Sequential Approaches for Predicting Business Process Outcome and Process Failure Warning

Mai Le[1], Detlef Nauck[2], and Bogdan Gabrys[1]

[1] Bournemouth University, Bournemouth, UK
mai.phuong@bt.com,bgabrys@bournemouth.ac.uk
[2] BT research, Ipswich, UK
detlef.nauck@bt.com

Abstract. Large service companies like telecommunication businesses run complex customer service processes in order to provide communication services to their customers. The flawless execution of these processes is essential since customer service is an important differentiator for these companies. They must also be able to predict if processes will complete successfully or run into exceptions in order to intervene at the right time, pre-empt problems and maintain customer service. Business process data is sequential in nature and can be very diverse. Thus, there is a need for an efficient sequential forecasting methodology that can cope with the diversity of the business data. In response to these requirements, in this paper we propose an approach which is a combination of KNN (K nearest neighbour) and sequence alignment for predicting process outcome. The proposed approach exploits temporal categorical features of the extracted data to predict the process outcomes using sequence alignment technique, and also addresses the diversity aspect of the data by considering subsets of similar process sequences, based on KNNs. We have shown, via a set of experiments, that our model offers better results when compared with original KNNs and random guess-based methods. We also introduce a rule based technique based on GOSPADE, which detects the repetitions of individual tasks and investigates the relationship between them and the process failure. The results are demonstrated in a comprehensive performance study on real business process data sets.

1 Introduction

Process mining is a relatively young research discipline but has been attracting growing attention recently due to its capability for predicting process outcomes. In every organisation, there are a large number of business processes to be dealt with on a daily basis. A business process is a sequence of tasks/activities that need to be completed to produce a product or to provide a service. Business processes need to be carefully monitored and controlled (business process management) to be effective and efficient.

The idea behind process mining is to discover, monitor and improve processes, aiming for excellent process execution. Process mining models contribute to many phases in business process management including the diagnosis phase,

operational support, etc. [1], [2], [3]: during the execution phase the process is monitored and can be slightly adjusted without redesigning the process; in the diagnosis phase, the enacted process is analysed and the outcome of this analysis can be used to redesign the process; Predictions and recommendations based on models learnt from historical data can be used for online maintenance because being aware of what might happen helps managers to set up suitable strategies to intervene on time [4]. For example, it is of interest to investigate certain loops that might lead to process failure; suggesting an optimal way to complete the process starting from the current step etc.

Such models can be drawn from a rich source of mathematical models in data mining [5]. However, due to the complexity and diversity of process data as well as the sequential nature of process data, it seems to be poorly aligned with any single classical data mining technique. Similar to data from business process executions, customer behaviour data also consists of asynchronous sequences and is considered as a kind of stochastic process. Customer behaviour data is described by sequences of interactions between the customers and the company while business processes are described by sequences of process events, i.e. steps or tasks. In both cases the events and the entities (customers or jobs) that move through the process instances are described by additional attributes. Therefore, in this study we use both types of data, business process data and customer behaviour data in order to test out if our proposed model is generic in terms of dealing with sequential data.

This paper introduces a new K nearest sequence method which is able to deal with sequential data. Our objective is to group similar sequences together expecting that sequences which behave similarly in earlier steps go to the same final step. In order to create a KNN model, which addresses the diversity and the temporal character of the data, an original KNN is combined with sequence alignment technique.

In our area of application we consider a number of sequences in which we want to find K sequences which are most similar to a given sequence $S(s_1^{(j)}, \ldots, s_{n_j}^{(j)})$, where j is the identity of the sequence. The similarity is determined using a distance function and, the prediction should be the majority class among classes of the K nearest sequences. This method is used in [6] to predict churn using data from a telecommunication company. The authors combined KNN and the theory of survival. In their work, Euclidean distance is used to calculate the distance between the given sequence and all sequences in the data sample. As our data consists of event sequences, in this study we use the edit distance in the process of comparing two sequences.

It is also of interest to look into loops, which occurred in the data, to identify some of them might lead to process failure. Hence, apart from the proposed predictive models, in this paper we employ a potential technique in order to provide warnings about process failure. It is based on a special algorithm in association rules called GOSPADE [7] and it enables us to detect loops and the links between them and the process failure.

The problem can be formulated as follows: A business process instance (S_j) is a composition of discrete events (or tasks) in a time-ordered sequence, $S_j = \left\{s_1^{(j)}, s_2^{(j)}, \ldots, s_{n_j}^{(j)}\right\}$, s_k takes values from a finite set of event types $E = \{e_1, \ldots, e_L\}$. Apart from its starting time $t_i^{(j)}$ and ending time $T_i^{(j)}$, each of these events has its own attributes. For simplicity, we assume that a process does not contain any overlapping events, that means there are no parallel structures.

The goal is to predict the outcome (success/failure) of a given process instance $S_{N+1} = \left\{s_1^{(N+1)}, s_2^{(N+1)}, \ldots, s_{i-1}^{(N+1)}\right\}$ based on the data from closed process instances $S = \{S_1, S_2, \ldots, S_N\}$ and to determine if the consecutive repetition of a task S_j potentially leads to a failure for the considered process instance.

The remainder of this paper is organised as follows: Section 2 presents the proposed models for predicting the process outcome. Section 3 discusses the GOSPADE algorithm based loop failure detection (LFD) technique. It is followed by Section 4 with a brief review of the data used in our analysis before we present the results of our experiments. In Section 5, we conclude the paper and discuss future directions.

2 Sequential KNNs

To determine sequence similarity both distance measure and similarity measure can be used. For numeric variables well-known distance measures exist and can be easily applied. However, in sequence analysis sometimes we have to work with sequences which are constructed from symbols, e.g. categories (churn prediction), phonemes (speech recognition) or characters (hand-writing recognition), etc. We need specialised functions which have the ability of measuring the similarity of symbolic sequences.

2.1 Sequence Alignment

Sequence alignment is very common in bio-informatics and has a relatively long history in this domain. The target entities of sequence alignment in bio-informatics are amino acid sequences of proteins, DNA sequences, etc. Sequence alignment is used for a number of purposes [8]. Algorithms used in sequence alignment are mainly divided into global alignment and local alignment. Global alignment provides a global optimisation solution, which span the entire length of all query sequences. In contrast, local alignment aims to find the most similar segments from two query sequences. In this work, both types of alignment are investigated to verify which one is effective in determining the similarity between the two sequences, the overall comparison between two given sequences or the most similar (consecutive) segments. The similarity between process sequences is used to predict the process outcomes.

– *Global algorithm*

In this kind of algorithms, sequences are aligned from the first event to the last one. One such algorithm was introduced by Needleman and Wunchs [8]. There are three characteristic matrices associated this algorithm: substitution matrix, score matrix and traceback matrix. The role of the substitution matrix is to generate the degree of matching any two events from the set of event types, or in other words matching subsequences of length 1. This degree which is irrespective of the position of the events then contributes to the matching score in the score matrix that consider the complete sequences, i.e. all events in the order they occur. Given two sequences, we then have to consider the order of the events and compute the score of matching the i^{th} event in one sequence with the j^{th} event in the other sequence, $i = \{1, \ldots, len_1\}$, $j = \{1, \ldots, len_2\}$ and len_1, len_2 are the lengths of the two given sequences. These scores define the score matrix. Finally, the trace back matrix encodes the optimal way of matching both sequences from a number of possible matches. We now introduce these three matrices.

1. Substitution matrix: in biology a substitution matrix describes the rate at which one amino acid in a sequence transforms to another amino acid over time. The entries of this matrix present the probabilities of transforming one amino acid to another. There are different ways of generating the substitution matrix. The simplest way is to not take into account the amino acid mutation factor, instead just give a score of 1 to the same amino acids and use a score of 0 to a pair of different amino acids.

$$s(i,j) = \begin{cases} 0 \; if & eventi \neq eventj \\ 1 & otherwise \end{cases}$$

   In this case, the substitution matrix is an identity matrix, the elements of the main diagonal are 1 and all the others are 0.
2. Score matrix: This matrix's elements are similarity degrees of events from the two given sequences.

$$h_{i0} = -\delta \times i, \tag{1}$$

$$h_{0j} = -\delta \times j, \tag{2}$$

$$h_{ij} = \max\left\{h_{i-1,j} - \delta, h_{i-1,j-1} + s(x_i, y_j), h_{i,j-1} - \delta\right\}, \tag{3}$$

   where $i = \{1, \ldots, len_1\}$, $j = \{1, \ldots, len_2\}$. δ is a specific deletion/insertion penalty value chosen by users. h_{i0} and h_{0j} are the initial values needed for the recursive formula in order to compute the entries of the score matrix h_{ij}. x_i and y_j are events at positions i and j from the given sequences. $s(x_i, y_j)$ is the score from the substitution matrix corresponding to events x_i and y_j.

3. Traceback matrix: Elements of this matrix are left, diag or up depending on the corresponding h_{ij} from the score matrix. These entries are built as follows.

$$q(i,j) = \begin{cases} diag\ if & h(i,j) = h(i-1,j-1) + s(i,j) \\ up & if \quad h(i,j) = h(i-1,j) - \delta \\ left & if \quad h(i,j) = h(i,j-1) - \delta \end{cases} \tag{4}$$

This matrix is used to track back from the bottom right corner to the top left corner to find the optimal matching path. Starting from the bottom right element, one moves in the direction given by the element which can be left, up or diag. This leads to another element with its own instruction (up, left or diag). By following the chain of directions the element at the top left corner is reached. The obtained path is the optimal way of matching the two given sequences.

– *Local algorithm*
The aim of local algorithms [9], [10] is to find a pair of most similar segments, from the given sequences. In this algorithm, a matrix of similarity degrees is built based on the following formula:

$$h_{i0} = h_{0j} = h_{00} = 0, \tag{5}$$

where h_{i0}, h_{0j} and h_{00} are the initial values for the recursive formula that is used to compute h_{ij}. Note that this is different from the global alignment. The initial values are set to be 0 because in local alignment it is not important where the common segment starts, the aim of the local alignment is to find the most similar segments of two given sequences.

$$h_{ij} = \max\left\{h_{i-1,j} - \delta, h_{i-1,j-1} + s(x_i, y_j), h_{i,j-1} - \delta, 0\right\}, \tag{6}$$

where $s(x_i, y_j)$ is the element of the substitution matrix as presented in the previous paragraph for global alignment.
The i^{th} event in a sequence can be aligned to the j^{th} event in another sequence, or can be aligned to nothing (deletion). This leads to a number of possible matchings of the two sequences. The optimal pair of aligned segments is identified by first finding the highest score in the matrix. This element is the end of the optimal aligned path. Then, the path is filled by tracking back from that optimal highest score diagonally up toward the left corner until 0 is reached.

2.2 KNNs Combined with Sequence Alignment

KNN is one of the classical approaches in data mining [11]. The model automatically selects sequences from the continuously updated data. KNN can be used as original non sequential approach [12] or extended into sequential approach [6]. The core idea is to find similar sequences expecting these sequences have a common behaviour and outcome. This expectation is understandable and is

common, for example in biology, similar DNA or protein sequences are hoped to have the same shape function.

Due to the sequential nature of business processes, we want to extract K similar sequences in terms of their temporal characteristics not numerical quantities. That is why the initial idea is to adopt the sequence alignment approach from biology and combine it with KNNs. To estimate the similarity between two sequences, the longest common segments between them can be used as criterion. The local sequence alignment can be applied directly. Alternatively, two sequences can be aligned globally, this finds the optimal matching by aligning from the first event to the last one of the two sequences. In this case, global matching allows us to compare the sequences in a whole rather than focus on their most similar consecutive segments. The resulting approach is named KNsSA (K nearest sequence with sequence alignment).

3 GOSPADE based Loop Failure Detection

Association rules are popular in areas like marketing, decision making and management support. They are used to detect information in the form of if-then rules or sequential patterns. For instance, retailers want to know which products customers usually purchase together (basket analysis). There are a number of studies in this area covering different aspects: rules of single level concepts, multiple level concepts, usefulness of rules and how to efficiently detect rules from a large data set with consecutive repeated tasks [13], [14], [11], [15], [7], [16], [17], [18].

3.1 Association Rule Mining

The aim of association rule mining is to find frequent patterns in a data set. Given e_i and e_j which are transactions or events, $e_i \Rightarrow e_j$ means if a sequence S_k in S contains e_i then e_j is contained in that sequence with a certain probability. In addition, the number of sequences in which $e_i \Rightarrow e_j$ appears is required to be larger than a given minimum support.

Definition 1. *The support of an event or association rule is the ratio of the number of sequences in the data sample which contain the considered event/rule with the number of all sequences.*

Definition 2. *The confidence of an association rule is the ratio of its support with the support of.*

There are a number of algorithms in association rules. Some of them were developed to capture the taxonomy structure of the problem, some of them are suitable to deal with repetition in the data:

- A-priori algorithm and some of its extensions [14] (basic).
- Generalizations of sequential patterns algorithm [18] (a-priori extension).
- ML (multiple levels) algorithm and some variations [15] (taxonomy).
- SPADE and GO-SPADE algorithms [7] (repetition).

3.2 GOSPADE Algorithm Variant for Loop Failure Detection

Intuitively, repetitions might slow down the completion of a process. Hence, we would like to look into the data and search for rules related to task repetition (loops) and the outcome of the process (success/failure). The GO-SPADE algorithm was designed to deal with consecutive data and it is suitable for detecting loops. We now introduce some concepts in association rule mining in order to help us to illustrate the principle of the GOSPADE algorithm which is briefly discussed afterwards:

Definition 3. *A rule of length k (composed of k elements) is called k-frequent sequence (pattern).*

Definition 4. *Prefix p of a a k-frequent rule z is the subsequence of z which consists of the first* $(k-1)$ *elements.*

Definition 5. *Suffix s of a rule z is its last element.*

Definition 6. *The location information of a pattern is listed and stored in a table, called idList. This information consists of the Id of the sequence in which the pattern occurred, the position of the pattern in the sequence and the rule itself (each pattern has its own idList). These sequence Ids and positions are denoted sid and eid respectively (please see Tables 1). While scanning through the data, each time the considered pattern occurs, a new row is added to the list for storing the information about the Id of the corresponding data sequence as well as the position of the pattern in the sequence. If the pattern is of length 1 then eid is the position of the task in the given sequence, otherwise eid is the position of the last task in k-frequent sequences when* $k > 1$*. To present the repetition of a pattern, instead of storing repeatedly it in different rows in the list, only one row is added to store the pattern by listing all of the eids. Since the information of the sequence Id and the pattern itself are the same for these repetitions. The eids of the replaced pattern are then written in interval form* $[i, i+j]$*, where* i*,* $i+1, \ldots, i+j$ *are the eids of the consecutive repeats of the pattern. If the pattern occurs only once at position* i*, we write* $[i, i]$*.*

The GO-SPADE algorithm consists of two stages:

- Generate candidates step: in the original algorithm, SPADE distinguishes two types of patterns, event patterns and sequence patterns. These types of patterns are determined based on the relationship between a prefix and its corresponding suffix. If the last item of the prefix p occurs at the same time as the suffix s, the pattern is called event pattern and is denoted ps. If the suffix s occurs strictly after the last item of the prefix p then the pattern is called sequence pattern and it is denoted $p \rightarrow s$. In our case, the available data consists of sequences of events. As mentioned in the introduction parallel structures are not considered and we are looking at sequence patterns. However, we work with loops only so the only IdList we have to investigate

is IdList of length 1. It then does not matter if we are dealing with event patterns or sequence patterns.
Let us consider event patterns for the sake of explaining the algorithm in a simple way. For more detail, refer to [7]. In the first step, candidate frequent patterns are generated. To generate frequent patterns of length $k+1$, all frequent patterns of length k which have the same prefix $k-1$ are considered. Once we have all k frequent patterns with the same $k-1$ prefix, the corresponding IdLists are used. Consider two k-sequences $z_1 = (p_1, s_1)$, $z_2 = (p_2, s_2)$, $p_1 = p_2$ and the corresponding IdLists IdList(z_1) and IdList(z_2). If prefixes p_1 and p_2 are the same, z_1 and z_2 are merged to generate a new frequent sequence z. For instance, the generated sequence is $z = p_1 s_1 s_2$.
To compute the IdList for the generated sequence IdList(z), a join operation is used. If sid1 is the same as sid2 then compare the eid of suffixes s_1, s_2 if eid1 is smaller than eid2 then chose eid2 as the eid of the new frequent sequence in case the support of generated sequence is bigger than minimum support.

– Counting support step: in GO-SPADE, it is easy to count the support of the generated sequences. It does not need to go through the data base to count the support of a candidate sequence. All the required information is already stored in its IdList.

Example: given four process instances (sequences) $P1$: $AAACB$ - success, $P2$: BAB - failure, $P3$: BCD - success, and $P4$: $ABBCDD$ - failure. Assume that we want to detect loops and just need to generate the IdLists of single events. The 1-IdList(A) generated from the given sequences is illustrated in Tables 1: Based on the IdLists for A, B, C and D, we select all the loops to generate a

sid	eid	Task	Loop
P1	[1, 3]	A	L
P2	[2, 2]	A	No
P4	[1, 1]	A	No

Table 1. The corresponding IdList of pattern of length 1 IdList(A).

new list as illustrated in Table 2. Based on the resulting list, we compute the percentages of failure and success. This GOSPADE based technique is the loop failure detection technique.

4 Evaluation

4.1 Data Preparation

We carried out a number of experiments based on records from three real processes ($DS1-3$) from a multi-national telecommunications company. In these datasets, the population of process instances turned out to be very diverse and

sid	eid	Outcome	Task	Loop	Support	Confidence
P1	[1, 3]	S	A	L	25%	1.0
P4	[2, 3]	F	B	L	25%	1.0
P4	[5, 6]	F	D	L	25%	1.0

Table 2. Loops and corresponding process outcomes found by LFD technique from a studied data set.

not straightforward to work with. $DS1$ represents a small scale set with only 10000 entries, 633 process instances, and hundreds of unique tasks. $DS2$ is also a real process with 11839 entries, 576 process instances with different lengths, and also has hundreds of unique tasks. The lengths of process instances in $DS2$ vary considerably. Last of all, $DS3$ is a customer behaviour data set used for churn prediction. This data set consists of 8080 customer records, each record is a sequence of events related to a customer. There are four types of events in churn data, churn, complaint, repair and provision. These customer event sequences are also of different lengths. Among these 8080 sequences, only 161 sequences contain a churn event. This shows the skewness of the data. Therefore, we artificially created new datasets with different ratios between non-churn and churn sequences in order to reduce the skewness of the data and to investigate if it had strong influence on the prediction model.

As we aim to predict the process outcome and to provide warning about process failure, it is necessary to label the outcome as either success or failure and this needs to be done before the proposed method can be applied. It is non-trivial to define process success or failure. In the case of $DS2$, the difference between the actual delivered date and the delivered date promised to the customer is used as the criterion to determine the success and failure. Particularly, if the actual delivered date is before the agreed date, that process instance is classified as success, otherwise, it is classified as failure. In contrast to $DS2$, $DS1$ and $DS3$ (churn and no churn labels) have available labels and therefore, they can be used directly as input for the proposed model.

4.2 Results for Predictive Models

To evaluate KnsSAs, we benchmarked our models with two other approaches:

- *RM - Random Model*: in order to find the outcome of the process, we randomly generate a number between 0 and 1, if the generated number is greater than 0.5 the outcome is success (1) and vice versa if the generated number is smaller than 0.5 the outcome is failure (0).
- *Original KNN*: we chose K nearest sequences in terms of having common unique tasks. For example, given two sequences ABD and AAC, there are one A, one B and one D in the first sequence, there are two A's and one C in the second one. Each unique task can be considered as one category, distance for each category is computed then the sum is taken to obtain the total distance between any two given sequences. For instance, the two

sequences given above consist of four categories A, B, C and D. The distance of category A is $d_A = 1$, and those of categories B, C, D are $d_B = 1$, $d_C = 1$ and $d_D = 1$ respectively. The resulting total distance of these two sequences is $d = d_A + d_B + d_C + d_D = 4$.

For the proposed models, we investigate the effect of K as it is important to get a reasonable number of similar sequences. As the labels are 0, 1, we decide to select odd values for K so we can always extract the outcome/label of the given sequence based on the K obtained sequences. The data sets $DS1$ and $DS2$ have a large number of unique tasks and the difference between the lengths of the sequences is substantial. Intuitively, the value of K should be small taking into account the diversity of the data. The results of the local KnsSA applying on data sets $DS1$ and $DS2$ are presented in Figure 1:

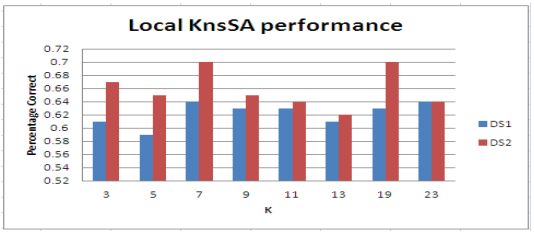

Fig. 1. Percentage of correct predictions of local KnsSA using data sets DS1, DS2.

We then tested our global KnsSA using the same datasets. The results are illustrated in Figure 2

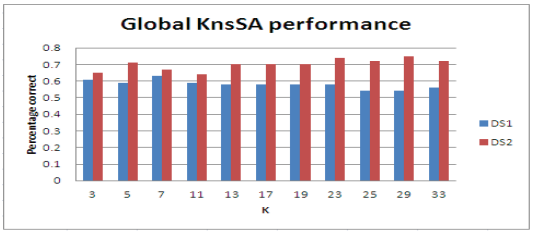

Fig. 2. Percentage of correct predictions of global KnsSA using data sets DS1, DS2.

Figures 1 and 2 show that both global KnsSA and local KnsSA are more accurate when they are applied to $DS2$. For $DS2$, global KnsSA with a higher value of K provides a better performance. When $K = 29$ the performance of the global KNsSA is 75%. On the other hand, applying global KnsSA to $DS1$ did not

achieve similar high performance. The highest correct percentage obtained is only 64% with $K = 7$. Moreover, when K is increased, global KnsSA's performance on $DS1$ decreases. This can be explained as $DS1$ is more diverse than $DS2$.

Global and local KnsSAs do not show any difference when they are applied to $DS1$. However, they show some difference in the case of $DS2$. This indicates that there are no important segments which have influence on the process outcome in $DS1$.

The performances of the original KNN, random guess and the proposed models, local and global KnsSAs, applied to the three data sets are presented in Figure 3:

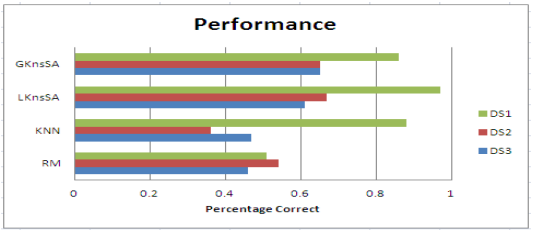

Fig. 3. Percentage of correct predictions of different models on data sets DS1, DS2 and DS3.

The results show that the proposed models outperform both benchmark models, original KNN and random guess. Especially, in the churn data, the proposed models capture churn event with a high degree of success whilst the other models do not. This also implies that the temporal characteristics of the data is important for predicting the process outcome.

We now present the results of the experiments by applying the two models, global and local KnsSAs to the churn data, $DS3$. The results are presented differently from the results for $DS1$ and $DS2$. This change is mainly made for the sake of dealing with the skewness of the churn data, the proportion between class 0 and class 1 in $DS3$ is 98:2 whilst that proportion for $DS1$ and $DS2$ is about 40:60. There are four tables which illustrate different aspects of the experiments' objectives. Table 3, shows the difference obtained by varying K, using local KnsSA and the original churn data. Table 4 demonstrates the results obtained by using local KNsSA when artificial data were created by changing the ratio between churn and no churn sequences in order to decrease the skewness of the data. Table 5 shows the performance of global KnsSA when applied to the former artificial data sets.

It can be seen from Table 3 that $K = 3$ is the best case for the churn data as our dataset is not big. Also, when the original data were modified in order to reduce its skewness, the performance of the model in terms of the churn detection objective improved even though the overall performance worsened. Intuitively,

Results/K	3	5	7
Total test data	809	809	809
Actual tests successful	793	793	793
Actual tests failure	16	16	16
Predicted tests successful	803	805	805
Predicted tests failure	6	4	4
Predicted tests success correct	793	793	793
Predicted tests failure correct	6	4	4
Correct ratio	0.99	0.99	0.99

Table 3. Local KnsSA applied on original DS3 with different K, $K = 3$, 5 and 7

Results/ratio	0.05	0.10	0.15
Total training data	511	875	1189
Total test data	59	100	135
Actual tests successful	45	83	120
Actual tests failure	14	17	15
Predicted tests successful	45	84	124
Predicted tests failure	14	16	11
Predicted tests success correct	44	80	118
Predicted tests failure correct	13	13	9
Correct ratio	0.97	0.93	0.94

Table 4. Local KnsSA applied on original DS3 with different churn and non churn ratios $K = 3$

when the population of churn sequences strongly dominates, it is very likely that our model could not catch the full churn set. It is shown in Table 3 that there are only 6 predicting cases with churn and all of them are correct. In the actual test data, there are 16 cases with churn. Although our model did not catch all of them, it achieves 100% correctness regarding the churn cases that it predicted. With the amended data, the overall performance of the model is reduced as well as the prediction rate for churn. Nonetheless, it is still of interest because out of 14 churn cases in the testing data, our model predicts 14 churn cases and 13 of them are correct. This amended data set consists of 161 (roughly 2% of the original data set) churn cases and 10% of the non churn cases of the original data set.

The results in Tables 4 and 5 show that the local KnsSA outperforms the global KnsSAs when applied to the churn data set. It might be caused by the fact that in customer behaviour sequences, only a set of special segments has strong influence on churn action.

4.3 Results for Warning Technique LFD

The results of the experiments on analysing the relationship between loops and process outcome using LFD applied to $DS1$, $DS2$ and $DS3$ are illustrated in the Tables 6, 7 and 8 correspondingly.

Results/ratio	0.05	0.10	0.15
Total training data	504	839	1208
Total test data	58	95	137
Actual tests successful	43	79	121
Actual tests failure	15	16	16
Predicted tests successful	48	90	134
Predicted tests failure	10	5	3
Predicted tests success correct	38	78	118
Predicted tests failure correct	5	4	0
Correct ratio	0.74	0.86	0.86

Table 5. Global KnsSA applied on original DS3 with different churn and non churn ratios with $K = 3$

Success-Loop 72.88%, Failure-Loop 27.12%						
sid	eid	Outcome	Task	Loop	Support	Confidence
9	[1, 2]	F	A	L	0.31	1.00
13	[2, 3]	S	A	L	0.17	1.00
17	[2, 3]	S	A	L	0.31	1.00
23	[2, 3]	S	A	L	0.31	1.00

Table 6. Loops and corresponding process outcomes found by the LFD in DS1.

Success-Loop 17.17%, Failure-Loop 82.83%						
sid	eid	Outcome	Task	Loop	Support	Confidence
416	[41, 42]	F	C	L	0.19	1.00
416	[1, 2]	F	B	L	0.21	1.00
421	[15, 16]	F	B	L	0.21	1.00
471	[16, 17]	S	B	L	0.21	1.00

Table 7. Loops and corresponding process outcomes found by the LFD in DS2.

Success-Loop 87.66%, Failure-Loop 12.34%						
sid	eid	Outcome	Task	Loop	Support	Confidence
1042	[1, 3]	F	4	L	0.89	1.00
1043	[1, 2]	S	4	L	0.89	1.00
1044	[1, 3]	S	4	L	0.89	1.00
1047	[1, 2]	S	4	L	0.89	1.00

Table 8. Loops and corresponding process outcomes found by the LFD in DS3.

As the goal is to verify if there is a link between a specified pattern and the outcome, specifically there is a link between a specified loop and failure, minimum support and minimum confidence are varied in order to filter out loops which are not frequent and not interesting. When minimum support is raised, only loops with high frequency are considered. In general, there is no loop in the data sets $DS1$ and $DS2$ which has high support and confidence.

The experiments don't show the link between loops and failure as expected whilst applying LFD to $DS1$. It is understandable as the loop task is 'contacting customer'. Obviously, people who executed this process tried to provide good service by repeating the task. In the case of $DS2$, the results of the experiments, in contrast, show the link between loops and failure. $DS3$ is a special case, there is no link between loops and failure or success because in general there are loops in each customer behaviour sequence, loops in task 2 and task 4 have 84% and 89% support respectively.

5 Conclusions

KNNs as a classical data mining approach have been widely used for modelling and predicting customer behaviour. This paper addresses some shortcomings of these predictive models, which occur when they are used for sequential data. Particularly, we propose some extensions to KNNs. These extensions were introduced in order to capture the temporal characteristic of the data and to profit from KNNs ability to deal with diverse data. Our extensions are tested on real business process data from a multi telecommunications company and the experiments provide some interesting results. First, the influence of global and local algorithms change depending on the data employed. For example, in the $DS1$ dataset, there is not much difference between using global KnsSA and local KnsSA. However, for $DS2$, global KnsSA is more accurate and in the case of $DS3$ local KnsSA outperforms global KnsSA. Even though the highest performance when predicting process outcome of the proposed models is just 75%, it outperforms the original KNNs, which proves that it is important to use sequential data in our problem.

Of all experiments, churn prediction shows the most interesting result. As churn is a rare event (2% of the data consist of churn), in the testing data set, there are only around 15 churn cases. Our models correctly capture most of these cases and percentage correct of churn prediction is very high. In the case of local KnsSA with $K = 3$, applied to $DS3$, out of 14 churn cases, the model predicts that there are 14 churn cases and 13 of them are correct.

It is interesting to see how the link between loops and process failure changes from case to case (different processes). In $DS1$, a specific loop implies that it is likely the process instance will end up with success. In $DS2$, there are certain loops which lead to process failure with high probability.

The experimental results encourage us to adopt the strategy of first grouping data into similar groups and then dealing with them using suitable approaches in our future work.

References

1. van der Aaslt, W.: Business process management: A personal view. Business Process Management Journal **10**(2) (2004) 135–139
2. van der Aaslt et al, W.: Process mining manifesto. In Daniel, F., Barkaoui, K., Dustdar, S., eds.: Business Process Management Workshops 2011. Volume 99 of Lecture Note in Business Information Processing. Springer, Berlin (2011)
3. Weijters, A., van der Aalst, W.: Process mining discovering workflow models from event-based data. In: ECAI Workshop on Knowledge Discovery and Spatial Data. (2001) 283–290
4. van der Aaslt, W., Schonenberg, M., Song, M.: Time prediction based on process mining. Information Systems **2** (2011) 450–475
5. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley, New York (2001)
6. Ruta, D., Nauck, D., Azvine, B.: K nearest sequence method and its application to churn prediction. In Corchado, E., Yin, H., Botti, V., Fyfe, C., eds.: Intelligent Data Engineering and Automated Learning IDEAL 2006. Volume 4224 of Lecture Notes in Computer Science. Springer, Berlin (2006) 207–215
7. Leleu, M., Ligotti, C., Boulicaut, J., Euvrard, G.: Go-spade: Mining sequential patterns over datasets with consecutive repetitions. In: Conference on Machine Learning and Data Mining in Pattern Recognition. (2003) 293–306
8. Needleman, S., Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology **48** (1970) 443–453
9. Smith, T., Waterman, M.: Identification of common molecular subsequences. Journal of Molecular Biology **147** (1981) 195–197
10. Waterman, M.: Estimating statistical significance of sequence alignments. Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences **344** (1994) 383–390
11. Berry, M., Linoff, G.: Data Mining Techniques: for Marketing, Sales, and Customer Relationship Management. Wiley, Newyork (2004)
12. Eastwood, M., Gabrys, B.: A non-sequential representation of sequential data for churn prediction. In: Proceedings of the KES2009 Conference, Santiago, Chile (2009)
13. Agrawal, R., Srikant, R.: Fasts algorithms for mining association rules in large databases. In: VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases. (1994) 487–499
14. Agrawal, R., Srikant, R.: Mining sequential patterns. In: ICDE '95 Proceedings of the Eleventh International Conference on Data Engineering. (1995) 3–14
15. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: VLDB '95 Proceedings of the 21th International Conference on Very Large Data Bases. (1995) 420–431
16. Finding Interesting Rules from Large Sets of Discovered Association Rules. In: CIKM '94 Proceedings of the Third International Conference on Information and Knowledge Management. (1994)
17. Mohammed, J.Z.: Spade: An efficient algorithm for mining frequent sequences. Machine learning **42** (2001) 31–60
18. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: 5th International Conference on Extending Database Technology: Advances in Database Technology. (1996) 3–17