt, B., Mobasher, B., Nakagawa, M., Spiliopoulou, M.: The impact of site structure and user environment on session reconstruction in web usage analysis. In: Zaïane, O.R., Srivastava, J., Spiliopoulou, M., Masand, B. (eds.) WebKDD 2002. LNCS (LNAI), vol. 2703, pp. 159–179. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-39663-5_10
4. Bernard, G., Andritsos, P.: A process mining based model for customer journey mapping. In: Forum and Doctoral Consortium Papers Presented at the 29th International Conference on Advanced Information Systems Engineering (CAiSE Forum), pp. 49-56. CEUR workshop proceedings (2017)
5. Bernard, G., Andritsos, P.: CJM-ab: abstracting customer journey maps using process mining. In: Mendling, J., Mouratidis, H. (eds.) CAiSE 2018. LNBIP, vol. 317, pp. 49–56. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-92901-9_5
6. Bernard, G., Andritsos, P.: Truncated trace classifier. removal of incomplete traces from event logs. In: Nurcan, S., Reinhartz-Berger, I., Soffer, P., Zdravkovic, J. (eds.) BPMDS/EMMSAD -2020. LNBIP, vol. 387, pp. 150–165. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49418-6_10
7. Bernard, G., Senderovich, A., Andritsos, P.: Cut to the trace: Technical report. Technical report, University of Toronto (Mar 2021). https://github.com/gaelbernard/cjp/raw/master/TechnicalReport/TechnicalReport.pdf
8. Bose, R.P.J.C., van der Aalst, W.M.P.: Analysis of patient treatment procedures. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM 2011, Part I. LNBIP, vol. 99, pp. 165–166. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28108-2_17
9. Bose, R.J.C., Mans, R.S., van der Aalst, W.M.: Wanna improve process mining results? In: IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 127–134. IEEE (2013)
10. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recogn. **30**(7), 1145–1159 (1997)
11. Daigler, J., Davies, J., Manusama, B., Bharaj, G.: Market guide for customer journey analytics. Technical report, Gartner (Feb 2019)

12. Dev, H., Liu, Z.: Identifying frequent user tasks from application logs. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces, pp. 263–273 (2017)
13. van Dongen, B., Ferreira, D.R., Weber, B.: Business processing intelligence challenge 2011 (bpic 11). Technical report, IEEE Task Force on Process Mining (2011). https://www.win.tue.nl/bpi/doku.php?id=2011:challenge
14. Dustdar, S., Gombotz, R.: Discovering web service workflows using web services interaction mining. Int. J. Bus. Process Integr. Manag. **1**(4), 256–266 (2006)
15. Evermann, J., Rehse, J.-R., Fettke, P.: A deep learning approach for predicting process behaviour at runtime. In: Dumas, M., Fantinato, M. (eds.) BPM 2016. LNBIP, vol. 281, pp. 327–338. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58457-7_24
16. Fazzinga, B., Flesca, S., Furfaro, F., Masciari, E., Pontieri, L.: Efficiently interpreting traces of low level events in business process logs. Inf. Syst. **73**, 1–24 (2018)
17. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. textbf9(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
18. Kumar, A., Salo, J., Li, H.: Stages of user engagement on social commerce platforms: analysis with the navigational clickstream data. Int. J. Electron. Commer. **23**(2), 179–211 (2019)
19. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
20. Lemon, K.N., Verhoef, P.C.: Understanding customer experience throughout the customer journey. J. Mark. **80**(6), 69–96 (2016)
21. Leno, V., Augusto, A., Dumas, M., La Rosa, M., Maggi, F.M., Polyvyanyy, A.: Identifying candidate routines for robotic process automation from unsegmented ui logs. In: 2020 2nd International Conference on Process Mining (ICPM), pp. 153–160. IEEE (2020)
22. Leonardi, G., Striani, M., Quaglini, S., Cavallini, A., Montani, S.: Towards semantic process mining through knowledge-based trace abstraction. In: Ceravolo, P., van Keulen, M., Stoffel, K. (eds.) SIMPDA 2017. LNBIP, vol. 340, pp. 45–64. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11638-5_3
23. Mannhardt, F., Tax, N.: Unsupervised event abstraction using pattern abstraction and local process models, pp. 55–63 (2017)
24. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.N.: Web usage mining: discovery and applications of usage patterns from web data. ACM SIGKDD Explor. Newsl. **1**(2), 12–23 (2000)
25. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.: Mining process model descriptions of daily life through event abstraction. In: Bi, Y., Kapoor, S., Bhatia, R. (eds.) IntelliSys 2016. SCI, vol. 751, pp. 83–104. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-69266-1_5
26. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.: Mining local process models. J. Innov. Digit. Ecosyst. **3**(2), 183–196 (2016). https://doi.org/10.1016/j.jides.2016.11.001. http://www.sciencedirect.com/science/article/pii/S2352664516300232
27. Tax, N., Verenich, I., La. Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30

28. 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)
29. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Trans. Intell. Syst. Technol. **10**(4) (2019). https://doi.org/10.1145/3331449
30. van Zelst, S.J., Mannhardt, F., de Leoni, M., Koschmider, A.: Event abstraction in process mining: literature review and taxonomy. Granular Comput. 1–18 (2020). https://doi.org/10.1007/s41066-020-00226-2