Structural Feature Selection for Event Logs

Markku Hinkka[1,2(✉)], Teemu Lehto[1,2], Keijo Heljanko[1,3], and Alexander Jung[1]

[1] Department of Computer Science, School of Science,
Aalto University, Espoo, Finland
{markku.hinkka,keijo.heljanko,alex.jung}@aalto.fi
[2] QPR Software Plc, Helsinki, Finland
teemu.lehto@qpr.com
[3] HIIT Helsinki Institute for Information Technology,
Espoo, Finland

Abstract. We consider the problem of classifying business process instances based on structural features derived from event logs. The main motivation is to provide machine learning based techniques with quick response times for interactive computer assisted root cause analysis. In particular, we create structural features from process mining such as activity and transition occurrence counts, and ordering of activities to be evaluated as potential features for classification. We show that adding such structural features increases the amount of information thus potentially increasing classification accuracy. However, there is an inherent trade-off as using too many features leads to too long run-times for machine learning classification models. One way to improve the machine learning algorithms' run-time is to only select a small number of features by a feature selection algorithm. However, the run-time required by the feature selection algorithm must also be taken into account. Also, the classification accuracy should not suffer too much from the feature selection. The main contributions of this paper are as follows: First, we propose and compare six different feature selection algorithms by means of an experimental setup comparing their classification accuracy and achievable response times. Second, we discuss the potential use of feature selection results for computer assisted root cause analysis as well as the properties of different types of structural features in the context of feature selection.

Keywords: Automatic business process discovery · Process mining
Prediction · Classification · Machine learning · Clustering
Feature selection

1 Introduction

In Process Mining, unstructured *event logs* generated by systems in business processes are used to automatically build real-life process definitions and as-is models behind those event logs. There is a growing need to be able to predict

E. Teniente and M. Weidlich (Eds.): BPM 2017 Workshops, LNBIP 308, pp. 20–35, 2018.
https://doi.org/10.1007/978-3-319-74030-0_2

properties of newly added event log cases, or process instances, based on case data imported earlier into the system. In order to be able to predict properties of the new cases, as much information as possible should be collected that is related to the event log cases and relevant to the properties to be predicted. Based on this information, a model of the system creating the event logs can be created. In our approach, the model creation is performed using machine learning techniques.

One good source of additional case related features is the information stored into the *sequence of activities* visited by cases. This information includes, e.g., number of times an event log case has visited a certain *activity* and the number of times a case has *transitioned* between two specific activities. Features collected in this fashion are often highly dependent on each other. E.g., a patient whose visit to hospital takes long time (outcome) has quite often been in surgery from which he/she has moved into a ward. In this example, we already can easily find three structural features of which any can be used to predict whether the visit lasted long or not: visited surgery, transitioned from surgery to ward, visited ward. However, depending on all the other patients in the data set, it may be that there are no cases where only a subset of these three features occurs, thus making it redundant to have all three features taken into account when building a model for prediction purposes. Thus, one feature could well be enough to give as accurate predictions as having them all.

Another important aspect in Process Mining is that it is often desired to be able to show dependencies between features. Thus, selecting a feature selection algorithm that produces also this information for minimal extra cost is often tempting. For this purpose, the list of the most relevant features and the extent of their contribution should somehow be returned. One example of this kind of root cause analysis technique is influence analysis described in [13].

The primary motivation for this paper is the need to perform classification based on structural features originating from *activity sequences* in event logs as accurately as possible and using a minimum amount of computing resources and maximizing the throughput in order to be able to use the method even in some interactive scenarios. This motivation comes from the need to build a system that can do classification and root cause analysis activities accurately on user configurable phenomena based on huge event logs collected and analyzed, e.g., using *Big Data processing frameworks* and methods such as those discussed in our earlier paper [12]. The response time of this classification system should be good enough to be used as part of a web browser based interactive process mining tool where user wants to perform classifications and expects classification results to be shown within a couple of seconds.

The rest of this paper is structured as follows: Sect. 2 introduces main concepts related to this paper. Section 3 discusses the feature selection concept and gives brief introduction to the methods used in this paper. Section 4 will then present a framework used for comparing performance of the selected feature selection approaches. The results of the tests will be presented in Sect. 5 after

which we will discuss related work in Sect. 6. Finally Sect. 7 draws the conclusions from the test results.

2 Problem Setup

The concepts and terminology used throughout this paper follows the principles used commonly in process mining community. For more detailed examples and discussion about *event logs*, *activities*, *cases*, *traces* and other related terminology, see, e.g., the book by van der Aalst [22].

2.1 Structural Features

As opposed to normal case attributes added to cases in event logs, structural feature term in this paper is used for representing properties of activity sequences of cases. Thus, they can be derived directly from actual activity sequences without need to include any additional custom properties. Having a case identifier, activity identifier and order information such as a time stamp for each event occurrence, is enough. In order to simplify the tests and keep requirements for applying the results of this paper to its minimum, we decided to only concentrate on structural features as predictors in this paper. However, in real use cases, the best results are achieved by including also all the available additional case attributes such as duration, age, etc. into features from which the feature selection is performed [14].

There are several different types of structural features to select from. In this paper, we use notations similar to those used in regular expressions [20] combined with notation commonly used for activity sequences [22]. The patterns we focused in this paper are listed in Table 1 with examples of matches when the sequence of activities is illustrated as $\langle S, a, b, b, E\rangle$. As we were interested in minimizing the response time, we decided to consider only the listed patterns because having more complex patterns, such as tandem repeats and maximal repeats [2], would have made the combinations of different predictor types and features to become too big to be able to perform exhaustive tests for and leading to the problem of *curse of dimensionality* [9]. Also, the extraction of all the feature types presented in the literature would have required much more computation time than the selected relatively simple features used in this work.

Table 1. Structural feature types

Pattern/predictor type	Example sequences(s)
Activity	$\langle a\rangle, \langle b\rangle$
Transition/2-grams	$\langle S, a\rangle, \langle a, b\rangle, \langle b, b\rangle, \langle b, E\rangle$
Starter	$\langle S, a\rangle$
Finisher	$\langle b, E\rangle$
Ordering	$\langle a\rangle \rightarrow \langle b\rangle$

For every predictor type listed in the Table 1, there can be several possible implementations. In this paper, we consider structural features of *activity* and *2-gram* predictor types to be such that their values correspond to the number of occurrences of that pattern within each activity sequence. *Starter and finisher* predictor types however are boolean values indicating whether that pattern is valid for an activity sequence. Order feature type is considered to be a boolean value such that it is true only if the first occurrences of both ends of the order relation are in the specified order.

The difference between 2-gram and order pattern is that order allows any number of activities to be between the activities of the ordering relation, whereas in 2-gram, the activities of the relation must be successive in the whole sequence of activities. The importance of predictor types also depends very heavily on the type of the data set and the scenario being predicted.

One more factor to take into account when selecting the actual features is how to handle situations where a feature has more than two different values. E.g., a patient may have visited surgery multiple times while visiting a hospital. In some cases, depending on what we are trying to predict and what kind of prediction models are to be built, it could be better to split these kinds of features into several boolean features. Thus, e.g., we could have a feature for a patient having visited surgery 4 times. However, one has to be careful when to split features like this in order to avoid an explosion in the number of features created for each original feature.

One final step before passing features to the actual model training is to identify and remove any duplicate features that have behaved identically through the whole training set. Some training methods do this automatically, but some do not.

2.2 Classification

Since all the tests performed in this paper are performed on data sets consisting of already completed cases, we are performing *classification* using machine learning prediction algorithms. Classification in machine learning usually consists of two phases: training a model and performing the actual classifications using the trained model. In this paper, we concentrate on building the classification model using *supervised learning methods*, where algorithms are trained using *predictors*, together with their *outcomes*. A core part of the model building is the selection of *features* to be used as predictors. Often the more you have independent features that may have effect in the outcome, the better. As shown in Sect. 2.1, a lot of features can be created directly from the activity sequences of the cases themselves.

Another important factor that has direct effect to the prediction performance of the model is the algorithm that is used for building the model and making the predictions. In this paper, we focus on the feature selection part. However, we need to also validate the performance of the feature selection using a set of algorithms. In order to minimize the skew in the results caused by the validation

algorithm itself, we decided to compare the efficiency of the selected features using two different approaches. First, for a given set of selected features, we determined the prediction accuracy obtained by a particular supervised learning method, i.e., the gradient boosting machine (GBM). GBM was selected due to its good reputation [9,18] and performance in both accuracy and response times in our own tests. The second method was to approximate the mutual information score [16] between each of the selected set of features and two different data sets. The first data set consisted of all the available predictors without any feature selection. The second data set consisted only of the outcomes to be predicted.

As we are concentrating on features originating from process mining, at the granularity level of a case, the prediction inputs that are usually used in the field are actually custom case properties such as the customer name or an identifier of the owner of the case. The outcomes that we want to predict are usually values of some custom case properties or some calculated case content dependent values such as durations, some kind of cost of the case or some other metrics measuring the quality of the case. In this paper, we concentrate only in features inherent to the activity sequences inside cases and measure how well certain outcomes can be predicted only based on those features.

The used data set is split into two parts: training and test. Training data set is used for two purposes. First, features are selected from the whole training data set. After this, a model is trained using all the cases in the training data, but only using the selected features as predictors. Finally, once the model has been built, the model is tested against the test data and its performance is estimated using accuracy and mutual information metrics.

3 Feature Selection Methods

The aim of feature selection is to reduce the dimensionality of the structural features constructed from the raw data. Reducing the dimensionality not only reduces the computational complexity of the subsequent prediction methods, it may also lead to an improved prediction accuracy. Indeed, learning algorithms based on a smaller set of features are less prone to overfitting, i.e., the effect of erratic statistical variations of a particular observed dataset is reduced. Finally, feature selection also enhances the interpretability (visualization) of the features and understanding classifications based on them (e.g., if only two numerical features are selected, we can illustrate them by means of a scatterplot).

Initially we also considered testing a couple of feature extraction algorithms. Feature extraction differs from feature selection in that they create new features that will be used instead of the original features. The newly created features try to maximize the variance and expressive power of the features by combining several original features into one new feature. This has a drawback that it hides the original features and makes it harder to understand the properties of the created model. E.g., in root cause analysis, it is often desirable to understand how much the outcome depends of certain features and also to understand which features have an effect to the outcome. Due to this shortcoming, we decided to not include any feature extraction algorithms into this paper.

No additional parallelization techniques were used, thus if the algorithm did not support parallelism out of the box, it was not run in parallel. The following subsections briefly describe the basics of each of the feature selection methods tested in this paper including information on the used R programming language packages and their configurations. We also briefly tested an algorithm based on *Support Vector Machine* (*SVM*) [1,25] using radial kernel, but decided to leave it out of the paper due to very poor results and extremely long response times, which were order of magnitude slower than with any of the other tested algorithms. The following subsections will briefly describe the details of all the remaining tested algorithms.

Random Selection. The most trivial of all the tested algorithms was a randomized selection where the desired number of features were just randomly selected from all the available features. This method was used as a baseline in order to gain a better understanding on the quality of other used selection algorithms when compared with an algorithm that does not in any way take any properties of the selected features themselves into account. This serves as a baseline selection algorithm. There should not be any algorithm that performs consistently worse than this. In order to alleviate the effect of inherently noisy random selections, median of three separate test runs was used in the experiments. Thus, only the test which yielded the median prediction accuracy was used as the actual result. In the graphs and analysis below, this algorithm is labeled as *Random*.

Feature Clustering. This method is influenced by the idea provided by Covoes et al. [5]. In the algorithm developed for this paper, the training data is first clustered so that every structural feature in the training set constitutes one clustering data point. Each activity sequence in the training data represents one dimension for clustering data points with values equaling the number of times that structural feature occurs within that activity sequence. K-means algorithm is then used to generate K clusters using kmeans R-function which is based on algorithm by Hartigan and Wonget [10]. For each K clusters, the feature having the minimum distance to the mean of that cluster will be selected as the representative for all the features in that cluster. It should be noted also that as a side product of applying this method for feature selection, every selected feature will actually represent all the features within the same cluster. Thus, for every original feature, you have one cluster it belongs into and exactly one feature that is representing that feature in that cluster. This could be useful, e.g., in some root cause analysis scenarios.

Two different versions of this algorithm are tested in this paper. One that first removed all the features having exactly the same occurrence pattern within all the cases thus removing duplicate vectors before the actual clustering step. The other variation of this algorithm does not perform this preprocessing step. The results of different variations being nearly the same except for the processing time, which was clearly faster with the algorithm that first dropped out all the features having exactly the same values for all the cases in the training set. Thus,

we decided to limit our tests only to this algorithm variant. In the graphs shown below, this algorithm is labeled as *Cluster*.

Minimum Redundancy Maximum Relevancy. This is a mutual information based approach [6], which uses mutual information as a proxy for computing relevance and redundancy among features. The implementation used in this paper was provided by mRMRe -R package which claims to provide a highly efficient implementation of the mRMR feature selection via parallelization and lazy evaluation of mutual information matrix. We used ensemble method both with solution_count set to 1, which provides results resembling classic mRMR, and also with 5, which does 5 separate runs and combines the results in the end. This time the results were also otherwise quite the same, except the 5-run version provided clearly better mutual information scores. Thus, in the graphs and analysis below, we use only 5 run version labeled as *mRMREns5*.

Least Absolute Shrinkage and Selection Operator (LASSO). This is a regression analysis method that can be used for feature selection [21]. It is related to least squares regression where the solution minimizes the sum of the squares of the errors made. The unique property for this regression technique is the usage of additional regularization that enables discarding irrelevant features and forces usage of simpler models that do not include them. Since the LASSO implementation in *glmnet* R-package in itself did not provide means of sorting features by their importance and since it was not possible to directly adjust the desired number of target features, the actual used algorithm first performed 10 iterations of LASSO algorithm each yielding slightly different results. After this, all the results were collected into a single list with each feature weighted by the number of occurrences of that feature within all the LASSO results. Finally, this list was sorted from the largest height to smallest and the desired number of features were picked from the beginning of this sorted list. Two different variations of this algorithm were tested: one using lambda.1se as the prediction penalty parameter and the other using lambda.min. Due to the results being almost identical in both the cases, we selected the one using lambda.1se prediction penalty parameter. In the results below, this algorithm is labeled as *LASSO1se*.

Markov Blanket. Markov blanket of a variable X is a minimal variable subset conditioned on which all other variables are probabilistically independent of X. [26] For Bayesian networks, Markov blanket of X consists of the union of the following three types of neighbors: the direct parents of X, the direct successors of X, and all direct parents of X's direct successors. Bayesian networks can be inferred from the training data after which Markov blanket for the created network is calculated by selecting the outcome as X. The result is the set of features to select. bnlearn -R package was used to perform Markov Blanket based feature selection. *Hill-Climbing* algorithm is first used to construct a Bayesian network structure out of the training data. After this, Markov Blanket is extracted out

of the network for the outcome feature. Finally, out of these results, the desired number of features are selected from the beginning of the returned list, or if the result does not have all the required features, only the returned features are selected. In the results below, this algorithm is labeled as *Blanket*.

Variable Importance. In variable importance based feature selection, some Machine learning algorithm capable of building variable importance information, such as random forest [15], is first performed. After this, the results of the algorithm are used to pick N variables having the greatest effect on the outcome. These N variables are then used as the selected features. Since the performance of variable importance algorithm itself was found to be very poor when using predictor sets having hundreds of features we decided to use a hybrid approach where we first use clustering to remove about 75% of all the features, after which variable importance is calculated for each feature using random forest algorithm and from there, the desired number of the most important target features is picked. In this paper, randomForest -R package is first used to generate a model after which varImp -R function in Caret-package is used to extract the most important features based on the information gathered by the random forest algorithm. This algorithm is labeled as *ClusterImp* in the graphs and analysis below.

Recursive Feature Elimination. Recursive feature elimination [8] starts with estimating the variable importances of all the features in the training data as in the Variable Importance -technique. After this, a smaller subset of the most important features is selected and variable importances are estimated again. This is repeated until the desired feature subset size is reached after which the resulting features can be picked. In this paper, three different variations of this method were tested: a test with only one iteration, another with two iterations and the third one with four iterations. Caret R-package's *rfe* algorithm was used for these tests, with the default method based on random forests. After the initial tests, it was found out that while the accuracy and mutual information of all the cases were very close to each other, the average processing time of the 2-step algorithm was clearly better than the others. Thus, in the graphs and analysis below, we only concentrate on this algorithm labeled as *Rec2S*.

4 Test Setup

Testing was performed on a single system using Microsoft R Open version 3.3, Windows 10 operating system. The used hardware consisted of 3.5 GHz Intel Core i5-6600K CPU with 8 GB of memory. Tests were performed using two publicly available data sets. The first one from Rabobank Group ICT [24] that was also used in BPI Challenge 2014. This data set contained total of 46617 cases of a real-life event log from Rabobank Group ICT company. The average length of a case is nearly 10 events split into 39 different event types. The second

used data set is a real-life data set from Dutch academic hospital [23] consisting of 1143 cases. The average length of a case in this data set is slightly over 130 events split into 624 event types.

All the test runs were performed using an R function that ran all the desired test runs in sequence. At the beginning of every test run, random seed was initialized. Thus, the random case samples and other random values generated within the used algorithms behaved the same way in every run, provided that the algorithm used random -methods that support setting the seed using *set.seed* -R function.

In the tests, the training data was first extracted so that 25% of the provided data rows were randomly selected. This training data was first used by the feature selection algorithm to be tested, after which it was used to build the classification *GBM* model for predicting given phenomenon. This classification model was then used for measuring the performance of the feature selection. Mutual information metrics were approximated also at this final phase.

The first run of tests was performed using test data having 4000 cases extracted from the full BPI Challenge 2014 data set. For this first run, all the algorithms were tested so that the number of selected features were 5, 10 and 30. For each of these combinations, 13 different sets of feature patterns were selected. The selected structural feature patterns were different combinations of the following patterns described in Table 1: activity, starter and finisher, 2-grams and ordering. Activity and 2-grams features included occurrence counts, while the others were just boolean values indicating whether the feature occurs at least once in a case. The combinations were created in a way that all the possible combinations of the patterns were tested where activity pattern was present, thus generating 8 different combinations. In addition to these, we also tested 2-grams and ordering separately, as well as having both 2-grams and ordering. Most emphasis was given for activity pattern since it is usually the easiest to extract from event logs and also doesn't yield that many features. Also, we did not want to include any other patterns due to the number of potential new features that would have been needed in order to cover all the possible cases. E.g., adding 3-grams would potentially have generated N^3 additional features where N equals the number of different activities in the training data, which in this case is 39 yielding the maximum of 60000 new features. When using a case sample of 4000 cases, the numbers of features added were as follows: 39 features for activity pattern, 20 features for starter and finisher, 772 features having 2-grams and 1033 features were generated for ordering-pattern. Thus, the total maximum number of features extracted from event logs when all the patterns are included is 1864.

All the tests performed on the first data set were run to predict two different outcomes, which are later in this paper referred to as scenarios. The first scenario was whether the case duration is longer than 7 days. In this case, nearly 37% of all the cases in the small test set had this outcome. This is an example of a prediction that can be trained directly from the event information without any need for additional case or event attributes. The second scenario that was

tested is based on additional case-level information provided with the event data: Does the case represent a "request for information" or something else such as an "incident"? In this case, nearly 20% of all the test cases in the small sample had this outcome. For all the tests performed on the same sample size, the actual used cases and their predictors were always the same.

Finally, we compared the results achieved from these earlier tests to tests run on the second Dutch academic hospital data set. In this case, we used all the 1143 cases in the data set in the tests. The number of features added by activity patterns were as follows: 624 features for activity pattern, 36 features for starter and finisher, 4272 features for 2-grams and 79571 features for ordering-pattern.

5 Test Results

We began the actual analysis of our first round of tests by estimating the average classification accuracy of all the tested algorithms for all the tested feature counts and all the tested structural feature patterns using both the test scenarios in 4000 case sample. The results of this analysis are shown in Fig. 1. The first column on the chart labeled *None* shows the accuracy achieved by not performing any feature selection. Based on these results, we can see that the feature selection algorithms ordered in descending order of accuracy are: Recursive, Cluster, Blanket, mRMR, Cluster Importance and LASSO. The rankings of three first algorithms are not changed even with the tested bigger data set size.

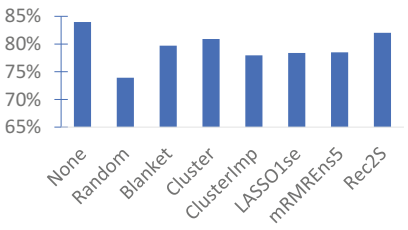

Fig. 1. Average accuracy of all the tested algorithms.

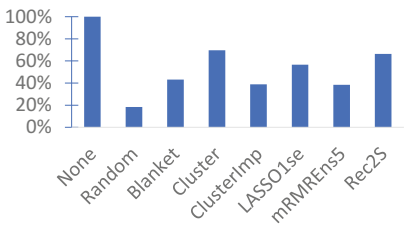

Fig. 2. Average mutual information of all the tested algorithms.

With the same test data, we also measured the average percentage of the mutual information of a data set filtered using feature selection algorithm when compared to the mutual information calculated when the feature selection is not performed at all. This is shown in Fig. 2. The ranking of algorithms in the mutual information case are: Cluster, Recursive, LASSO, Blanket, Cluster Importance and mRMR. Also, in this case, the rankings of three first algorithms are not changed even with the tested bigger data set sizes. It should be noted at this point that identical rankings were obtained in the case the mutual information was calculated between the result of the feature selection algorithm and only the correct outcomes.

After this, we analyzed the response times for all the tested feature selection algorithms with the same test data. This time however, we did not include starter and finisher predictor sets since they would have made the readability of the figure much worse and also would not have provided much additional information due to the small effect they have into the results in the tested scenarios.

As seen in the Fig. 3, the time required to perform feature selection for the tested algorithms and predictor types varied very much. In the worst cases, the difference between the slowest and the fastest algorithm was three orders of magnitudes, with Cluster and mRMR usually performing much faster than all the others.

The largest performance variations within one algorithm were measured using Blanket algorithm which performed in the fastest predictor set case, almost as fast as the two fastest algorithms, but in the slowest case, it performed almost over four orders of magnitudes slower.

Both Clustering and mRMR performed so well in this analysis that they could be incorporated without changes into some interactive process mining systems preferring under ten second response times when the size of the event log used for training is close to 1000 cases. Since we are especially interested in response times, we took Clustering and mRMR algorithms for closer inspection.

First, we analyzed the classification accuracy of both the algorithms separately for both the tested scenarios. Figure 4 shows this information for Cluster algorithm and Fig. 5 for mRMR algorithm.

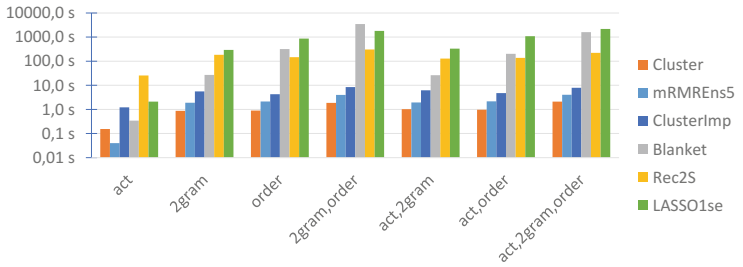

Fig. 3. Average feature selection response time of all the tested algorithms.

From these results, it can be seen that there is a lot of variation between the two tested scenarios. In both the cases, in almost all the predictor sets, predicting case duration produced clearly worse results than in the categorization scenario. It seems that mRMR algorithm was not able to get any additional accuracy into its predictions by including any additional predictor types on top of activity predictors, whereas Cluster algorithm managed in the categorization case to get better accuracy even, e.g., by selecting only *2-gram* or *order* type predictors. Also, in duration scenario, it managed to get better accuracy when adding *order* type predictors in addition to *activities*. Also, in almost all the test cases, having *order* type predictors is more valuable than having *2-gram*

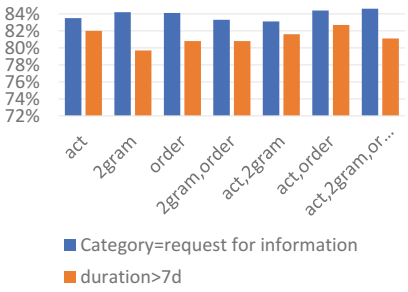

Fig. 4. Average accuracy for cluster algorithm separately for each scenario and predictor types.

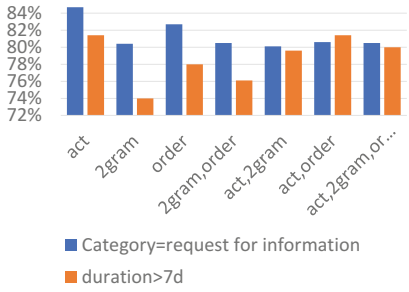

Fig. 5. Average accuracy for mRMR algorithm separately for each scenario and predictor types.

type predictors. Having them both is in most cases worse than having *order* only. The number of features has slightly greater effect into the performance of mRMR than for Cluster algorithm. However, for the data set size, the situation is reversed and Cluster is affected clearly more than mRMR. The recommended data size used for feature selection is in our experiments under 10000 cases in order to maintain good response times.

It should be also noted that the time required for building a prediction model with a feature selection algorithm selecting 10 features was only about 1%–3% of the total time required when building the model with all the 1864 tested structural features without any feature selection. When using 3000 cases to build a model, the total measured time difference in the test system was about 250 s. During this time, it would have been possible to run the clustering feature selection several times. Thus, it is clear that having a feature selection performed before model building, at least when GBM is used, is essential when trying to improve the time required for model building.

Finally, we also performed similar tests with the Dutch academic hospital dataset. Tests performed in this dataset indicated that it is absolutely critical to perform some kind of feature selection before training the model since building the model without any selection failed when attempting to use all the 79571 ordering features valid in this event log. From the six tested feature selection finalists, only clustering and clustering with variable importance managed to provide results for all the tested predictor combinations. Out of these two algorithms, clustering with variable importance provided this time slightly more accurate results while both of them still managing to provide relatively good response time of 25–40 s. This result is caused by the fact that clustering itself does not take the outcome into account at all, whereas variable importance based feature selection does. In this case, clustering first filters out most of the features but still tries to maintain as much of the original information content in the predictors as possible. Applying variable importance after this with the specified outcome will filter out all the clusters of original features that do not have anything to do with the outcome.

Thus, in order to fulfill all the requirements, we have for the algorithm, based on the performed tests, the best option from the selected set of algorithms is Cluster based feature selection with only *activity* type predictors with values being the occurrence counts of the *activity* within a *case*. In our experiments, it gave the best overall trade-off in performance considering mutual information, classification accuracy and response time. If a slightly better classification accuracy is required, we recommend adding *order* predictor types to the set of features to pick the actual features from. For event logs having large number of event types and relatively small amount of cases, mixing clustering with variable importance provided better accuracy than performing only clustering.

6 Related Work

In recent years, several papers have been written on the subject of applying data mining and machine learning techniques into predicting outcomes of the business processes. In [3], the authors present a framework that is capable of automatically detecting "signatures" that can be used to discriminate between desired and undesired behavior within traces both seen or unseen. These signatures are essentially combinations of structural features similar to those described in Sect. 2.1. This paper does not in itself specify any automatic feature selection method. Instead, the user is required to specify manually the desired activity sequence patterns, referred to as sequence feature types. After this all the matching features will be used for signature detection. Thus, our research complements the research made in this paper by experimenting with different automatic feature selection methods that could be applied before this signature detection phase in order to reduce the computational cost of signature detection at the cost of some prediction accuracy.

In [17], the authors evaluate the accuracy achieved with three different prediction algorithms using several combinations of structural feature patterns for three different datasets. As result, they find out that just having Activity frequencies often yield, if not the best, then at least almost as good results as the best tested structural feature pattern combination. This finding is visible also in our tests as shown in Figs. 4 and 5.

In [7], the authors present a framework for predicting outcomes of user specified predicates for running cases using clustering based on activity sequence prefixes and classification using attributes associated to events. In [14], the authors have assessed the benefits of including case and event attributes when performing predictions based on sequences of activities. In [19], the authors present a predictive process monitoring framework that is also able to mine unstructured textual information embedded into attributes related to events. In [4], the authors propose a recommendation system that automatically determines the risk that a fault will occur if the input the user is giving to the system will be used to carry on a process instance.

Until now there has not been systematic testing of applying automatic feature selection algorithms after selecting structural feature patterns to use and

before building models used for classification. The aim of this feature selection is to minimize the computational cost of the building of classification models. Creating such an approach is crucial for obtaining predictions with interactive response time requirements. This is the primary contribution of this paper.

7 Conclusions

In this paper, we have designed a system for assessing the performance and response times of selected feature selection algorithms specifically tuned into the context of selecting structural features extracted from properties of sequences of activities derived from event logs. Using this system, we tested six feature selection techniques using a total of twelve different algorithms.

Each algorithm was tested using a publicly available real-life Rabobank Group ICT dataset and tuned for two different classification use cases: Predicting whether the duration of a case is longer than seven days, and classifying whether a case is of type request for information. Most of the tests were also run using two different sample sizes out of the full dataset. For sanity checking and benchmarking purposes, we also added test runs without any feature selection and also with randomized feature selection.

We also proposed a rough categorization method for some of the types of structural features that can be extracted from event logs. In this paper, we selected four of these types for closer inspection.

As summary for all the tests and their results, it can be clearly seen that structural features can provide additional means for improving the precision to classifications made for cases in event logs. When the number of selected features is small, the most efficient source of features is activities. Increasing the number of features improves the classification accuracy, but also while doing so, best results are achieved by adding features from other structural feature types such as event type orderings into the set of structural features from which the feature selection is made. However, there is a drawback that having a bigger pool of features to select from makes creating classification models as well as the feature selection slower. As our goal was also to find an algorithm that could perform feature selection and classification with interactive response times using the sample sizes used in this paper, we found out that only one feature selection algorithm of the tested algorithms provided both the speed and accuracy required for the task.

According to the tests, the most consistently well performing algorithm was *Cluster* algorithm we developed for this paper which first used *k-means* algorithm for clustering features into the desired number of clusters by having cases as clustering dimensions, after which the features closest to the center of each cluster were selected as the selected features. This algorithm was not in all the tests as fast as *mRMR*, but it provided feature selections yielding more accurate classifications and it worked without problems with all the tested data sets. Also, it was not as accurate as some other algorithms, such as *Recursive Feature Selection*, in both the scenarios, but it consistently achieved very accurate results in

all the tested scenarios within the response time requirements set in this paper. For event logs having very large number of event types, mixing feature clustering with variable importance provided more accurate results than clustering only. For computer assisted root cause analysis, in addition to providing the list of the most important features, *Cluster* algorithm provides a mapping from each of the original structural features to one selected feature that most closely resembles the original feature in the set of selected features.

All the gathered raw information from the performed test runs can be found in the support materials [11].

Acknowledgements. We want to thank QPR Software Plc for funding our research. Financial support of Academy of Finland projects 139402 and 277522 is acknowledged.

References

1. Bennett, K.P., Campbell, C.: Support vector machines: hype or hallelujah? SIGKDD Explor. **2**(2), 1–13 (2000)
2. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Abstractions in process mining: a taxonomy of patterns. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 159–175. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_12
3. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Discovering signature patterns from event logs. In: IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16–19 April 2013, pp. 111–118. IEEE (2013)
4. Conforti, R., de Leoni, M., Rosa, M.L., van der Aalst, W.M.P., ter Hofstede, A.H.M.: A recommendation system for predicting risks across multiple business process instances. Decis. Support Syst. **69**, 1–19 (2015)
5. Covões, T.F., Hruschka, E.R., de Castro, L.N., Santos, Á.M.: A cluster-based feature selection approach. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 169–176. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02319-4_20
6. Ding, C.H.Q., Peng, H.: Minimum redundancy feature selection from microarray gene expression data. J. Bioinform. Comput. Biol. **3**(2), 185–206 (2005)
7. Francescomarino, C.D., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. CoRR, abs/1506.01428 (2015)
8. Granitto, P.M., Furlanello, C., Biasioli, F., Gasperi, F.: Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. Chemom. Intell. Lab. Syst. **83**(2), 83–90 (2006)
9. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2000)
10. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: a k-means clustering algorithm. J. Royal Stat. Soc. Ser. C (Applied Statistics) **28**(1), 100–108 (1979)
11. Hinkka, M.: Support materials for articles (2017). https://github.com/mhinkka/articles. Accessed 13 Mar 2017
12. Hinkka, M., Lehto, T., Heljanko, K.: Assessing big data SQL frameworks for analyzing event logs. In: 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2016, Heraklion, Crete, Greece, 17–19 February 2016, pp. 101–108. IEEE Computer Society (2016)

13. Lehto, T., Hinkka, M., Hollmén, J.: Focusing business improvements using process mining based influence analysis. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNBIP, vol. 260, pp. 177–192. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45468-9_11
14. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23063-4_21
15. Liaw, A., Wiener, M.: Classification and regression by randomforest. R news **2**(3), 18–22 (2002)
16. Meyer, P.E.: Information-theoretic variable selection and network inference from microarray data. Ph.D. thesis. Université Libre de Bruxelles (2008)
17. Nguyen, H., Dumas, M., La Rosa, M., Maggi, F.M., Suriadi, S.: Mining business process deviance: a quest for accuracy. In: Meersman, R., Panetto, H., Dillon, T., Missikoff, M., Liu, L., Pastor, O., Cuzzocrea, A., Sellis, T. (eds.) OTM 2014. LNCS, vol. 8841, pp. 436–445. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45563-0_25
18. Ogutu, J.O., Piepho, H.-P., Schulz-Streeck, T.: A comparison of random forests, boosting and support vector machines for genomic selection. In: BMC Proceedings, vol. 5, no. 3, p. S11 (2011)
19. Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C.: Predictive business process monitoring with structured and unstructured data. In: La Rosa, M., Loos, P., Pastor, O. (eds.) BPM 2016. LNCS, vol. 9850, pp. 401–417. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45348-4_23
20. Thompson, K.: Programming techniques: regular expression search algorithm. Commun. ACM **11**(6), 419–422 (1968)
21. Tibshirani, R.: Regression shrinkage and selection via the Lasso. J. Royal Stat. Soc. Ser. B (Methodological) **58**(1), 267–288 (1996)
22. van der Aalst, W.M.P.: Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer, Berlin (2011)
23. Van Dongen, B.: Real-Life Event Logs - Hospital Log (2011). https://doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
24. Van Dongen, B.: BPI Challenge 2014. Rabobank Nederland (2014). http://dx.doi.org/10.4121/uuid:c3e5d162-0cfd-4bb0-bd82-af5268819c35
25. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T.A., Vapnik, V.: Feature selection for SVMs. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA, pp. 668–674. MIT Press, Cambridge (2000)
26. Zeng, Y., Luo, J., Lin, S.: Classification using Markov blanket for feature selection. In: The 2009 IEEE International Conference on Granular Computing, GrC 2009, Lushan Mountain, Nanchang, China, 17–19 August 2009, pp. 743–747. IEEE Computer Society (2009)