Recommendation-based Modeling Support for Data Mining Processes

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Simon Fischer
RapidMiner GmbH, Germany
sfischer@rapidminer.com

ABSTRACT

RapidMiner is a software tool that allows users to define data mining processes based on a visual model and implements a variety of so-called "operators" for data extraction, manipulation, model learning and analysis. The large number of available operators can however make it challenging for the process designer to find the appropriate operators for the problem at hand. At the same time, some operators are only meaningful when combined with certain others.

In this work, we evaluate different strategies of recommending additional operators to the user during the design of the process. The recommendation models are learned using a pool of several thousand existing data mining processes and evaluated in an offline experiment. The results indicate that good predictive accuracy can already be achieved with comparably simple co-occurrence based algorithms.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

Keywords

Recommender Systems; Evaluation; Process Modeling

1. INTRODUCTION

Practical data mining processes typically consist of several sequential steps including, for example, data-preprocessing, feature selection, model learning or performance evaluation. RapidMiner [9] is an open source software tool that supports the visual design as well as the execution of data mining and analytics processes[1]. A data mining process in RapidMiner consists of a number of process steps (called "operators") and connections between them. The operators implement some pre-defined functionality and the connections represent data flows. Figure 1 shows a fragment of a process designed in RapidMiner.

[1]`http://www.rapidminer.com`

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

http://dx.doi.org/10.1145/2645710.2645755.

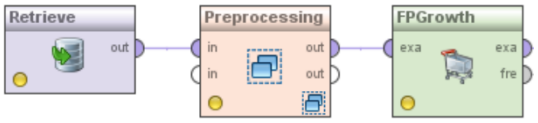

Figure 1: A process fragment.

While the graphical editor makes it easy for the designer to construct and test the process on a visual level, the design of a data mining process can still be challenging. First, process developers can be easily overwhelmed by the huge number of pre-implemented operators. Depending on the number of installed add-ons and custom extensions, the pool of available operators can comprise several hundred items. Second, some operators are only meaningful when combined with others or require that the data is transformed before it can be fed into the operator. In the example in Figure 1, for example, the frequent itemset mining operator FP-Growth is typically followed by an association rule generation operator. Likewise, the data that is fed into FP-Growth has in many application scenarios to be first transformed into a format that is suitable for itemset mining, e.g., by pivoting.

In this paper, we design and evaluate different strategies to automatically recommend additional operators to the user during the modeling process. Overall, the recommendations should make the modeling process easier for the developer by pointing him to relevant additional operators and at the same time help him avoid errors that can occur when a required step is missing in the model.

Our evaluation is based on an offline experimental design and a pool of several thousand data mining processes that were developed with RapidMiner. The algorithms tested so far are mainly based on the identification of operator co-occurrence patterns in the pool of existing processes. To compare the different strategies, we use an evaluation protocol in which we hide different operators from the processes in the test set and calculate the recall (hit rate) and the mean reciprocal rank measure for the top-n recommendation lists.

In the following sections, we will first describe the data set and the tested algorithms and then present the results of the experimental evaluation. The paper ends with a discussion of related approaches and an outlook on future works, which will in particular include user and field studies that will be conducted using a fully-functional plug-in to the RapidMiner system that has already been developed.

2. EXPERIMENTAL SETUP

2.1 Dataset

We created a collection of 6,520 data mining processes using different sources including, for example, a number of publicly posted processes on the "myexperiment" Web site[2] or other forums. The original dataset was even larger, but we applied a very strict strategy with respect to duplicates in that we removed all processes that contained the same sequence of internal operator IDs. Processes consisting of only one single process step were also removed. Table 1 shows some statistics of the dataset.

Number of processes:	6,520
Average number of operators per process:	15.5
Median number of operators per process:	10
Number of different operators:	871

Table 1: Dataset characteristics

Note that the mean and median of the number of operators per process include the operators appearing in subprocesses; when counting the different operators, we also included user-developed operators that are not contained in RapidMiner.

An important aspect in that context is that the distribution of how often the individual operators appear in the processes is highly skewed (see Figure 2).

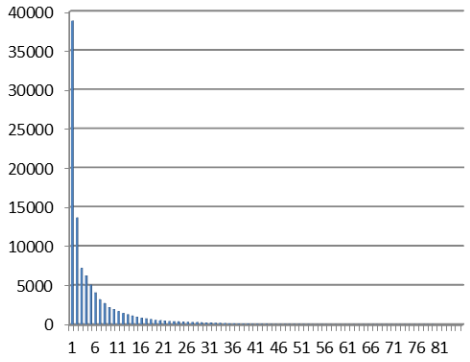

Figure 2: Distribution of operators.

To visualize the distribution, we have sorted the operators by their frequency (popularity) of appearance in the data and grouped them into bins, each containing 10 operators. The first bin therefore contains the 10 most frequently used operators. These operators appear more than 35,000 times in the processes and correspond to very general tasks like data import, data sampling, or the computation of derived attribute values. On the other hand, more than 400 operators were used less than 10 times.

2.2 Measurement method

We split the processes into training and test sets and applied a five-fold cross-validation procedure. To evaluate the accuracy of the recommendations, we removed individual operators from each process in the test set and the task of the recommender was to predict the hidden operator given the processes in the training set and the non-hidden elements of the current process. Given a top-n list of recommendations, we then determined the *recall* and the *mean reciprocal rank* (MRR) measures and averaged the values over all tested processes.

We evaluated several schemes of "hiding" elements including the selection of a random element, the selection of the first or last element, or the selection of the second to last element. Predicting the first is in our view not particularly interesting, because the first operation is typically related to data import and might be obvious. Beside the strategy of leaving out one single element, we also evaluated different "given-n" scenarios to simulate an incremental modeling process.

2.3 Algorithms

We evaluated the following recommendation strategies. Most of them rely on techniques to identify co-occurrence patterns in the data. We will discuss alternative approaches later on in Section 4.

MostFreq:	This baseline method recommends the most frequently used operators. Given the skewedness of the data, this can be a simple yet effective strategy.
CoOccur:	A recommendation strategy based on the conditional probabilities of the co-occurrence of two items.
AssocRules:	An association rule mining approach using FP-Growth [3]. The item ranking is based on the confidence of the applicable rules for a given test process.
CFP-Growth:	An association rule mining approach designed for finding itemsets and rules for "rare items" [4], which are quite common in our dataset.
SeqPatterns:	In order to be able to take the order of operators into account, a recent sequential pattern mining technique was included in the evaluation [2].
kNN:	A k-nearest-neighbors (kNN) approach in which we used cosine similarity as a distance measure and the usual weighted prediction scheme.
kNN-BPR:	Considering processes as users and operators and items, we train a Bayesian Personalized Ranking (BPR) model [10]. Predictions are made as in the kNN method with the difference that predictions are based on weighted BPR scores.

Table 2: Evaluated algorithms.

We implemented all algorithms in a Java-based evaluation framework. For the association rule mining approaches, we used the implementations from the SPMF-Framework[3]. Note that modern techniques for personalized recommendations based on matrix factorization cannot be directly applied in our setting, since the recommendations are only dependent on the current process and not the past processes modeled by a specific user.

[2] http://www.myexperiment.org

[3] http://www.philippe-fournier-viger.com/spmf/

3. RESULTS

Accuracy and coverage.

Table 3 summarizes the recall, MRR and coverage results for the setting in which the second to last element was hidden from each process in the test set[4]. When an algorithm places the hidden element of a process in the top-n list, recall is 1 and otherwise 0; the MRR in addition takes the position of the "hit" into account. The overall ranking of the algorithms was similar when a different strategy of hiding elements (like "first", "random", "last element") was applied.

Algorithm	Recall	MRR	Cov.	RecTime
kNN	**0.67**	**0.55**	1.00	20.9 ms
CoOccur	0.60	0.32	0.98	<1 ms
kNN-BPR	0.59	0.34	1.00	20.6 ms
SeqPatterns	0.58	0.34	0.99	16.7 ms
CFP-Growth	0.53	0.30	0.98	29.0 ms
AssocRules	0.50	0.30	0.98	4.75 ms
MostFreq	0.40	0.14	1.00	<1 ms

Table 3: Recall and MRR (top-10).

The results show that the kNN method with an empirically determined neighborhood size of 10 outperforms all other methods in terms of recall, and even more strongly in terms of the MRR. In our experiments, we used a typical top-n list length of 10, assuming that users will be willing to inspect up to 10 items. Experiments with shorter list sizes (e.g., 3) did not lead to different rankings of the algorithms. Quite interestingly, the simple CoOccur method works better in terms of recall than the association rules techniques. Co-occurrences of two elements[5] therefore seem very common in this domain.

The different association rule mining approaches outperform the basic frequency-based approach. The sequential pattern mining technique worked best among them, which indicates that the sequence of the operators is actually relevant. Mining rules for infrequent items is slightly more effective than the standard FP-Growth algorithm.

Recommending the most frequent items leads to recall values that were expected based on the data distribution shown in Figure 2. The coverage is high for all tested approaches.

Overall, the measurements show that in particular the neighborhood-based method is promising and was able to place the hidden item into the top-10 list in 2 out of 3 cases.

Parameter settings and running times.

A drawback of association rule mining techniques is that finding good threshold values for minimum support and confidence is tedious. Our experiments showed that lower values – e.g., minSup=0.02, minConf=0.02 for FP-Growth – led to the best results. When the values where further lowered, both the running times and memory consumption increased dramatically. For more sophisticated algorithms like CFP-Growth even more parameters have to be found.

The running times for prediction are crucial in our application since immediate feedback has to be provided during the interactive modeling process. In the right-most column of Table 3, we show the required computation times for generating one recommendation list[6]. The best-performing kNN approach needed about 20ms to produce a recommendation using a compact bitset-based representation of the training processes. The performance of the rule-based approaches is better and depends on the number of rules (ranging, e.g. up to over 400,000 for CFP-Growth). Further improvements can however be achieved through better engineering.

Given-N evaluation.

To estimate how much information the algorithms need to be able to predict suitable operators during the incremental design of a process, we used a *given-n* protocol in which we removed all but the first *n+1* elements in each process and hid the last element. Table 4 shows the results for different configurations in the form of *recall/MRR* at list length 10.

Algorithm	Given-1	Given-3	Given-5
kNN	0.39/0.14	**0.59/0.29**	**0.57/0.29**
CoOccur	0.44/0.17	0.49/0.21	0.49/0.21
kNN-BPR	**0.47/0.17**	0.52/0.21	0.55/0.21
SeqPatterns	0.40/0.15	0.45/0.20	0.47/0.20
CFP-Growth	0.33/0.12	0.40/0.18	0.39/0.18
AssocRules	0.31/0.11	0.41/0.19	0.40/0.18
MostFreq	0.38/0.14	0.33/0.31	0.31/0.12

Table 4: Different given-n configurations.

The results show that already for the extreme "cold-start" situation when the process only contains one element, several techniques are better than the frequency-based method. Due to the lack of suitable neighbors, the kNN method is not the best choice in the beginning. Very quickly, however, the method works again better than the others[7].

4. RELATED WORK

The recommendation of operators for data mining processes has to the best of our knowledge not been explored before. However, some proposals to provide recommendation support in the context of general business process modeling can be found in the literature. A classification of such approaches was proposed by Kluza et al. in [5]. According to their classification scheme, our work would fall into the category of single-element structural recommendation; the *given-N* simulation described in the previous section would correspond to a "forward-completion" technique.

In the work presented in [6] the goal is to support process designers during modeling by automatically ranking existing process fragments from a given repository of past processes. Technically, their approach is mainly based on frequency patterns and on matching the textual annotations (tags) of past processes with the currently modeled process using the typical vector space model for document encoding.

In [8], Matejka et al. explored automated command recommendation for the AutoCAD system based on command usage frequencies and nearest-neighbor methods. In contrast to our work, their algorithms produced personalized

[4]The selection was done heuristics-based since this element is not always unique because, e.g., of parallelism.

[5]E.g., X-Validation/Apply Model, Process Documents/ Filter stop words, kMeans/Normalization.

[6]For the experiments, we used a Laptop computer with an Intel i7 CPU and 8GB of RAM.

[7]Note that the absolute recall/MRR values cannot be compared on an absolute scale across the different configurations as there are processes that have less than, e.g., 5 operators.

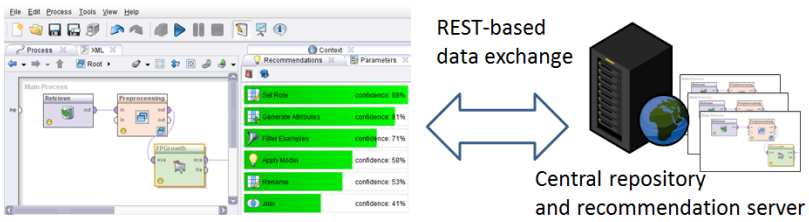

Figure 3: Architecture of recommendation system in RapidMiner.

recommendations, whereas our approach focuses on operators that are relevant in the current modeling context.

In the area of service-oriented architectures, the recommendation of suitable Web Services (e.g., for the purpose of service discovery or as a replacement of a broken services), has for example been explored in [1]. Their approach is based on the identification of patterns in past processes that are *structurally* similar to certain fragments of the currently developed process. Structural similarity is also the basis for recommendations in the more recent approach presented in [7]. Zheng et al. in [11], in contrast, proposes a collaborative filtering approach to Web Service recommendation, where the goal is to predict the expected Quality of Service (QoS) of several alternative services that can be included in a process. The input to the calculations consists of a record of past QoS levels achieved by the individual services and the goal is therefore quite different from our work.

Overall, there are two general schemes for recommendation-based modeling support which we see to be orthogonal to our co-occurrence based approaches: structural matching and "semantic"-based retrieval and ranking. Both schemes seem to be applicable in our application scenario and will be explored in our future work. Semantic information is for example available in the form of operator documentation or the in the names given by users to the process steps in their models. Structural patterns like sub-processes for cross-validation schemes or are also common in the domain.

In [6], the recommendation technique was implemented in a visual modeling editor and evaluated through laboratory studies that aimed to assess the effect of the tool on the modelers. In particular, the goal was to determine, among other aspects, if users will actually use the tool's recommendation and re-use existing fragments more often, if the tool support helps them build their models faster, if the resulting model quality is higher, and to which extent the effect depends on the experience of the modelers.

Our own work so far is based on an offline experimental design and our results indicate that our recommendation schemes can be helpful to users in different ways. In order to validate this hypothesis, we are currently preparing user and field studies in the sense of [6]. Specifically, a fully-functional plug-in component to the RapidMiner system has been developed which interacts with a central recommendation server that hosts the repository of past processes (Figure 3). So far, however, only a preliminary and informal pilot study with expert users has been made.

5. CONCLUSION

We have presented the results of an experimental analysis regarding the suitability of different algorithmic strategies for providing recommendation support for the end user during process modeling. The results indicate that co-occurrence based schemes can lead to good results in this application domain and the hybrid designs are advisable in cold-start situations. Our future works comprise (a) the evaluation of alternative approaches based on structural and semantic information, (b) an investigation of the applicability of existing process-mining techniques, as well as (c) systematic evaluations with real users.

6. REFERENCES

[1] N. Chan, W. Gaaloul, and S. Tata. Composition context matching for web service recommendation. In *Proc. IEE SCC 2011*, pages 624–631, 2011.

[2] P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. M. Nguifo. CMRules: Mining sequential rules common to several sequences. *Knowledge-Based Systems*, 25(1):63 – 76, 2012.

[3] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.

[4] Y.-H. Hu and Y.-L. Chen. Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism. *Decis Support Syst*, 42(1):1 – 24, 2006.

[5] K. Kluza, M. Baran, S. Bobek, and G. J. Nalepa. Overview of recommendation techniques in business process modeling. In *Proc. KESE 2013*, 2013.

[6] A. Koschmider, T. Hornung, and A. Oberweis. Recommendation-based editor for business process modeling. *Data and Knowledge Engineering*, 70(6):483–503, 2011.

[7] Y. Li, B. Cao, L. Xu, J. Yin, S. Deng, Y. Yin, and Z. Wu. An efficient recommendation method for improving business process modeling. *IEEE Trans Industr Informatics*, 10(1):502–513, 2014.

[8] J. Matejka, W. Li, T. Grossman, and G. W. Fitzmaurice. CommunityCommands: command recommendations for software applications. In *Proc. UIST '09*, pages 193–202, Victoria, BC, Canada, 2009.

[9] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proc. KDD '06*, pages 935–940, 2006.

[10] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proc. UAI '09*, pages 452–461, Montreal, Canada, 2009.

[11] Z. Zheng, H. Ma, M. Lyu, and I. King. WSRec: A collaborative filtering based web service recommender system. In *Proc. ICWS 2009*, pages 437–444, 2009.