Predicting Process Behavior Meets Factorization Machines

Wai Lam Jonathan Lee[a], Denis Parra[a], Jorge Munoz-Gama[a], Marcos Sepúlveda[a]

[a] *Pontificia Universidad Católica de Chile, School of Engineering, Department of Computer Science, Vicuña Mackenna 4860, Macul 7820436, Región Metropolitana, Chile*

Abstract

Predictive business process monitoring methods use datasets of completed cases related to a process to predict the behaviour of running cases. To handle the large amounts of available event data, recent works have turned to deep learning techniques and have achieved fairly accurate results. However, results from these techniques are often difficult to interpret and explain. In the area of Recommender systems, factorization models have been an important class of predictive techniques due to its scalability and ability to infer latent features. Motivated by research in Recommender systems, this paper presents a predictive model that combines matrix factorization techniques from Recommender systems and knowledge from Business Process Management to learn interactions between latent features that can be used to predict the next event of an ongoing case. Evaluation on two real-life datasets from a Dutch Financial Institute and Volvo IT Belgium shows that the approach yields results that are comparable and at times superior to state-of-the-art techniques such as neural networks, yielding at most a precision of 0.87 for next event predictions.

Keywords: Recommender Systems, Business Process Management, Predictive Business Process Monitoring

1. Introduction

The execution of a process in real-life can be highly unpredictable: unforeseen events can occur to cause deviations from the prescribed procedures, workarounds might be adopted to bypass insufficiencies of the established protocols, and concept drifts can lead to a process to change over time. Yet, the capability to predict future behavior of a business process is important (Evermann et al., 2016b). Being able to identify and accurately predict different factors of ongoing process instances helps facilitate better decision making such as resource allocation, alert of compliance issues, and recommendations to take different actions (Kang et al., 2012). This motivates **predictive business process monitoring**, a family of online process monitoring methods that seeks to predict the unfolding of running cases based on models learnt through historical event logs (Maggi et al., 2014).

However, despite the structured nature of event data, prediction tasks for processes are difficult. Logs of event data (event logs) often only contain a small portion of the possible executions permitted by the process. Moreover, as with many branches of data science, it suffers from the absence of negative samples; only positive examples of how the process can be executed are provided. This has motivated research into using well-known sophisticated prediction models, e.g., Long Short-Term Memory (LSTM) neural networks (Tax et al., 2016; Evermann et al., 2016b), to predict future process behavior using historical event data.

Existing approaches can be divided into two kinds: explicit process representations where the process model corresponding to the process is considered explicitly in the prediction model, and implicit process

*Corresponding Author: Wai Lam Jonathan Lee
Addr.: Vicuña Mackenna 4860, Macul 7820436, Región Metropolitana, Chile; Tel.: +56 2 2354 4439
Email addresses: walee@uc.cl (Wai Lam Jonathan Lee), dparra@uc.cl (Denis Parra), jmun@uc.cl (Jorge Munoz-Gama), marcos@ing.puc.cl (Marcos Sepúlveda)

representations where the process model is implicitly reflected in the prediction model (Evermann et al., 2016b). The approach proposed by this paper sits between the two ends, where features of a process model can be explicitly considered in the prediction model while latent features are also inferred from the historical data. This means that documented process models and domain knowledge as well as latent variables can be leveraged to improve prediction.

One research area where models with latent representations have had an important effect is Recommender systems (RS) (Koren et al., 2009; Koren, 2008; Liu & Wu, 2016). In Recommender systems, one of the major challenges is the small amount of information on consumed items per user compared with the catalogue of millions of items on offer (Khusro et al., 2016); one can imagine that an average user of Amazon hardly consumes a significant amount of the millions of items that are on offer. This fundamental challenge of high sparsity has motivated different machine learning techniques that are scalable to the large volume of data (Koohi & Kiani, 2017). In particular, factorization models have been an important class of techniques due to its ability to infer latent features from a highly sparse dataset to facilitate prediction and recommendation tasks. Similarly, a case in a process represents a small portion of the possible behavior in the business process (van der Aalst, 2016). This common challenge of highly sparse data in the two research areas motivated our exploration of factorization models in Recommender systems for predictive tasks in a business process setting. In particular, Factorization Machines (FM) is chosen for its ease of incorporating different types of input data as "context" for the prediction tasks, its computational efficiency as compared to other similar methods, e.g., Support Vector Machine (SVM), and most importantly, its ability to infer latent features from a highly sparse dataset.

Overall, our contributions can be summarized as follows:

- The identification of a promising predictive model from the field of Recommender systems that meets the requirement for the next event prediction task in a business process context.
- A framework that incorporates Factorization Machines for the next event prediction task of an ongoing case in a process in Section 5.2.
- Leveraging the assumption of an underlying process in event data to improve prediction precision. This involves the definition of a novel representation of event data to enable the prediction task in Section 5.1.
- Address the absence of negative feedback information in training dataset to improve prediction precision in Section 5.3.
- Empirically validating the effectiveness of the proposed approach by comparing experimental results between the approach and existing state-of-the-art approaches on two real-life datasets in Section 7.

The rest of the paper is structured as follows: Section 2 provides a brief background for the paper. Next, Section 3 presents the related work in the literature. This is followed by the preliminaries in Section 4. Section 5 presents the proposed predictive model and the representation of event data to enable the prediction task. The experimental setup and results are presented in Section 6 and Section 7 respectively. The paper ends with the conclusions and future work in Section 8.

2. Background

In this paper, we present a predictive model motivated by the research in Recommender systems –which addresses data sparsity with latent factor models– and combined with techniques from Business Process Management. In this section, we provide a brief background to the two research areas.

2.1. Recommender systems

Recommender systems are software tools and techniques providing suggestions for items to be of use to a user (Ricci et al., 2015). The suggestions relate to different decision-making process such as what items

to buy, which movie to watch, or which song to listen to. Therefore, one important aspect is to accurately predict a user's preference or view towards an item that has yet to be consumed by him/her based on the user's previous item consumption. As such, it is easy to see the similarity between the task of predicting the next event of an ongoing case in a business process and the predicting the preference of a user towards an unconsumed item (the ongoing case can be seen as the user and the executed events of the ongoing case can be seen as the previously consumed items).

2.2. Business Processes Management (BPM)

The BPM research discipline uses various tools, techniques, methods and methodologies to support business processes and to take advantage of improvement opportunities Dumas et al. (2013). Traditionally, BPM has been a model-driven discipline where processes are analyzed through model representations e.g., BPMN, Petri nets, and etc. However, with the rapidly increasing availability of event data about processes, there has been a gaining momentum of adapting data-driven techniques to improve processes. In particular, the discipline of Process Mining has emerged in the recent years as the driving force to adapting data-driven approaches to improving processes (van der Aalst, 2016).

Next, the related works in the existing literature are presented.

3. Related work

Similar to the previous section, to structure the related work, we present the related work from the two areas, analyzing the state-of-the-art techniques for the prediction task of future events based on a sequence of previous events.

3.1. Recommender systems

One popular family of predictive models are the factorization models, which use matrix factorization techniques to learn latent featurest that can be used for prediction tasks (Koren et al., 2009). Intuitively, this could appreciated with the example of predicting movie ratings given by different users. Unobserved user characteristics such as his/her preference for romantic movies can influence the rating that a user gives to a movie. In general, matrix factorization techniques aim to learn these "latent" features to facilitate prediction tasks. In the typical factorization model, such as the SVD++ model (Koren, 2008), the order in which the previous items are consumed is not considered, i.e., the time factor. However, this is fundamental in the context of a business process as different situations can often require different execution order of the same set of activities. Hence, we focus on the much more limited list of factorization models that take into the time factor of previous consumptions.

The Factorizing Personalized Markov Chain (FPMC) model ranks products based on the last purchases of the user (Rendle et al., 2010). While matrix factorization methods are used to learn latent features from observed data, Markov chain methods model sequential behavior by learning a transition matrix over items that is used to predict a user's future items based on his/her recent actions. While modeling sequential behavior using Markov chain methods is appropriate in the recommender system context where the average user only has one or two previous actions, it is much more difficult in the context of a process where the average case lengths can be very long. There are several similar sequential recommenders that capture sequential behavior using Markov chain methods (He et al., 2016; Natarajan et al., 2013; He & McAuley, 2016).

Other than using Markov chain methods, one can learn item embeddings by factorizing item co-occurrence. The co-factorization model, CoFactor, simultaneously factorizes both the observed user-item interaction matrix and the item-item co-occurrence matrix so that co-occurrence relations between items can be learnt (Liang et al., 2016).

Point of interest (POI) recommendation is another area where factorization models are used for data that have sequential information in the form of temporal information, e.g., the sequence of places that a user visits in a period of time. In (Cai et al., 2018), a Feature-Space Separated Factorization Model (FSS-FM) is proposed that the POI feature spaces as separated slices.

In this paper, we adapt the factorization model, Factorization Machine (FM), as the predictive model (Rendle, 2010). FMs are a general predictor that works with any real valued feature vector. It can been shown that the FM is a generalized model that can mimic many of the existing state-of-the-art factorization model, e.g., FPMC and SVD++. By identifying the common challenge in tackling data sparsity between Recommender systems and event prediction in BPM, factorization machine is the appropriate base model due to its ease of incorporating different types of input data as "context" for the prediction tasks, its computational efficiency, and its ability to learn latent features from sparse data. To the best of our knowledge, there has yet to be any applications of FMs for prediction tasks in the context of a business process.

3.2. Recommendation in Business Processes Management (BPM)

Predictive techniques are one of the important subareas to a data-driven approach to improving processes. Whereas in the past, process prediction research has been focused on prediction of process outcomes such as remaining time to completion, there have been a number of recent work on prediction of the next event in a process (Breuker et al., 2016; Evermann et al., 2016a; Le et al., 2012; Lakshmanan et al., 2015; Ceci et al., 2014; Unuvar et al., 2016). These work can be best separated by the way in which the underlying process is represented in the predictive model. Specifically, there are those that use an explicit model representation and those that use an implicit model representation.

Works that use explicit model representations include state-transition model, HMM (hidden Markov models), or PFA (probabilistic finite automatons) models. In the paper (Lakshmanan et al., 2015), a five step approach is proposed to predict the next event of an ongoing case. First, a process model is discovered from an event log. Second, a decision tree is learnt using the process model. Third, given a partial trace of a running process instance, the decision tree is used to create an instance-specific markov chain for next event predictions. The fourth and fifth step consider parallelism in the model to enhance next event predictions. Similarly, in the paper (Unuvar et al., 2016), different representations of executed traces are used to build decision trees as prediction models.

One limitation of markov models is the limited coverage of observed traces in the event log compared with the possible process behavior. While higher-order markov chains promise better accuracy, many of the combinations at the state-transition matrix of higher-order markov chains would not be observed in the event data. This led to the use of lower-order markov models that only consider the previously executed event.

The Markov Sequence Alignment (MSA) approach proposed in the paper (Le et al., 2012) alleviates the issue by supporting a markov model with sequential alignment techniques. In MSA, each trace prefix is modeled as a state and the next activity as a transition in a state-transition matrix. This means that a state transition matrix can be built from the observed prefixes and their next events in the event log. Assuming that similar sequences are likely to produce the same outcome, predictions for a running case that has reached a state not contained in the state-transition matrix are made by referring to most similar observed cases. Similarity between traces is computed using string edit distance. The approach is evaluated on two datasets from a telecommunications company with reported optimal accuracies on next event prediction to be at $\sim$25% and $\sim$70% respectively.

RegPFA (Breuker et al., 2016) uses a probabilistic finite automaton (PFA) to allow the future hidden state be a probabilistic function of both the previous hidden state and the last observed event (which itself is a probabilistic consequence of the previous hidden state). RegPFA uses an EM algorithm to estimate the model parameters of the PFA. The evaluation uses data from the 2012 and 2013 BPI Challenges. We note that this uses an explicit model representation and the work does not allow the aggregation of other features observed in an event log such as the resources, time, and etc.

Limitations of using explicit model representation include the requirement of prior assumptions about the model and that it can be difficult to extend the predictive model to consider the diverse information observed in an event log. Also, models such as decision trees can suffer from overfitting. With the exception of the use of decision trees, all the mentioned approaches only consider the activity attribute of events. This means the predictive models will have redesigned to consider new event attributes.

On the other hand, model can be built in an implicit manner. This dispenses the need to form any ex-ante assumptions. Existing approaches are focused on the use of different variants of neural networks, e.g., recurrent neural networks (Evermann et al., 2016b; Tax et al., 2016). These approaches have demonstrated promising results. In (Tax et al., 2016), it is shown that a long short term memory (LSTM) predictive model can reach a prediction result of 0.7669 in terms of Damerau-Levenshtein Similarity (a result of 1 means perfect precision). The Damerau-Levenshtein Similarity compares two strings by the number of operations required to transform one into the other. In this case, it is the number of operations required to transform the string representation of the predicted result to the string representation of the executed events in the case. Similarly, in (Evermann et al., 2016b), it is shown that a long short term memory (LSTM) predictive model can reach a precision of 85.9% in predicting the next activity of an ongoing case. However, training neural network models require significant amount of computing resources and available data. This can be challenging a real-life context.

It is clear that research in predictive models that rely on implicit model representations is still quite recent and has been mostly focused on neural network techniques (Tax et al., 2016; Evermann et al., 2016a). By identifying the common challenge in tackling data sparsity between Recommender systems and event prediction in BPM, this paper explores the adaptation of matrix factorization based predictive models which can learn latent features from sparse data. Through leveraging the assumption of an underlying process, event data can be modeled in a meaningful representation. Moreover, additions to the input data are made so that negative information can be incorporated into the training of the predictive model. The proposed model is supported by experimental results which show significant improvement over the general user-item based data representation.

There have been other works in the field of BPM that integrate recommender system techniques. For example, collaborative filtering is used by extending the *Algorithm Recommender System* (ARS) approach (Misir & Sebag, 2013) to recommend process discovery algorithms to users (Ribeiro et al., 2014). However, this work tackles the next event prediction problem rather than recommending algorithms.

4. Preliminaries

This section introduces basic concepts used in later sections of the paper.

4.1. Basic notation

Definition 1 (Multisets). *Let X be a set, a multiset of X is a mapping $M : X \to \mathbb{N}$. $\mathcal{B}(X)$ denotes the set of all multisets over X. Let M and M' be multisets of X. M contains M', denoted $M \geq M'$, if and only if $\forall_{x \in X} M(x) \geq M'(x)$. The union of M and M' is denoted $M + M'$, and is defined by $\forall_{x \in X}(M + M')(x) = M(x) + M'(x)$. The difference between M and M' is denoted $M - M'$ and is defined by $\forall_{x \in X}(M - M')(x) = (M(x) - M'(x))$ max 0.*

Note that $(M - M') + M' = M$ only holds if $M \geq M'$. For sets X and X' such that $X' \subseteq X$, we consider every set X' to be an element of $\mathcal{B}(X)$, where $\forall_{x \in X'} X'(x) = 1$ and $\forall_{x \in X \setminus X'} X'(x) = 0$.

Definition 2 (Sequence). *Let A be a set. A^* denotes the set of all sequences over A and $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ is a sequence of length n. $\langle\rangle$ is the empty sequence and $\sigma_1 \cdot \sigma_2$ is the concatenation of sequences σ_1 and σ_2. For $0 < k < n$, $hd(\sigma, k) = \langle a_1, a_2, \ldots, a_k \rangle$ is the prefix of length k of sequence σ and $tl(\sigma, k) = \langle a_{k+1}, \ldots, a_n \rangle$ is its suffix. Let $\sigma(k)$ denote the element at the k-index of σ.*

For example, for a sequence $\sigma = \langle a, b, c, d, e, f \rangle$, $hd(\sigma, 2) = \langle a, b \rangle$, $tl(\sigma, 2) = \langle c, d, e, f \rangle$ and $\sigma(1) = a$.

Definition 3 (Function domains and ranges). *Let $f \in X \nrightarrow X'$ be a (partial) function. With $dom(f) \subseteq X$ we denote the set of elements from X that are mapped onto some value in X' by f. With $rng(f) \subseteq X'$ we denote the set of elements in X' that are mapped onto by some value in X, i.e., $rng(f) = \{f(x) \mid x \in dom(f)\}$.*

Label	Activity	Note
a	A_Create Application	Creation of new application.
b	A_Denied	Application rejected by bank.
c	O_Accepted	Client acceptance of loan.
d	O_Cancelled	Offer has been canceled due to lack of reply from client.
e	O_Create Offer	Creation of loan offer.
f	O_Sent (mail and online)	Offer sent through mail and the online channel.
g	O_Sent (online only)	Offer sent through the online channel.
h	W_Call incomplete files	Call client due to incomplete documents.
i	W_Call after offers	Call client after sending offer.
j	W_Assess potential fraud	Assess application for fraud

Table 1: Details on the activities of the running example

$$
\begin{aligned}
L = [&\sigma_1 = \langle a, e, f, i, j, b\rangle,\\
&\sigma_2 = \langle a, e, f, i, j, c\rangle,\\
&\sigma_3 = \langle a, e, f, i, h, c\rangle,\\
&\sigma_4 = \langle a, e, f, i, h, b\rangle,\\
&\sigma_5 = \langle a, e, f, i, b\rangle,\\
&\sigma_6 = \langle a, e, g, i, c\rangle,\\
&\sigma_7 = \langle a, e, g, i, d\rangle,\\
&\sigma_8 = \langle a, j, e, g, i, h, c\rangle,\\
&\sigma_9 = \langle a, e, f, i, h, h, c\rangle,\\
&\sigma_{10} = \langle a, e, g, i, h, h, h, h, h, c\rangle]
\end{aligned}
$$

Figure 1: Running example: Event log L_1

Definition 4 (Functions on sequences and multisets). *Let $f \in X \nrightarrow X'$ be a (partial) function, let $\sigma \in X^*$ be a sequence of X, and let $M \in \mathcal{B}(X)$ be a bag of X. With $f(\sigma)$ we denote the application of f on all elements in σ, e.g., if $dom(f) = \{x, z\}$, then $f(\langle x, x, y, y, y, z\rangle = \langle f(x), f(x), f(z)\rangle$. With $f(M)$ we denote the application of f on all elements in M, e.g., if $dom(f) = \{x, z\}$, then $f([x^2, y^3, z]) = [f(x), f(x), f(z)]$.*

4.2. Event logs

Information related to executed events in a process can be extracted as an *event log*. An event log is a set of *traces*. Each trace describes a particular *case*, i.e., a process instance, in terms of the occurred events.

Let $\mathcal{E}$ denote the universe of all identifiable process events. A process event can have different attributes that provide additional information about the event. We denote $\mathcal{U}_A$ as the universe of attribute names. Given an attribute $attr \in \mathcal{U}_A$, we use the partial function $\pi_{attr} : \mathcal{E} \nrightarrow dom(attr)$ to represent the value of $attr$ for a process event from its domain $dom(attr)$. In this paper, we assume that $\mathcal{U}_A$ includes at least the following attributes for a process event $e \in \mathcal{E}$: *case*, which indicates the process instance at which e was executed, *act*, which denotes the activity that e corresponds to and *time* to indicate the timestamp related to the event.

Definition 5 (Trace, Event log). *Let $\mathcal{E}$ denote the universe of all identifiable process events. A trace is a finite sequence of events $\sigma \in \mathcal{E}^*$ such that each event occurs only once and time is non-decreasing, i.e., for $1 \leq i < j \leq |\sigma|$: $\sigma(i) \neq \sigma(j)$ and $\pi_{time}(\sigma(i)) \leq \pi_{time}(\sigma(j))$. Let $\mathcal{U}_T$ be the universe of all possible traces. An event log is a set of traces $L \subseteq \mathcal{U}_T$.*

Events can store different information related to the process such as resources, timestamps, or additional data elements recorded with the event log. These information are referred attributes. Given a trace $\sigma \in L$ and an attribute $attr \in \mathcal{U}_A$, we can get the attribute value of each event in the trace by applying $\pi_{attr}(\sigma) \in attr^*$ to the sequence.

Consider the event log L_1 in Figure 1. Log L_1 is a simplification of the real-life dataset BPIC2012 (van Dongen, 2012) which records the events of a loan application process at a Dutch Financial Institute. Simplification is done by selecting only a few cases and only include events of a subset of activities. Furthermore, for simplicity and space, we refer to each event by its activity attribute with each activity being represented by a corresponding letter label. Table 1 presents the mapping between labels and activity, as well as a short description of each activity.

Consider the trace $\sigma_1 = \langle a, e, f, i, j, b \rangle$. This sequence describes the activities that are executed in a particular loan application. It is initiated with *A_Create Application*. Then a loan offer is created (*O_Create Offer*). The offer is sent to the client by mail and online (*O_Sent (mail and online)*. A bank employee calls the client about the offer (*W_Call after offers*). However, the case is then assessed for fraud (*W_Assess potential fraud*) and the loan application is ultimately denied (*A_Denied*).

Often times, given an ongoing case, it is desirable to have a recommendation of the next activity to execute based on the past events and specific target variables. For example, for the loan application process captured by the running example in Figure 1, it is desirable for the institute to know as soon as possible whether if the loan application should be denied and to ensure that applicants would accept the loan conditions once it is offered by the institute. Moreover, while maximizing this acceptance rate, it is also desirable to minimize the usage of resources. Given an ongoing case $\sigma = \langle a, e \rangle$ with an offer created, the institute can choose to execute various different activities, e.g., activity f (O_Sent (mail and online)), activity g (O_Sent (online only)), activity j (W_Assess potential fraud). An activity recommendation can be made based on the executed activities and the target variable, e.g., occurrence likelihood.

Definition 6 (Next activity prediction). *Let $\mathcal{U}_T$ be the universe of all possible traces and let $L \subseteq \mathcal{U}_T$ be an event log. Let $\sigma \in L$ be a trace from log L and let $\sigma_k^{hd} = hd(\sigma, k)$ and $\sigma_k^{tl} = tl(\sigma, k)$ be the prefix and suffix of length k of trace σ respectively.*

For $1 \leq k < |\sigma|$ and an activity $a \in \mathcal{U}_{act}$, we predict its occurrence likelihood based on the k prefix of trace σ, σ_k^{hd}, $pred_{act}(\sigma_k^{hd}, a) \in [0, 1]$.

Consider trace $\sigma_1 \in L_1$ in Figure 1. For $k = 2$, the activities f, g and j are evaluated as candidates to be recommended for the trace prefix ${\sigma_1}_k^{hd}$. Using a predictive model, we find that $pred_{act}(\sigma_k^{hd}, f) > pred_{act}(\sigma_k^{hd}, g) > pred_{act}(\sigma_k^{hd}, j)$. This means that activity f should be recommended based on its occurrence likelihood.

Definition 7 (Activity recommendation). *Let $\mathcal{U}_T$ be the universe of all possible traces and let $L \subseteq \mathcal{U}_T$ be an event log. Let $\sigma \in L$ be a trace from log L. For $1 \leq k < |\sigma|$, let $\sigma_k^{hd} = hd(\sigma, k)$ be the prefix of length k of trace σ.*

- *An activity $a \in \mathcal{U}_A$ is recommended if the prediction score $pred_{act}(\sigma_k^{hd}, a)$ is higher or equal to the prediction score of any other possible activity prediction $pred_{act}(\sigma_k^{hd}, a')$.*
- *$rec_{act}(\sigma, k) \in \mathcal{E}^* \rightarrow \mathcal{U}_A$ is a deterministic mapping that assigns any valid case prefix to an activity recommendation.*

Consider again trace $\sigma_1 \in L_1$ in Figure 1. For $k = 2$, one possible k activity recommendation based on occurrence likelihood could be $rec_{act}(\sigma, 2) = f$, i.e., activity f is recommended. Coincidentally, activity f is the next activity executed at case σ_1. We note that a model could be trained to recommend activities based on different target variables, e.g., execution time and cost.

In the following section, we introduce factorization models as the class of models to be used for activity recommendations on event data. Specifically, we focus on factorization machine which is a general predictor which works with any real valued feature vector.

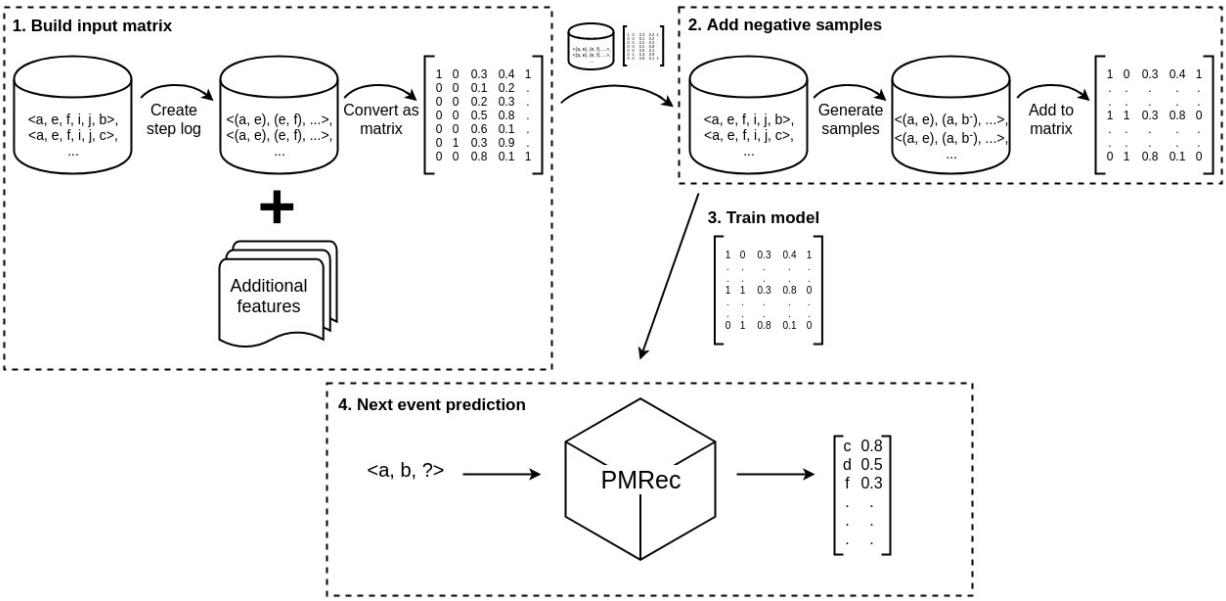

Figure 2: Overview of PMRec

5. Factorization machine for process prediction

In this section we describe in detail our proposed *Process Mining Recommender* (PMRec hereinafter). As shown in Figure 2, there are three main parts to the process. First, an event log consisting of traces is converted into a "step log" representation that will allow cases to be represented as vectors and a log as an input matrix. Other data attributes recorded with the event data can also be added as additional features to the matrix. Second, to tackle the underlying limitation of event logs in lacking negative samples, artificial negative samples are added to the input matrix to increase prediction accuracy. After training the factorization machine predictive model PMRec with the input matrix, next activity predictions can be made for running cases of the corresponding process.

In the following, we first introduce the novel representation of process events. Second, we present factorization machine as the prediction model from recommender system to be adapted for the next activity prediction task. We also explain the computation details of the prediction task. Third, we present the addition of negative samples to deal with the underlying limitation of event logs in terms of data completeness to increase prediction accuracy.

5.1. Process event representation

Typically general recommender systems disregard any sequential information on users' item consumption and represent the consumed items as a vector where consumed items are assigned with their corresponding rating Rendle et al. (2010). Conversely, MF-based sequential recommenders such as the FPMC model consumed items sequences as tensors where each slice of the tensor marks the probabilities of consuming particular items given the history of consumed items for each user.

While modeling sequential behavior using Markov chain methods is appropriate in the recommender system context where the average user only has one or two previous actions, this is much more difficult in a business process context where the average case lengths can be long. Moreover, the execution order of activities in a process is often much more important since different execution orders can affect related decision makings such as resource allocation. This motivates the emphasis in directly capturing the dependencies between activities in the data representation so that information of the control flow, i.e., the execution order of activities, can be incorporated into the predictive model. One idea is to model an event sequence as a series of "steps".

Let $L \subseteq \mathcal{U}_T$ be an event log. Each trace $\sigma \in L$ can be described by a set of steps where each step is composed of a sequence of events $e \in \sigma$ that occurred in the trace.

Definition 8 (Process step). *Let $E \subseteq \mathcal{E}$ be a set of events. For $k > 1$, a k-process step is a k tuple $(e_1, e_2, \ldots, e_k)$ where each element $e_i \in E$ is an event. A k-process step set $S^{(k)} = \{(e_1, e_2, \ldots, e_k) \mid e_i \in E \text{ for all } 1 \leq i \leq k\}$ contains all k-process step composed of events in E.*

For a process with an activity set $A \subseteq \mathcal{U}_{act}$ where each activity $a \in A$ can be executed in different lifecycles of the set $Lf \subseteq \mathcal{U}_{lifecycle}$, it has a set of possible events $E \subseteq \mathcal{U}_{\mathcal{E}}$ where $|E| = |A| \times |Lf|$. E contains all events with all possible combinations of attributes $act \in A$ and $lifecycle \in Lf$. We denote $S^{(2)}_{activity,\ lifecycle}$ as the set of 2-process steps from E. For simplicity, we consider events with only activity attributes and refer to $S^{(2)}_{activity}$ as $S^{(2)}$ for the rest of the paper.

Definition 9 (Step case, Step log). *Let $\mathcal{E}$ denote the universe of all identifiable process events and let $\mathcal{U}_T$ be the universe of all possible traces. Let $L \subseteq \mathcal{U}_T$ be an event log. Let $A \subseteq \mathcal{U}_{act}$ be the set of possible activities for the underlying process of log L. Let $E \subseteq \mathcal{E}$ contains all events with all attributes $act \in A$. Let $S^{(2)} = E \times E$ be the 2-process step set.*

Let $\sigma \in L$ be a particular case of length $|\sigma| = n$. $step(\sigma, S^{(2)}) = \{\langle e_i, e_j \rangle \mid \exists_{s_k \in S^{(2)}} \pi_{act}(s_k) = \pi_{act}(\langle e_i, e_j \rangle) \wedge e_i \in \sigma \wedge e_j \in \sigma \text{ for all } 1 \leq i < j < n\}$ is a step version of case σ, i.e., a step case.

$step(L, S^{(2)}) = \{step(\sigma, S^{(2)}) \mid \sigma \in L\}$ is the step version of log L, i.e., a step log.

We note that an artificial start and end events can be appended to the ends of each case so that all cases in the log can be represented as a step case.

Given a step log $step(L, S^{(2)})$, it is clear that it can be represented as a matrix of m rows and n columns where $m = |L|$ is the number of cases in log L and $n = |S^{(2)}|$ is the number of possible steps in 2-process step set $S^{(2)}$.

5.2. Factorization Machine

Factorization Machines (FM) is a model class that combines the advantages of Support Vector Machines (SVM) with factorization models (Rendle, 2010). It is a general predictor that works with any real valued feature vector. In the field of recommender systems, FMs can be seen as the generalized model of the factorization models such as Matrix Factorization, SVD++, and Factorized Personalized Markov Chains (FPMC) due to its ability to mimic these models by specifying the input data (i.e., the feature vectors).

The mechanism of FMs is that interactions between variables are modeled using factorized parameters. This means that interactions can be estimated even in problems with high sparsity. Furthermore, FMs are computationally efficient since the model equation can be reduced to linear complexity in both k (the number of factors) and n (the number of features). This means that prediction time is linear and the number of parameters to learn during the training phase is reduced.

Definition 10 (Factorization machine (Rendle, 2010)). *Let $\boldsymbol{x} \in \mathbb{R}^n$ be a feature vector of dimension n. Let $\langle \cdot, \cdot \rangle$ be the dot product of two vectors of size $k \in \mathbb{N}_0^+$:*

$$\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}$$

The model equation for a factorization machine of degree $d = 2$ is defined as:

$$\hat{y}(\boldsymbol{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j$$

where $w_0 \in \mathbb{R}$ is the global bias, $\boldsymbol{w} \in \mathbb{R}^n$ models the strength of the i-th variable and $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ models the interaction between the i-th and j-th variable.

Note that k is a hyperparameter that defines the dimensionality of the factorization. This means that instead of using the model parameter $w_{i,j} \in \mathbb{R}$ for the interaction between two features, it is estimated in a factorized manner. This is the key point that reduces the complexity for learning the model parameters from $\mathcal{O}(n^2)$ to $\mathcal{O}(kn)$.

The model parameters w_0, $\mathbf{w}$ and $\mathbf{V}$ can be learnt by gradient descent methods such as stochastic gradient descent (SGD). The gradient of the FM model, considering up to 2-way interactions, is:

$$\frac{\partial}{\partial\theta}\,\hat{y}(\mathbf{x}) = \begin{cases} 1 & \text{if } \theta \text{ is } w_0 \\ x_i & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{n} v_{j,f} x_j - v_{i,f} x_i^2 & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

Given that cases in an event log can be represented as in vector form, process steps can be used as model features in FMs to predict occurrence likelihood of activities at ongoing cases.

However, one common limitation of training datasets is the lack of negative samples. This limitation can be addressed by defining a set of artificial negative samples for each case prefix, as introduced in the Bayesian Personalized Ranking model (Rendle et al., 2009).

5.3. Artificial negative samples

As previously mentioned, one limitation of the datasets used for training predictive models is the lack of negative feedback information. This is present in event logs as well; prohibited events are not recorded in the event data. This is made more complicated with the fact that unobserved process behavior does not mean that the behavior is prohibited but might just have not happened yet.

However, one approach to address this is to make an assumption of completeness where the event log is assumed to contain all observed behavior to a certain degree. In (Goedertier et al., 2009; vanden Broucke et al., 2014), it was reported that by making an explicit completeness assumption, negative events can be generated as the events that did not occur at a certain point in time of a case. In (Goedertier et al., 2009), negative events are used to discover process models that are more precise from dataset related to a process. We adopt the same idea to process steps to provide negative information to the predictive model.

Definition 11 (Negative process step). *Let $\mathcal{E}$ denote the universe of all identifiable process events and let $\mathcal{U}_T$ be the universe of all possible traces. Let $L \subseteq \mathcal{U}_T$ be an event log. Let $A \subseteq \mathcal{U}_{act}$ be the set of possible activities for the underlying process of log L. Let $E \subseteq \mathcal{E}$ contains all events with all activity attributes $a \in A$. Let $S^{(2)} = E \times E$ be the 2-process step set.*

Let $\sigma \in L$ be a particular case of length $|\sigma| = n$. For $1 \leq k < n$, the set of negative events for case prefix σ_k^{hd} is $\{\langle e_1, e_2\rangle \mid \pi_{act}(e_1) = \pi_{act}(\sigma(k)) \wedge \pi_{act}(e_2) \in A \setminus \{\pi_{act}(\sigma(k+1))\}\}$, i.e., the set of possible steps except for the observed next process step.

Consider case $\sigma_5 = \langle a, e, f, i, b\rangle$ in Figure 1. The step case representation is $step(\sigma_5, S^{(2)}) = \{(a, e), (e, f), (f, i), (i, b)\}$. For the case prefix ${\sigma_5}_2^{hd} = \langle a, e\rangle$, the set of negative steps consists of $\{\langle e, a\rangle, \langle e, b\rangle, \langle e, c\rangle, \langle e, g\rangle, \langle e, h\rangle, \langle e, i\rangle, \langle e, j\rangle\}$.

5.4. Training input matrix

As previously mentioned, events can store different information related to the process. These information can be specified as feature vectors for a FM model for prediction tasks. One of the most important features of event data related to processes is the causal relations between different activities. This is often described through business process modeling notations such as BPMN (Weske, 2012), Petri Nets (Murata, 1989), and YAWL (ter Hofstede et al., 2010). With process steps, it is possible to explicitly specify causal relations between activities of a process model as feature vectors at a FM model and relate information event data to these features using step cases. This forms a fundamental part of PMRec where user can leverage information from prior process models to improve predictive prediction of PMRec. However, in this paper we would like to evaluate the performance of the basic predictive model and therefore we do not select features using a prior process model.

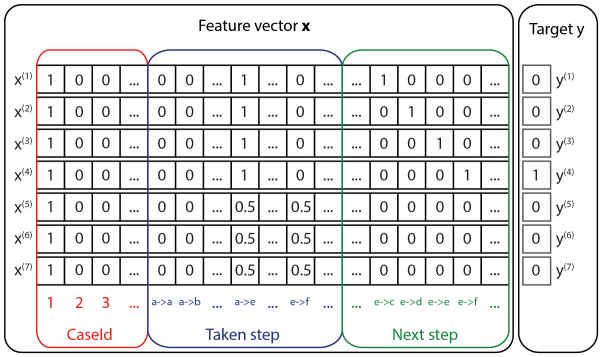

Figure 3: Example training input matrix of case σ_1 in Figure 1 under input matrix variant "Step"

Other than the causal relations between activities, other information such as time, number of executed events, and resources can be incorporated into PMRec. Consider case σ_1 in Figure 1. The step case representation would be $step(\sigma_1, S^{(2)}) = \{\langle a, e\rangle, \langle e, f\rangle, \langle f, i\rangle, \langle i, j\rangle, \langle j, b\rangle\}$. This case can incrementally added into the input matrix step by step.

Consider a negative sample size of 3, Figure 3 shows how feature vectors can be created for the above training input. Here, there are $|CaseId|$ binary indicator variables (red) that represent the current case (caseID 1). The next group of indicator variables (blue) contains information about the taken steps of the case. For each row, the variables in this group are normalized so that they sum up to 1. For example, for the first four feature vectors, $x^{(1)}, \ldots, x^{(4)}$, the set of taken steps contains only $\{\langle a, e\rangle\}$. Finally, the last group of binary indicator variables (green) records the possible next step. For example, given the taken step $\{\langle a, e\rangle\}$, the actual next step is $\langle e, f\rangle$. This means that a 1 should be assigned to the variable $\langle e, f\rangle$. We also pick three negative samples $\{\langle e, c\rangle, \langle e, d\rangle, \langle e, e\rangle\}$ from the set of possible steps $\{\langle e_1, e_2\rangle \mid \pi_{act}(e_1) = e \wedge \pi_{act}(e_2) \in A \setminus \{f\}\}$. As such, the feature vectors $x^{(1)}, x^{(2)}, \ldots, x^{(4)}$ are created. For the target $y^{(1)}, y^{(2)}, \ldots, y^{(4)}$, 1 is assigned if the corresponding feature vector is associated with the actual next step. Otherwise, a 0 is assigned.

Next, we present the approach to evaluate the performance of PMRec.

6. Experimental Method

In this section, we present the datasets and methodology used for the evaluation of the proposed approach. To obtain evaluation results that are comparable to the state-of-the-art for the next event prediction task, we use the same datasets (the BPI Challenge 2012 and 2013 datasets (van Dongen, 2012; Steeman, 2013a,b)) and similar experimental approaches as detailed in (Evermann et al., 2016b; Tax et al., 2016).

6.1. Data

The BPI Challenge 2012 dataset (van Dongen, 2012) is a real-life dataset from a Dutch Financial Institute. Specifically, the event data are related to a loan application process such that each case is of a particular loan application. The dataset contains events related to 13,087 cases. Furthermore, it can be separated into three subprocesses. Subprocess A relates to the states of the application itself, subprocess O relates to the states of an offer communicated to the customer, and subprocess W relates to the states of work items that occur during the approval process. The first letter of each task name can be used to identify the subprocess that it relates to. The BPI Challenge 2013 datasets (Steeman, 2013a,b) are real datasets from the incident and problem management process at Volvo IT Belgium with 7,554 and 2,306 cases respectively. These incidents and problems are related to the IT-services delivered and/or operated by Volvo IT such as incidents that may disrupt plant production (Vehicles / Product) and result in lost units of production. The problem management process dataset is comprised of two separate datasets where one is related to cases whose problem is still open at the time of data collection and the other is related to cases with closed

Dataset	Number of activities	Number of events
BPI2013.Incidents	13	65533
BPI2013.Problems	7	7460
BPI2012 (completion events)	24	164506
BPI2012 (all events)	36	262200
BPI2012.W (completion events)	6	72413
BPI2012.W (all events)	19	170107
BPI2012.A	10	60849
BPI2012.O	7	31244

Table 2: Characteristics of datasets used in the experimental evaluation

problems. We note that there are 465 cases which reside at both open and closed datasets. For these overlapping cases, since the record at the closed dataset encompasses the record at the open dataset, we used the record at the closed dataset for the experimental evaluations. There are 1,841 cases in the combined problem management process dataset. The number of activities and events per dataset is shown in Table 2.

Similar to (Evermann et al., 2016b), we evaluated the BPI 2012 dataset by the three subprocesses and as combined datasets as well. For the events related to the W subprocess, events can be associated with three different lifecycles: "Start", "Schedule", and "Complete". The different lifecycles records the different stages that each work item goes through, e.g., "Start" relates to the moment when the work item is obtained by a resource. For the rest of the datasets, all events are only associated with the "Complete" lifecycle. As such, we created two datasets for both the subprocess W dataset and the combined BPI 2012 dataset. In the first version, only events associated with the "Complete" lifecycle, i.e., activity completion events, are included and in the second version, events are identified by the concatenation of both the activity and lifecycle, e.g., "Activity A+Start" This allows a more detailed description of the recorded events and provides more information to the prediction model. The case information as observed in the datasets are converted to input matrix to PMRec using the previously defined data representations.

6.2. Building input matrix

Table 4 shows the make-up of each input matrix variant. Each feature is composed of a set of possible values, e.g., the *CaseId* feature is composed of all the case IDs in the event log. The number of possible values per feature is shown in Table 3. In order to predict for unseen cases in the validation and test set, the case IDs in both sets are included as possible values for the *CaseId* feature. We note that the case ID of each case is only a unique identifier used to differentiate between different ongoing and historical cases. In a real-life implementation, the *CaseId* feature can be periodically updated so that new columns are added in anticipation of the caseIDs of future cases. As such, we emphasize that no case information from the validation and test sets are included into the training of the prediction model. We order the possible values of each feature in ascending order to maintain the same structure for the same feature across all variants (string values are first converted into integer). The different column features are ordered as indicated in Table 4. As such for the input matrix variant "Act" of dataset BPI2013.Incidents, the input matrix would have $7554 + 13 + 169 + 5 = 7769$ columns.

Each case is first converted into its step case representation. Consider the input matrix variant *Step* and case $\sigma_6 = \langle a, e, g, i, c\rangle$ in Figure 1. Its 2-process step representation would be $step(\sigma_6, S^{(2)}) = \{\langle a, e\rangle, \langle e, g\rangle, \langle g, i\rangle, \langle i, c\rangle\}$. Next, it is incrementally added as feature vectors starting from the first step with each increment being the next taken step. For example, at the first two steps, the taken steps is $N^2_{\sigma_6} = \{\langle a, e\rangle, \langle e, g\rangle\}$ and the next taken step is $s = \langle g, i\rangle$. A feature vector x with $n = |\text{CaseId}\cup\text{Taken steps}\cup\text{Next step}|$ components can then be built as follows:

Dataset	Number of possible values			
	CaseId	Taken / Next act	Taken / Next step*	Impact
BPI2013.Incidents	7554	13	169	5
BPI2013.Problems	1841	7	49	5
BPI2012 (completion events)	13087	24	576	na
BPI2012 (all events)	13087	36	1296	na
BPI2012.W (completion events)	13087	6	36	na
BPI2012.W (all events)	13087	19	361	na
BPI2012.A	13087	10	100	na
BPI2012.O	13087	7	49	na

* Steps are of step size 2 and is comprised of only the activity attribute

Table 3: Number of possible values per feature

Input Matrix	Features
Act	CaseId, Taken acts., Next act.
Step	CaseId, Taken steps, Next step
StepLc	CaseId, Taken steps, Next step
StepImpact	CaseId, Taken steps, Impact, Next step

Table 4: Features for different input matrix variants

$$x_j := \begin{cases} 1, & \text{if } j = 6 \vee j = s \\ 1/|N^2_{\sigma_6}|, & \text{if } j \in N^2_{\sigma_6} \\ 0, & \text{otherwise.} \end{cases}$$

In essence, values are assigned to components corresponding to the observed case information so that each feature sums to 1. The target variable value is set as 1. One limitation of observed dataset is that only positive information is provided; negative events are used to provide negative information for the training of PMRec.

6.3. Adding negative steps

As recalled, by making an explicit completeness assumption, negative events can be induced from the event log. Negative events serve as negative information on a particular event which cannot occur at a particular position in an event sequence. For simplicity, we sample from the set of unobserved steps as negative events for each partial case. Consider again case σ_6 in Figure 1. As previously shown, at the first two steps, the taken steps is $N^2_{\sigma_6} = \{\langle a, e\rangle, \langle e, g\rangle\}$ and the next step is $s = \langle g, i\rangle$. By using the set of possible steps following the last executed activity g, $S_g = \{\langle g, e_i\rangle \mid e_i \in A\}$, negative steps are taken from $S_g \setminus \{\langle g, i\rangle\}$.

For each negative step, we construct a separate feature vector, replacing the observed next step with the negative step and assigning 0 to the target variable rather than 1. This means that if 3 negative steps are sampled, 4 feature vectors would be constructed instead of 1.

Given an ongoing case, prediction is done by predicting all possible steps then returning the top ranked result as the most probable next event.

6.4. Predicting next event

To predict the next event of an ongoing case, it is first converted into feature vectors with all possible next steps. Consider again case σ_6 in Figure 1. As before, we consider the first two steps of the case such that taken steps is $N^2_{\sigma_6} = \{\langle a, e\rangle, \langle e, g\rangle\}$. Assuming that this is an ongoing case and that the actual next

Parameter	Value	Note
Gradient learning		
gradient_descent	als	Gradient descent method to learn model parameters
init_stdev	0.1	Sets the std. deviation for the initialization of the parameter
l2_reg	0	L2 penalty weight for all coefficients
l2_reg_V	0.01	L2 penalty weight for linear coefficients
l2_reg_w	0.5	L2 penalty weight for pairwise coefficients
min_iter	2000	Minimum number of iterations for training
max_iter	10000	Maximum number of iterations for training
random_state	123	The seed of the pseudo random number generator that initializes the parameters
rank	20, 50, 100	The rank of the factorization used for the second order interactions
Input matrix		
step_size	2	see Definition 8
negative_step	3	see Definition 11

Table 5: Configurations for learning parameters of PMRec.

step ($\langle g, i \rangle$) is unknown, we construct $|A|$ feature vectors where A is the set of activities for this process. For each feature vector, we set the next step value as $s \in S_g$ where $S_g = \{\langle g, e_i \rangle \mid e_i \in A\}$ is the set of possible next steps following the last executed activity g.

Predicting using PMRec would result in an output vector $\hat{y}$ where each component $\hat{y}_i$ corresponds to the prediction for a particular possible next step. The $\hat{y}_i$ with the highest value can be interpreted as the most probable next step. Finally, we output the last activity of the top-ranked next step as the predicted next event.

We note that for the datasets with all events, i.e., BPI2012 (all events) and BPI2012.W (all events), the next predicted next event, e, can be identified by solely the activity or by the concatenation of the activity and lifecycle. In the related work (Evermann et al., 2016b), the predicted next event is identified by the concatenation of both the activity and lifecycle.

The performance of PMRec is evaluated using a validation approach.

6.5. Evaluation method

In a validation evaluation approach, a validation set is used to prevent the over-training of the prediction model before performing evaluation using the test set. By convention, we use a 60/20/20 split on the cases of each dataset where 60% of the cases are included into the training set and 20% are included into the validation and test sets respectively.

Learning is monitored by the RMSE of the validation set and a maximum and minimum number of iterations are defined to ensure termination of model training. The model configurations of PMRec is shown in Table 5. To evaluate the results, precision is used to measure the proportion of top-ranked next step predictions that correspond to the actual next step, i.e., the number of correct next step predictions divided by the total number of next step predictions.

7. Experimental results

In our experiments we found that all datasets quickly converge close to the optimal precision and that the use of the validation set significantly helps prevent the overfitting of PMRec. As shown in 4, RMSE of the validation set converges to a low value very quickly and continues to decrease gradually with more iterations after a spike at the early stage. The drop in the RMSE of the validation set at the early stage coincides with the drop in RMSE of the train set.

We first show the effects that each of the proposed techniques has on the performance of PMRec.

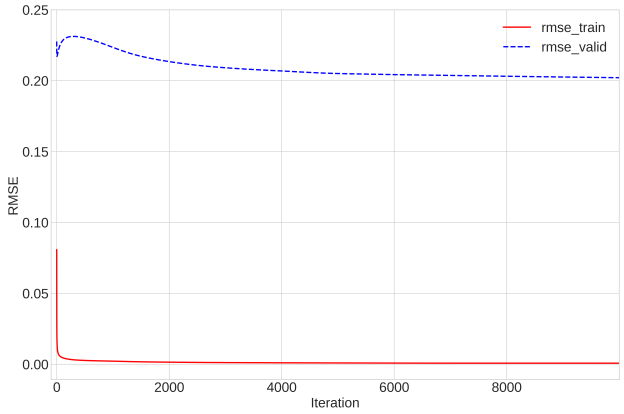

Figure 4: RMSE of train and validation set for dataset BPI2012.A using *Step* input matrix (can be viewed in black & white)

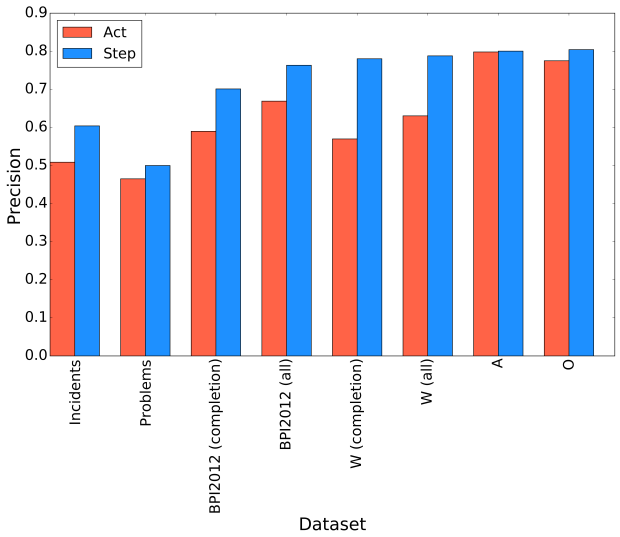

Figure 5: Precision comparison between input matrix variant "Act" and "Step" per dataset (can be viewed in black & white)

7.1. The effect of process steps

Representing cases with process steps gives significantly better performance than by simply taking into account the previous events of an ongoing case. Figure 5 shows the highest obtained precision under the input matrix variants, *Act* and *Step*, for each of the dataset. It is clear that the performance of PMRec using the *Step* input matrix variant is significantly better than the performance using the *Act* input matrix variant.

As shown in Figure 5, the performance of PMRec using the input matrix variant *Step* is better than the performance using the *Act* input matrix variant across all datasets. Furthermore, for the datasets which are more complex in terms of the underlying process, using the *Step* input matrix variant can give significantly better performances. For example, for BPI2012.W (completion), precision yielded by using the *Step* input matrix variant is 21.05% higher than when the *Act* input matrix variant is used. As recalled, the subprocess W tracks the work items related to loan applications. These work items are executed by resources (employees) and thus both the type and number of work items as well as the order in which they are executed vary significantly per case. Conversely, the subprocess A tracks the status of loan applications and therefore has less variability. This is reflected by the comparatively lesser performance differences between

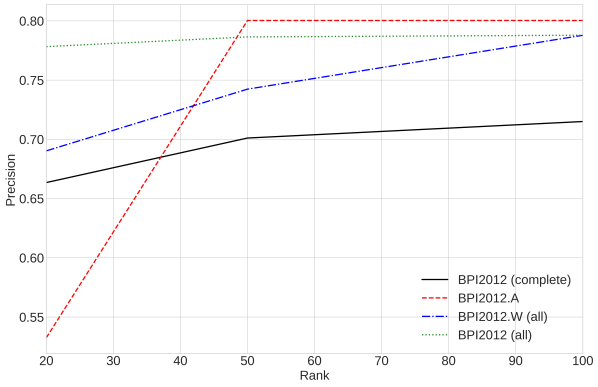

Figure 6: Effect of different rank values on precision per dataset (can be viewed in black & white)

the input matrix variant *Act* and *Step* for BPI2012.A.

7.2. The effect of rank

As previously mentioned, FM models all interactions between variables using factorized parameters. The rank value at the model configuration determines the number of latent factors required to describe the variable interactions. Intuitively, for input features vectors with complex and numerous interactions, higher rank would yield better precision, but over certain threshold results usually converge, as shown in (Koren et al., 2009; Larraín et al., 2015).

Figure 6 shows the effect of rank value on precision result per dataset. For smaller and more simple datasets such as BPI2012.A, increasing the number of rank from 20 to 50 improves precision dramatically but further increase in rank has little effect on precision. Conversely, for datasets that are larger and more complex such as BPI2012.W (all events), precision continues to improve as rank increases. The convergence of precision for the simpler datasets such as BPI2012.A is inline with the findings in the literature. However, the lack of convergence for some of the datasets suggests that higher rank values could be configured.

7.3. The effect of step size

We also experimented with how varying step size can affect performance. In terms of computation, increasing step size exponentially increases number of columns of the input matrix. Consider the dataset BPI2012.O which has 8 unique event types. With step size of 2, there would be 8^2 possible steps and at step size 3, there would be 8^3 possible steps. As such, due to computational requirement, step size 3 and 4 are only performed with selected datasets. As previously shown in Section 7.1, including control flow information by using the process step representation with a step size of 2 increases precision as compared to simply representing a case as a vector of occurred activities. One would expect to further improve precision by increasing step size so that non-local dependencies between activities can be incorporated as well (van der Aalst, 2010). As shown in Table 6, contrary to intuition, further increasing step size from 2 does not seem to increase precision. One possible way for further testing whether non-local dependencies can be captured by increasing step sizes could be through artificially adding non-local dependencies into real-life or synthetic datasets.

7.4. The effect of negative steps

One main contribution is the adaptation of negative events to provide negative information at training. Our experiments show that this is crucial in yielding results that are comparable to state-of-the-art approaches such as neural networks. Similar to increasing the step size, increasing the number of negative steps can also significantly increase the size of the input matrix. Therefore, experiments that vary the number of negative steps are only conducted for selected datasets. As shown in Table 7, increasing number of

Dataset	Step size		
	2	3	4
BPI2013.Incidents	0.604	0.598	0.599
BPI2013.Problems	0.5	0.488	0.494
BPI2012.W (completion events)	0.780	0.780	0.779
BPI2012.A	0.800	0.800	0.800
BPI2012.O	0.795	0.804	0.791

Table 6: Effect of varying step size to performance using *Step* input matrix variant

Dataset	Number of negative steps			
	0	3	5	All
BPI2013.Incidents	0.378	0.604	0.606	0.605
BPI2013.Problems	0.483	0.494	0.5	0.5
BPI2012.W (completion events)	0.634	0.780	0.779	0.779
BPI2012.A	0.297	0.800	0.800	0.780
BPI2012.O	0.740	0.802	0.804	0.803

Table 7: Effect of varying number of negative steps to performance

negative steps from 0 to 3 improves precision for all datasets. The column "All" refers to when all possible steps that are not the actual next taken step are included as negative steps.

As presented in (Goedertier et al., 2009), the inclusion of negative events makes a completeness assumption that the dataset contains all possible behavior of the underlying process. This can become problematic since the dataset often does not include all possible behavior and therefore an assumed negative step from the training set can actually be an observed step in the validation and test sets. This tradeoff can be seen in Table 7 where the benefit of negative steps reduces and even reverses for 5 and "All" negative steps.

7.5. The effect of impact level

One key advantage of using the Factorization Machine model is that different features can be easily incorporated into the prediction without significant modification to the model. We showcase this with the experiments on the datasets BPI2013 using the input matrix variant *StepImpact*. As previously mentioned, this dataset records the process of incident and problem management at Volvo IT Belgium. Each case refers to the processing of a particular incident (BPI2013.Incidents) or problem (BPI2013.Problems). Impact refers the criticality of an incident and is measured by the number of people or systems affected (Steeman, 2013a,b). Clearly, the impact of each case could potentially influence the unfolding of a case. The impact of each case can be easily included into each feature vector such that the possible impact values (Major, High, Medium, Low) are individual feature columns. Component corresponding to the impact of the case that the feature vector is representing is assigned with the value 1 and 0 is assigned to the other components.

As shown in Table 8, the inclusion of the impact feature improved precision for BPI2013.Problems. Precision improved 1.2% from 0.5 to 0.512. However, precision decreased 0.002% for BPI2013.Incidents. This suggests that there are some influence from the impact classification of a case on how that case unfolds. Overall, the results suggest that finding the appropriate additional feature that can significantly improve prediction precision is challenging. There are many feature selection techniques in the literature that can be applied for this task Tan et al. (2005). However, an exploration of the feature selection techniques to improve prediction precision is outside the scope of this paper.

7.6. Comparison to state-of-the-art techniques

Table 8 presents an overview of the experimental results. From the list of precision results associated with different rank values, we select the top precision result. With the exception of the datasets BPI2013.Incidents

Dataset	Precision							
	RegPFA	LSTM	LSTM (Act + lifecycle)	Act	Step	StepLc	StepLc (Act + lifecycle)	StepImpact
BPI2013.Incidents	0.714	0.735	-	0.529	0.612	-	-	0.610
BPI2013.Problems	0.690	0.628	-	0.465	0.5	-	-	0.512
BPI2012 (completion events)	-	0.788	-	0.597	0.746	-	-	-
BPI2012 (all events)	-	-	0.859	0.676	0.773	0.825	0.839	-
BPI2012.W (completion events)	0.719	0.658	-	0.570	**0.780**	-	-	-
BPI2012.W (all events)	-	-	0.832	0.680	0.859	0.874	**0.873**	-
BPI2012.A	0.801	0.832	-	0.798	0.800	-	-	-
BPI2012.O	0.811	0.836	-	0.775	0.804	-	-	-

Table 8: Experimental results compared to RegPFA (Breuker et al., 2016) and LSTM (Evermann et al., 2016b)

and BPI2013.Problems, it is clear that PMRec achieves performances that are comparable to the state-of-the-art approaches. For the dataset BPI2012.W (completion events), PMRec achieves a precision of 78% which is ∼6% higher than the next highest precision from RegPFA presented in (Breuker et al., 2016). For BPI2012 (completion events), BPI2012.A, and BPI2012.O, precision results are all less than 5% from the precision results achieved by the neural network predictive model presented in (Evermann et al., 2016b).

For the datasets BPI2012 (all events) and BPI2012.W (all events), the LSTM model was used to predict both the corresponding activity and lifecycle of the next event. To compare with the corresponding results, PMRec was also used to predict both the corresponding activity and lifecycle of the next event under the StepLc configuration. For the dataset BPI2012.W (all events), PMRec achieves a precision of 87% which is ∼4% higher than the precision from the LSTM model. For the dataset BPI2012 (all events), PMRec achieves a precision of 84% which is 5% less than the precision of the LSTM model.

As mentioned, for the datasets BPI2013.Incidents and BPI2013.Problems, PMRec performs worse than the state-of-the-art approaches. At worst, the precision achieved by PMRec is ∼10% worse than the precision achieved by RegPFA and LSTM. One possible reason is the number of cases. While the BPI 2012 dataset has 13,087 cases, BPI2013.Incidents has 7,554 cases and BPI2013.Problems has 1,841 cases. For the PMRec approach, having less cases means that there are less rows in the input matrix; it is difficult to learn the interactions between features from an input matrix with little rows. This showcases a limitation of PMRec in requiring a sufficient amount of event data in order to achieve good precision. In contrast, the number of cases have a much more minor impact on the performances of the state-of-the-art approaches.

Discussion. Compared to the neural network methods, the proposed PMRec model based on FM has less parameters to configure and to learn during the training phase. The good performances of PMRec therefore motivates the use of this comparatively simpler model. Moreover, the experimental results suggest that the characteristics of the predictive task in a process context are favorable to PMRec. The results suggest that good prediction precision can be achieved without modeling long term dependencies and incorporating high-order interactions between features, i.e., high-level latent features that describe the interactions between latent features. In fact, this is reported in the paper that presented the LSTM model where there are little differences in the prediction precision between a model that is trained on the previous 20 events to predict a particular event as compared to a model that is trained on the previous 10 or 5 events (Evermann et al., 2016b). However, further experiments need to be performed with more synthetic and real-life datasets to provide an understanding on whether these observations can be generalized.

8. Conclusions and future work

This paper introduced the use of factorization model for process prediction. The presented approach uses FM as the base predictive model and includes elements of an underlying process through process steps. The approach does not rely on explicit process models but known features of the process model, e.g., relations between activities, can be easily added to improve performance. This means that the model lies between adopting an implicit process representation and an explicit process representation.

Moreover, the choice to investigate factorization machine as the predictive model is motivated by the common challenge in having to handle data sparsity in the field of recommender systems and next event

prediction in BPM. Other than the selection of factorization machines as a potentially good predictive model, this paper presents PMRec, a framework that integrates factorization machines so that it is adapted for the task of predicting the next activity of a running case in a process. In order to explicitly model the control flow information between activities, this paper proposed a process step representation that represents a given case as a sequence of occurred activity k-tuples. Furthermore, to tackle the lack of negative samples in the event data, artificial negative samples are added into the input matrix. Experimental results have shown that precision results obtained using the presented approach are comparable to and at times exceeding the state-of-the-art approaches. Moreover, the results showed that including the control flow information by using the process step representation gives better prediction precision than the straightforward representation of a case as a vector of occurred activities. The experimental results also showed the benefits of training the predictive model using negative samples. Overall, this paper shows promising results of using recommender system techniques for predictive tasks in the domain of business processes.

As future work, we look to perform more experiments on other synthetic and real-life datasets to provide further understanding on the observations that we have gained in this paper. We also look to further explore the use of different feature selection techniques to select appropriate features. Known features of the underlying process model can be incorporated into PMRec, since FM allows including new features effortlessly. As such, process models discovered using process discovery algorithms can be used to limit the possible next step values given the last executed activity. Also, we look to modify PMRec for other prediction tasks such as predicting the time until case completion. Furthermore, the real valued feature vector representation of event data can be applied in other predictive models such as neural networks. Lastly, one identified challenge in the use of neural network is the lack of training data. Negative information generated using negative steps can not only increase the amount of training data but provide negative information that can improve precision.

Disclosure statement

The authors state that there are no interests to declare for this paper.

Acknowledgements

This work was supported by *CONICYT-PCHA / Doctorado Nacional / 2017-21170612*; *CONICYT/ Fondecyt grant 11150783*; *FONDECYT Iniciación 11170092*; *CONICYT Apoyo a la Formación de Redes Internacionales Para Investigadores en Etapa Inicial REDI170136*; the *Vicerrectoría de Investigación de la Pontificia Universidad Católica de Chile / Concurso Estadías y Pasantías Breves 2016*; and the *Departamento de Ciencias de la Computación UC / Fond-DCC-2017-0001.*

References

van der Aalst, W. M. P. (2010). Process discovery: Capturing the invisible. *IEEE Comp. Int. Mag.*, *5*, 28–41.

van der Aalst, W. M. P. (2016). *Process Mining - Data Science in Action, Second Edition*. Springer. URL: `https://doi.org/10.1007/978-3-662-49851-4`. doi:`10.1007/978-3-662-49851-4`.

Breuker, D., Matzner, M., Delfmann, P., & Becker, J. (2016). Comprehensible predictive models for business processes. *MIS Quarterly*, *40*, 1009–1034.

vanden Broucke, S. K. L. M., Weerdt, J. D., Vanthienen, J., & Baesens, B. (2014). Determining process model precision and generalization with weighted artificial negative events. *IEEE Trans. Knowl. Data Eng.*, *26*, 1877–1889.

Cai, L., Xu, J., Liu, J., & Pei, T. (2018). Integrating spatial and temporal contexts into a factorization model for POI recommendation. *International Journal of Geographical Information Science*, *32*, 524–546.

Ceci, M., Lanotte, P. F., Fumarola, F., Cavallo, D. P., & Malerba, D. (2014). Completion time and next activity prediction of processes using sequential pattern mining. In *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings* (pp. 49–61).

van Dongen, B. F. (2012). Bpi challenge 2012. URL: `https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`. doi:`10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f`.

Dumas, M., Rosa, M. L., Mendling, J., & Reijers, H. A. (2013). *Fundamentals of Business Process Management*. Springer.

Evermann, J., Rehse, J., & Fettke, P. (2016a). A deep learning approach for predicting process behaviour at runtime. In *Business Process Management Workshops - BPM 2016 International Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers* (pp. 327–338).

Evermann, J., Rehse, J., & Fettke, P. (2016b). Predicting process behaviour using deep learning. *CoRR*, *abs/1612.04600*. URL: `http://arxiv.org/abs/1612.04600`.

Goedertier, S., Martens, D., Vanthienen, J., & Baesens, B. (2009). Robust process discovery with artificial negative events. *Journal of Machine Learning Research*, *10*, 1305–1340.

He, J., Li, X., Liao, L., Song, D., & Cheung, W. K. (2016). Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.* (pp. 137–143).

He, R., & McAuley, J. (2016). Fusing similarity models with markov chains for sparse sequential recommendation. In *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain* (pp. 191–200).

ter Hofstede, A. H. M., van der Aalst, W. M. P., Adams, M., & Russell, N. (Eds.) (2010). *Modern Business Process Automation - YAWL and its Support Environment*. Springer.

Kang, B., Kim, D., & Kang, S. (2012). Real-time business process monitoring method for prediction of abnormal termination using knni-based LOF prediction. *Expert Syst. Appl.*, *39*, 6061–6068.

Khusro, S., Ali, Z., & Ullah, I. (2016). Recommender systems: Issues, challenges, and research opportunities. *Information Science and Applications (ICISA)*, *376*.

Koohi, H., & Kiani, K. (2017). A new method to find neighbor users that improves the performance of collaborative filtering. *Expert Syst. Appl.*, *83*, 30–39.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008* (pp. 426–434).

Koren, Y., Bell, R. M., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, *42*, 30–37.

Lakshmanan, G. T., Shamsi, D., Doganata, Y. N., Unuvar, M., & Khalaf, R. (2015). A markov prediction model for data-driven semi-structured business processes. *Knowl. Inf. Syst.*, *42*, 97–126.

Larraín, S., Parra, D., & Soto, A. (2015). Towards improving top-n recommendation by generalization of slim. In *RecSys Posters*.

Le, M., Gabrys, B., & Nauck, D. (2012). A hybrid model for business process event prediction. In *Research and Development in Intelligent Systems XXIX, Incorporating Applications and Innovations in Intelligent Systems XX: Proceedings of AI-2012, The Thirty-second SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, England, UK, December 11-13, 2012* (pp. 179–192).

Liang, D., Altosaar, J., Charlin, L., & Blei, D. M. (2016). Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016* (pp. 59–66).

Liu, C., & Wu, X. (2016). Large-scale recommender system with compact latent factor model. *Expert Syst. Appl.*, *64*, 467–475.

Maggi, F. M., Francescomarino, C. D., Dumas, M., & Ghidini, C. (2014). Predictive monitoring of business processes. In *Advanced Information Systems Engineering - 26th International Conference, CAiSE 2014, Thessaloniki, Greece, June 16-20, 2014. Proceedings* (pp. 457–472).

Misir, M., & Sebag, M. (2013). Algorithm selection as a Collaborative Filtering Problem. *Technical Report, INRIA*, .

Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, *77*, 541–580.

Natarajan, N., Shin, D., & Dhillon, I. S. (2013). Which app will you use next?: collaborative filtering with interactional context. In *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013* (pp. 201–208).

Rendle, S. (2010). Factorization machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010* (pp. 995–1000).

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* UAI '09 (pp. 452–461). AUAI Press. URL: `http://dl.acm.org/citation.cfm?id=1795114.1795167`.

Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010* (pp. 811–820).

Ribeiro, J., Carmona, J., Misir, M., & Sebag, M. (2014). A recommender system for process discovery. In *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings* (pp. 67–83).

Ricci, F., Rokach, L., & Shapira, B. (Eds.) (2015). *Recommender Systems Handbook*. Springer.

Steeman, W. (2013a). Bpi challenge 2013, closed problems. URL: `https://doi.org/10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11`. doi:`doi:10.4121/c2c3b154-ab26-4b31-a0e8-8f2350ddac11`.

Steeman, W. (2013b). Bpi challenge 2013, incidents. URL: `https://doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee`. doi:`doi:10.4121/10.4121/500573e6-accc-4b0c-9576-aa5468b10cee`.

Tan, P., Steinbach, M., & Kumar, V. (2005). *Introduction to Data Mining*. Addison-Wesley.

Tax, N., Verenich, I., Rosa, M. L., & Dumas, M. (2016). Predictive business process monitoring with LSTM neural networks. *CoRR*, *abs/1612.02130*.

Unuvar, M., Lakshmanan, G. T., & Doganata, Y. N. (2016). Leveraging path information to generate predictions for parallel business processes. *Knowl. Inf. Syst.*, *47*, 433–461.

Weske, M. (2012). *Business Process Management - Concepts, Languages, Architectures, 2nd Edition*. Springer.