Expert Systems

Article

DOI: 10.1111/exsy.12079

A hybrid model for business process event and outcome prediction

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

(1) School of Design, Engineering and Computing, Bournemouth University, Bournemouth, UK
E-mail: mai.phuong@bt.com; bgabrys@bournemouth.ac.uk
(2) BT Research and Technology, Ipswich, UK
E-mail: detlef.nauck@bt.com

Abstract: *Large service companies run complex customer service processes to provide communication services to their customers. The flawless execution of these processes is essential because customer service is an important differentiator. They must also be able to predict if processes will complete successfully or run into exceptions in order to intervene at the right time, preempt problems and maintain customer service. Business process data are sequential in nature and can be very diverse. Thus, there is a need for an efficient sequential forecasting methodology that can cope with this diversity. This paper proposes two approaches, a sequential k nearest neighbour and an extension of Markov models both with an added component based on sequence alignment. The proposed approaches exploit temporal categorical features of the data to predict the process next steps using higher order Markov models and the process outcomes using sequence alignment technique. The diversity aspect of the data is also added by considering subsets of similar process sequences based on k nearest neighbours. We have shown, via a set of experiments, that our sequential k nearest neighbour offers better results when compared with the original ones; our extension Markov model outperforms random guess, Markov models and hidden Markov models.*

Keywords: artificial intelligence, method, system

1. Introduction

Very often, processes are poorly documented because they have evolved over time or the legacy information technology systems that were used to implement them may have changed. This adds the additional challenge to identify the actual process model. Process mining (van der Aalst, 2011) can reconstruct process models from workflow data to some extent by searching for a model that explains the observed workflow. This process model contains the list of all unique process prototypes (workflow sequences). Data from this mining process can help to address the problem of proactive process monitoring and process event prediction.

Firstly, in order to predict process events, we consider approaches from data mining (Berry & Linoff, 2004). Because of the nature of the data, approaches that can deal with temporal sequence data are the most relevant. One example from this class of approaches is Markov models (MMs) that have been used to study stochastic processes. In the work of Pitkow and Pirolli (1999), for example, it has been shown that Markov models are well suited to the study of Web-users' browsing behaviour. Event sequences are used to train a Markov model that encodes the transition probabilities between subsequent events. Event sequences can represent customer behaviour, customer transactions, workflow in business processes and so on. The prediction of the continuation of a given sequence is based on the transition probabilities encoded in the Markov model.

The order of a Markov model represents the number of past events that are taken into account for predicting the subsequent event. Intuitively, higher order Markov models are more accurate than lower order Markov models because they use more information present in the data. This has been shown to be true in the prediction of Web browsing behaviour. Lower order models may not be able to discriminate the difference between two subsequences, especially when the data are very diverse, that is if there are a large number of unique sequences. In the work of Ruta and Majeed (2011), it has been shown that higher order Markov models can be more accurate than lower order models in the context of customer service data.

However, higher order models can suffer from weak coverage in particular when the data are diverse and not clustered. For sequences that are not covered by the model, a default prediction is required. Default predictions typically reduce the accuracy of the model. It is obvious that with increasing order of the model, the computational complexity also grows. In the case of plain Markov models, there are already approaches to deal with the trade-off between coverage and accuracy. These approaches have been introduced in the studies of Deshpande and Karypis (2004), Eirinaki *et al.* (2005) and Pitkow and Pirolli (1999). The general idea is to merge the transition states of different order Markov models, which is followed by pruning 'redundant' states. A particular method is the selective

Markov model, an extension of All *K*th order Markov models (Deshpande & Karypis, 2004).

Secondly, to predict the process outcomes, bearing in mind that the data can be very diverse, our idea is to allocate a number of sequences from the historical data that are most similar to the given sequence expecting that they would have similar outcome. *K* nearest neighbour (KNN) is chosen as the predictive model here. In our area of application, we consider a number of sequences in which we want to find K sequences that are most similar to a given sequence $S\left(s_1^{(j)}, \ldots, s_{n_j}^{(j)}\right)$, where j is the index of the sequence. The similarity is determined using a distance function, and the prediction should be the majority class among the classes of *K* nearest sequences. This method is used by Ruta *et al.* (2006) to predict churn using data from a telecommunication company. The authors combined KNNs and the theory of survival. In their work, the authors used Euclidean distances to calculate the distance between the given sequence and all sequences in the data sample. In this study, as our data consist of event sequences, we use the edit distance in the process of comparing two sequences. In order to exploit the temporal characteristics of process data, we combine KNN with sequence alignment from biology to form a sequential KNN.

In this paper, we first present a hybrid approach to address the lack of coverage in higher order Markov models. This approach is a combination of pure Markov models and sequence alignment technique (Waterman, 1994). The sequence alignment technique is applied when the default prediction is required. This is the case when the given sequence cannot be found in the transition matrix. The matching procedure is applied in order to extract those sequences (patterns) from the transition matrix that are most similar to the given sequence. We determine the prediction for the given sequence based on the predictions for the sequences we obtained. We secondly present a sequential KNN approach that is used to predict process outcomes. Our sequential KNN matches sequences using local alignment. The output of the matching is used to rank the similarities of sequences.

We are testing our approaches on workflow data from a telecommunication business process, but they are likely to be applicable in a variety of other sequential prediction problems – especially in the case of our extended Markov model if Markov models have already been used and shown to be relevant.

The rest of this paper is organised as follows. Sequence alignment and distance measurement are presented in Section 2. Section 3 introduces the predictive models that are used in this study. Section 4 then presents the experimental results and evaluation. Finally, in Section 5, our conclusions and directions for future work are discussed.

2. Sequence alignment

To determine sequence similarity, both distance measures and similarity measures can be used because distance and similarity are dual concepts. For numeric variables, well-known distance measures exist and can be easily applied. However, in sequence analysis, sometimes, we have to work with sequences that are constructed from symbols, for example, categories (churn prediction), phonemes (speech recognition) or characters (hand-writing recognition). In this case, we need specialised functions that have the ability of measuring the similarity of symbolic sequences.

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 and so on. Sequence alignment is used for a number of purposes (Needleman & Wunsch, 1970); for example, to compare new DNA sequences with DNA databases. Transactions of DNA sequences into amino acid sequences are compared with protein databases in order to verify if the relationships that are found between them could have occurred by chance and so on. Algorithms used in sequence alignment are mainly divided into global alignment and local alignment. Global alignment provides a global optimisation solution, which spans the entire length of all query sequences. In contrast, local alignment aims to find the most similar segments from two query sequences. We use the idea of sequence alignment in Markov models but do not use existing sequence alignment algorithms. Instead, we build a simple alignment procedure for short subsequences. We also use the local alignment algorithm and combine it with KNNs. The local algorithm is presented in the following:

Given two sequences, we 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, $i = \{1, \ldots, len_1\}$, $j = \{1, \ldots, len_2\}$ where len_1 and len_2 are the lengths of the two given sequences. The aim of the local algorithm (Smith & Waterman, 1981; Waterman, 1994) is to find a pair of most similar segments in the given sequences. There are two key matrices used in local alignment: the substitution matrix and the score matrix.

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 one amino acid transforming (mutating) into another. There are different ways of generating the substitution matrix. The simplest way is to not take into account amino acid mutations and 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.

$$s(i,j) = \begin{cases} 0 & \text{if} \quad \text{event } i \neq \text{event } j \\ 1 & \quad\text{otherwise} \end{cases}$$

   In this case, the substitution matrix is an identity matrix, the elements of the main diagonal are 1 and all the others are 0. To present mutations, a more complicated form of substitution matrix is used. The elements of the matrix off the main diagonal can have a value different from 0, depending on the likelihood and frequency of transformation between the two amino acids.

2. Score matrix: A score matrix is built based on the following formula:

$$h_{i0} = h_{0j} = h_{00} = 0 \tag{1}$$

where h_{i0}, h_{0j} and h_{00} values are the initial values for the recursive formula that is used to compute h_{ij}.

$$h_{ij} = \max\left\{h_{i-1,j} - \delta, h_{i-1,j-1} + s\left(x_i, y_j\right), h_{i,j-1} - \delta, 0\right\} \tag{2}$$

where 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. Example: Given two sequences $ABCDE$ and $EBCAD$, the corresponding score matrix is as follows:

$$\left(\begin{array}{cccccc} & A & B & C & D & E \\ E & 0 & 0 & 0 & 0 & 1 \\ B & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 0 & 2 & 0 & 0 \\ A & 1 & 0 & 0 & 1 & 0 \\ D & 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

The ith event in a sequence can be aligned to the jth 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 towards the left corner until 0 is reached. In the example, the best match is BC.

3. Predictive models

This section presents Markov models, KNNs and their extensions that can help enhance the current approaches to process management in a telecommunication business context by predicting future events and process outcomes based on the current and past data. Here, a business process instance (S_j) is a composition of discrete events (or tasks) in a time-ordered sequence, $S_j = \left\{s_1^{(j)}, s_2^{(j)} \ldots s_{n_j}^{(j)}\right\}$ where s_j takes values from a finite set of event types $E = \{e_1, \ldots, e_L\}$. Apart from its starting time $t_i^{(j)}$ and duration of $T_i^{(j)}$, 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 goals of our predictive models are to predict the next event $s_{i+1}^{(N+1)}$ following event $s_i^{(N+1)}$ in 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 completed process instances $S = \{S_1, S_2 \ldots S_N\}$ or to predict the process outcomes (success/failure). A simple predictive model is illustrated in Figure 1.

In the following subsections, we will discuss some examples of process prediction models, namely Markov models, hybrid Markov models (MSAs), hidden Markov models (HMMs), KNNs and sequential KNNs (KnsSA).

3.1. Markov models

Markov models are a kind of stochastic model. They are based on the mathematical theory of Markov chains. Markov models (Papoulis & Pillai, 1991) have been used in studying stochastic processes in terms of modelling and predicting customer behaviour. The idea of the model is to use $k > 0$ steps to predict the following $(k+1)$th step. Given a sequence of random variables $\{X_n\}$, a first order Markov model uses the current step x_{i-1} to predict the next step x_i

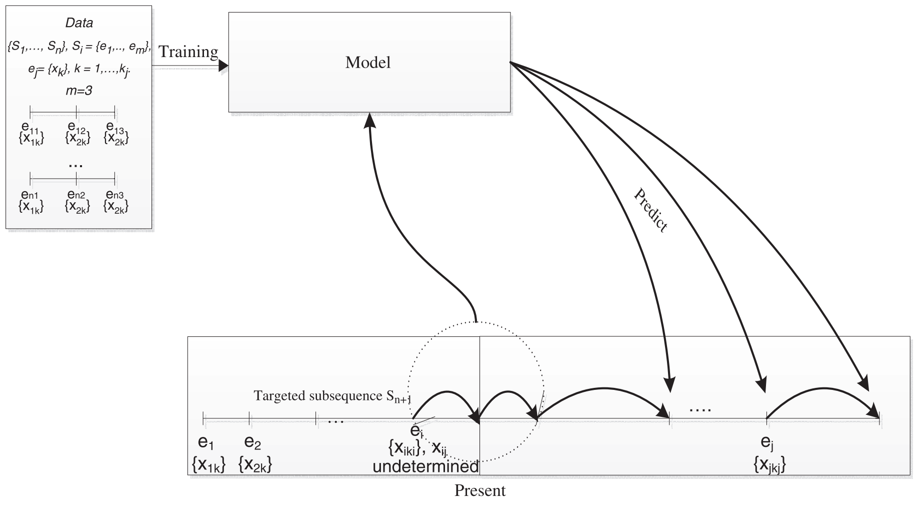

Figure 1: *A simple predictive model.*

and to generalise; a kth order Markov model uses the last k steps to predict the next one:

$$p(x_i|x_{i-1}, \ldots, x_{i-n}) = p(x_i|x_{i-1}, \ldots, x_{i-k}). \tag{3}$$

To construct a Markov model a matrix M, $M = (m_{ij})$ is created containing the probabilities for moving from the previous k steps to the next. The columns of the matrix correspond to the unique tasks, $i = (1, \ldots, I)$, and the rows of the matrix correspond to the unique patterns of length k that are present in the training data set that is used to construct the model. These unique patterns are called states $j = (1, \ldots, J)$. Elements m_{ij} of matrix M are the probabilities p_{ij}:

$$p_{ij} = \frac{n_{ij}}{n_i}. \tag{4}$$

where n_i is the number of times pattern i occurs in the data set, and n_{ij} is the number of times pattern i is followed by step j in the data set.

Higher order Markov models can be expected to be more accurate. However, we have to take into account their weak coverage property. For example, consider using a Markov model to predict Web page access and consider a sequence of Web pages $\{P_1, P_2, P_1, P_3, P_2, P_1\}$ that is used to create the model. In a second order Markov model, there would be no sample $\{P_2, P_3\}$. Consequently, the prediction for this pattern would have to be the default prediction of the model, that is, the most frequent step (P_1 in this example). One approach to overcome the problem is to merge the transition states of different order Markov models, after pruning the 'redundant' states. A particular method is the selective Markov model, an extension of all Kth order Markov models (Deshpande & Karypis, 2004).

Markov models are often not dynamic, and once the model is trained, the obtained matrix is fixed and used for predictions. It is important that the training set data are large enough to be representative for the encountered patterns. In order to accommodate changes in data, the matrix can be rebuilt from scratch after a certain period. It is straightforward to create a dynamic Markov model that adapts to new data by storing the counts n_{ij} and n_i and updating M with additional rows and/or columns if new patterns and/or steps are encountered in the new data. We can also apply a discounting factor to the current counts to give more weight to the pattern frequencies in new data.

3.2. Hidden Markov models

In the field of sequential data, HMMs are a very powerful tool. Similar to Markov models, HMMs are a type of stochastic model. They are also based on the mathematical theory of Markov processes that further developed into the theory of HMMs by Baum in the 1960s (Baum *et al.*, 1970). HMMs received much attention because of their application in speech recognition. Moreover, there is a wide range of applications ranging from telecommunication, recognition (gesture, speech and so on) to finance, biology and so on.

Many approaches in temporal data series only deal with observed variables. We can consider how the performance of the model changes after adding an unobservable or hidden variable that we assume to have an influence on the observed variables. HMMs assume that there is a latent variable that influences the values of the observed variable. The benefits of this approach are shown, for example, in the work of Cox and Popken (2002).

As an example, for an application domain, consider modelling customer behaviour based on customer event sequences (e.g. historic purchases and contacts with customer service). A latent variable that influences customer behaviour could be the level of customer satisfaction.

Let $O = \{o_1, \ldots, o_N\}$ be the sample of observed variables and Q a sequence of hidden states $\{q_1, \ldots, q_N\}$. The observed variable can be continuous and follow a (usually Gaussian) distribution, or it can be discrete and take values from a finite set $V = \{v_1, \ldots, v_K\}$. The hidden states are discrete, and their value space is Ω, $\Omega = \{s_1, \ldots, s_M\}$. The expression of the joint probability of observed sequence is as follows:

$$p(o_1, o_2, \ldots, o_n|q_1, \ldots, q_n) = p(q_1)\prod_{i=2}^{n} p(q_i|q_{i-1})\prod_{i=1}^{n} p(o_i|q_i) \tag{5}$$

Because of the complexity of the computation, lower order hidden Markov models are more popular. The most popular one is the first order hidden Markov model that assumes that the current state depends only on the preceding state and is independent from the earlier states.

$$p(q_n|q_{n-1}, o_{n-1}, \ldots, q_1, o_1) = p(q_n|q_{n-1}) \tag{6}$$

Intuitively, higher order HMMs can be expected to be more accurate (Thede & Happer, 1999). However, the lack of coverage problem needs to be considered, as mentioned in Section 3.2. It is obvious that with increasing order of the model, we are facing more complex computation. An HMM is defined by a set of three parameters $\lambda = (A, B, \pi)$:

$$A = (a_{ij}) = \left(p\left(q_i|q_j\right)\right) \tag{7}$$

where A is the transition matrix and an element a_{ij} is the probability to move from state j to state i. It is necessary to point out that only homogeneous HMMs are considered here, implying the time independence of the transitions matrix. The non-homogeneous type is discussed by Netzer *et al.* (2007). Furthermore,

$$B = (b_{ij}) = \left(p\left(o_i|q_j\right)\right) \tag{8}$$

where B is the matrix containing probabilities of having $o_t = v_i$, denoted as o_i, if the hidden state is q_j at time t. Finally,

$$\pi_j = p(q_1 = s_j) \tag{9}$$

is the probability that s_j is the initial state at time $t = 1$.

There are three fundamental problems in HMMs: estimation, evaluation and decoding. The estimation problem is about finding the optimal set of parameters $\lambda = (A, B, \pi)$ that maximise $p(O|\lambda')$ given a sequence O of observed states and some initial model λ'. The Baum–Welch algorithm is used for this. The evaluation problem means finding $p(O|\lambda)$ given the sequence of observations O and the model λ using the forward (backward) algorithm. To solve the decoding problem, the Viterbi algorithm is applied for finding the most realisable, suitable sequence of hidden states $q_1, \ldots, q_k$ for the associated sequence O and model λ.

Researchers have been working on improvements of the performance of HMMs. Similar to Markov models, there are approaches to deal with the trade-off between coverage and accuracy. However, merging the transition states of higher order hidden Markov models is much more complicated than for Markov models.

3.3. Hybrid Markov models

The work presented in this paper investigates the idea of combining higher order Markov models with a sequence alignment technique (Waterman, 1994) in order to maintain the high accuracy of the higher order Markov models and, at the same time, compensate for their lack of coverage. When a higher order Markov model is employed to predict process events and the given sequence has no match in the transition matrix M, we would be forced to produce a default prediction that would be detrimental to the model accuracy. To this end, we present our approach, MSA for Markov sequence alignment, which is based on the assumption that similar sequences are likely to produce the same outcome. Basically, the steps that MSA took to identify the next step for a previously unseen sequence S_{new} have no match in the transition matrix as follows:

- Compare S_{new} with all sequences (patterns and states) in the transition matrix, and extract l states that are most similar.
- Generate prediction based on these l selected states. For each state i in the transition matrix $M = (m_{ij})$, there is a corresponding predictive row vector, $(m_{i1}, \ldots, m_{iJ})$. The predicted next step of state i is the task j with $m_{ij} = max\ (m_{i1}, \ldots, m_{iJ})$. Thus, the predictions for the l selected states contain l^* unique events, $l^* <= l$, $E^* = \{e_1, \ldots, , e_{l^*}\}$. Then, the final prediction for S_{new} is the most frequent event in E^*. If, however, there is no single most frequent event in E^*, then the one with highest probability is selected. Alternatively, we go back to the l most similar states and select the sequence that is most similar to S_{new} and use its predicted step as the prediction for S_{new}. If there is no single most similar sequence, we select the one with the highest frequency from this set.

In order to compare S_{new} with the sequences in the transition matrix, we need to be able to define and measure their similarities or distances. Here, the sequence alignment technique is applied (Waterman, 1994). In particular, we adopt the edit (deletion and insertion) weighted distance function for its flexibility in determining the degree of similarity. Given two sequences, a weight is assigned to each pair of events at the same position in the two sequences. The weight is 0 if the two events are identical, 1 or δ (a specified value) if they are different. In case both events of the pair must be deleted in order to keep the longest possible similar subsequences, the corresponding weight is 1. If only one of the two events must be deleted, the weight is δ. The sum of weights of the comparison is then used to determine the similarity of sequences. The lower the weight is, the more similar two sequences are.

This algorithm starts by constructing a matrix where the elements represent similarity between any pair of events from the two sequences to be matched. This matrix is then used to find the most suitable editing of the given sequences where deletions and insertions are penalised. The optimal editing is chosen based on the total score computed. For example, given two sequences $ABCDE$ and $EBCAD$, the matrix looks as follows:

$$\left(\begin{array}{cccccc} & A & B & C & D & E \\ E & 0 & 0 & 0 & 0 & 1 \\ B & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 0 & 1 & 0 & 0 \\ A & 1 & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 1 & 0 \end{array}\right)$$

Next, the first events A and E from both sequences are deleted that results in a weight of 1. The second event in both sequences is B and has weight 0. The fourth event A is deleted from the sequence $EBCAD$ in order to match two Ds in fourth and fifth positions in the two sequences. We add δ to the sum of weights. Finally, we delete the last event E from the sequence $ABCDE$, adding another δ. The total weight for matching the two sequences is $w = 1 + 0 + 0 + \delta + \delta = 1 + 2\delta$.

The following example will illustrate our method for the case of the transition matrix of a third order Markov model

$$\left(\begin{array}{cccc} & A & B & C \\ ABC & 0.1 & 0.2 & 0.7 \\ CBC & 0.1 & 0.5 & 0.4 \\ BAA & 0.2 & 0.5 & 0.3 \end{array}\right)$$

and a given sequence BCC. This sequence has not occurred before and is not stored in the matrix. The aims are to find the most similar sequences from the matrix and use their predictions to generate the prediction for the given sequence BCC. We first match BCC to ABC:

$$\left(\begin{array}{cccc} & A & B & C \\ B & 0 & 1 & 0 \\ C & 0 & 0 & 1 \\ C & 0 & 0 & 1 \end{array}\right)$$

The weight of this comparison is $w = \delta + 0 + \delta = 2\delta$. Similarly, the resulting weights of matching CBC and BAA against BCC are $w = 2\delta$ and $w = 2$, respectively. Let $\delta = 0.4$ and $l = 2$, the two chosen sequences in this case are then ABC and CBC and their predictive vectors are $(0.1, 0.2, 0.7)$ and $(0.1, 0.5, 0.4)$, respectively. Based on these vectors, the predictive next steps are $\{B, C\}$. The weights and frequency of occurrence of these two events are equal; hence, the next step of the given sequence BAC is predicted to be C because of higher transition probability, $m_{13} = 0.7$ that is greater than $m_{22} = 0.5$.

Algorithms 1 and 2 illustrate the procedure of matching two sequences by deletion and insertion of symbols with penalties.

3.4. *KNNs combined with sequence alignment*

KNN is one of the classical approaches in data mining (Berry & Linoff, 2004). It is essentially a non-parametric approach; hence, one of its advantages is that there is no training procedure required. The model automatically selects sequences from the continuously updated data. The only configuration possible is a choice of different values for $K \geq 1$, that is, how many nearest neighbours we want to use. KNNs can be used in its original non-sequential form (Eastwood & Gabrys, 2009), or it can be extended into a sequential approach (Ruta *et al.*, 2006). The core idea is to find similar sequences expecting that these sequences have a common behaviour and outcome. This is a reasonable expectation, for example, in biology where similar DNA or protein sequences are expected to have the same shape function.

Algorithm 2 Function calculateScore(n, maxCommonIndexes1, maxCommonIndexes2, maxCommonLength): computes the score of matching two sequences

```
1: totalPoints=0
2: totalDeltas = 0
3: maxCommonIndexes1(maxCommonLength + 1) = n + 1
4: maxCommonIndexes2(maxCommonLength + 1) = n + 1
5: last1 = 0
6: last2 = 0
7: for i = 1 : maxCommonLength + 1 do
8:     dist1 = maxCommonIndexes1(i) – last1 – 1
9:     dist2 = maxCommonIndexes2(i) – last2 – 1
10:    if (dist1 > dist2) then
11:        noPoints = dist2
12:        noDeltas = dist1 – dist2
13:    else
14:        noPoints = dist1
15:        noDeltas = dist2 – dist1
16:    end if
17:    totalPoints = totalPoints + noPoints
18:    totalDeltas = totalDeltas + noDeltas
19:    last1 = maxCommonIndexes1(i)
20:    last2 = maxCommonIndexes2(i)
21: end for
22: return totalPoints + totalDeltas*deltaValue
```

Because of the sequential nature of business processes, we want to extract K similar sequences in terms of their temporal characteristics and not in terms of numerical quantities. This can be seen as a string comparison problem. Our idea is to adopt the sequence alignment approach from biology and combine it with KNNs. The resulting approach is named K nearest sequence with sequence alignment (KnsSA). By combining with sequence alignment, KNNs allow us to sequentially compare symbolic sequences. Given N sequences, our approach builds a distance matrix $(d_{ij})_{N*N}$

Algorithm 1 Function scan(row, col, length): identifies the optimal editing process to match two sequences via deletion/insertion of symbols

```
1: commonIndexes1(length) = row
2: commonIndexes2(length) = col
3: if (length > maxCommonLength) then                ▷ store the longest path
4:     maxCommonLength = length
5:     for i = 1 : maxCommonLength do
6:         maxCommonIndexes1(i) = commonIndexes1(i)
7:         maxCommonIndexes2(i) = commonIndexes2(i)
8:     end for
9:     maxCommonScore =
10:    calculateScore(n,maxCommonIndexes1,maxCommonIndexes2,maxCommonLength)
11: else
12:    if (length == maxCommonLength) then
13:        score = calculateScore(n,commonIndexes1,commonIndexes2, length)
14:        if (score < maxCommonScore) then     ▷ if the paths are equal, store the smallest score
15:            for i = 1 : maxCommonLength do
16:                maxCommonIndexes1(i) = commonIndexes1(i)
17:                maxCommonIndexes2(i) = commonIndexes2(i)
18:            end for
19:            maxCommonScore = score
20:        end if
21:    end if
22: end if
```

storing the alignment scores for matching any two sequences. The elements of the distance matrix are then used to rank the sequences. The elements of the distance matrix are obtained by matching every pair of sequences using local alignment.

4. Evaluation

Having defined MSA and KnsSA, we now present a discussion of our empirical evaluation. We assess the efficiency of our approaches and explore potential future research directions for employing our Markov extension approach in business process event forecasting and our sequential symbolic KNN approach in business process outcome forecasting. Our models have been implemented in MATLAB and run against four data sets of process instances from the records of a major telecommunication company. The first data set (*DS1*) consists of telecommunication line fault repair records for a 9-month period. The second (*DS2*) covers a 1-month period and represents a process for fixing broadband faults. The third data set *DS3* is from a different fault repair process. Data sets *DS1* and *DS2* are used in the experiments of our extended Markov model. Data sets *DS3* and *DS4* are used in the KnsSA experiments.

4.1. Process analysis and data preprocessing

In our data sets, the population of process instances turned out to be very diverse and not straightforward to work with. For instance, *DS1* is very diverse and contains 28963 entries, 2763 unique process instances and 285 unique tasks. The process instance length varies from 1 to 78. On the other hand, *DS2* represents a significantly smaller scale set with only 5194 entries, 794 process instances and 10 unique tasks. The process instance length varies from 2 to 32. *DS3* represents a small scale set with only 10000 entries, 633 process instances and hundreds of unique tasks. *DS4* is also a real process with available labels with 11839 entries, 571 process instances with different lengths and hundreds of unique tasks. The length of process instances in *DS3* and *DS4* varies considerably.

Figure 2 shows a visualisation of 10% of set *DS1*. The diagram has been created by BT's process mining tool Aperture (Taylor *et al*., 2012) that uses workflow data to create a process diagram. For this process, the diagram illustrates the complexity of the workflow data. It is basically impossible to visually analyse or understand the process from this figure. This is a typical scenario for processes that have evolved over time and are poorly documented.

4.2. Extension Markov model experiments

To evaluate MSA, we benchmarked our model with three other approaches:

- *RM – random model*: In order to find the next task following the current one, we randomly select from the set of potential next tasks. For example, if from the

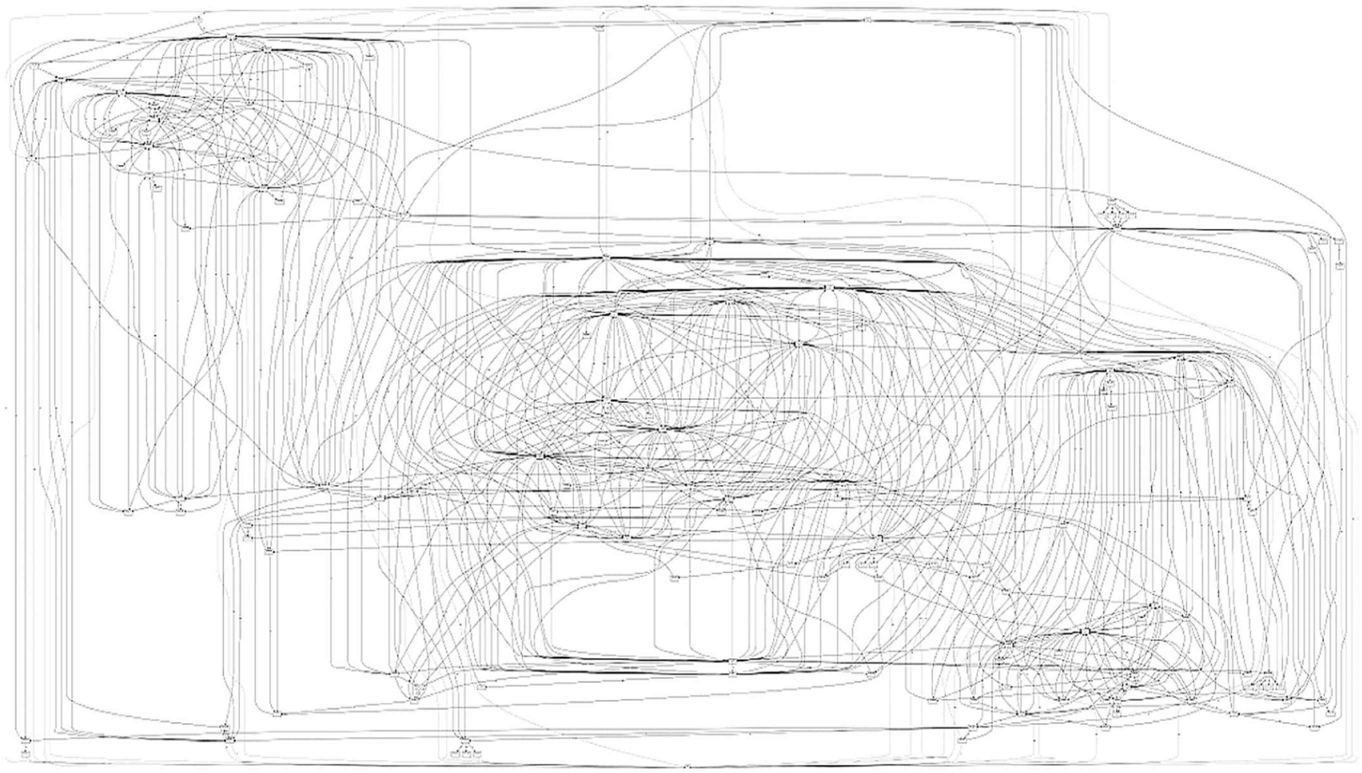

Figure 2: *Process model obtained by using Aperture for DS1 visualising a highly complex process.*

historical data, we know that tasks following A belong to the set $\{C, D, E\}$, we randomly select a value from that set as the predicted next step.

- *All Kth Markov models*: A number of different order Markov models are generated from first order to kth order. Given a sequence, we start with the highest order Markov model. If the given sequence cannot be found in the transition matrix, we create a new shorter sequence by removing the first event in the sequence. We then continue the procedure with the next lower order Markov model until we either find a match in a transition matrix or after trying the first order Markov model, a default prediction is required.
- *HMM – hidden Markov models*: We tested several first order HMMs with different lengths for the input sequences and different number of hidden states and picked the best one.

For each comparison, we used 90% entries of the data sets as training data and the remaining 10% were used as test data. Basically, for each sequence in the test data, we checked if our model can correctly predict the next step of a particular task (i.e. we measured the number of successful predictions). The results were evaluated using 10-fold cross-validation. The results are displayed as a percentage. After the first order to seventh order MSAs were built, we investigated different values for l, the number of sequences (patterns) in the transition matrix that is most similar to the given sequence. l was varied from 1 to 7, and the results show that $l = 5$ is optimal for *DS1* and $l = 3$ is optimal for *DS2*. We now look at the specific results.

Figures 3 and 4 illustrate MSAs accuracy against the two data sets, *DS1* and *DS2*, respectively. The results show that by incorporating the default prediction improvement module, MSA outperforms other comparable models, especially when the order of the Markov model increases. This is because the relevance of the comparison between sequences is directly proportionate to their lengths. As can

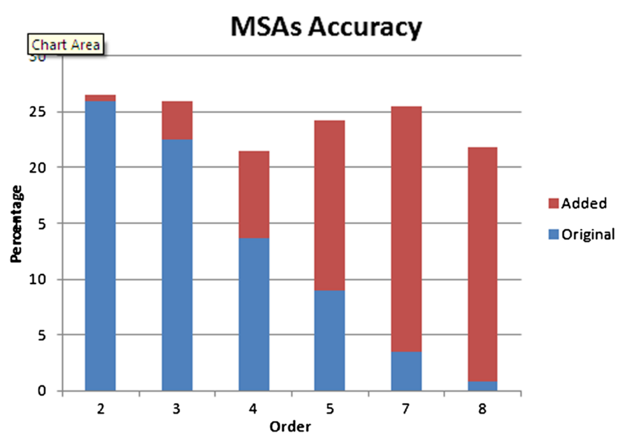

Figure 3: *Percentage of correct predictions before and after applying the default prediction module in Markov models using data set DS1.*

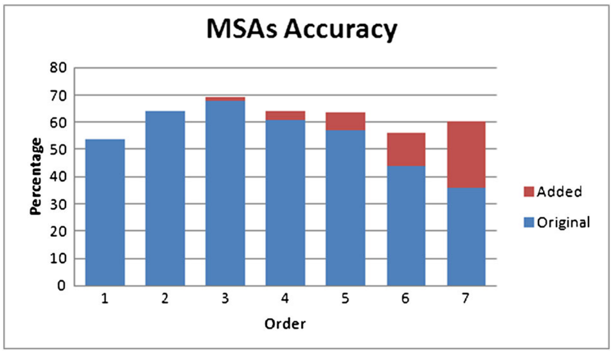

Figure 4: *Percentage of correct predictions before and after applying the default prediction module in Markov models using data set DS2.*

be seen, the second order MSA performs best among a range of different orders of MSA. For the case of *DS1*, it is about 27% correct. The third order MSA works best in the case of *DS2*, it provides correct predictions in about 70% of the cases. Nonetheless, the highest performance order depends on the data set. For the data set that provides a good performance with a relatively high Markov order (i.e. fifth and above), the role of our default prediction improvement module becomes highly significant.

Figure 5 shows the performances of all models against the two data sets. As can be seen, RM performs worst with only around 10% success for *DS2*. When applied to the bigger data set *DS1*, the result goes down to nearly 2% (14 times worse than that of MSAs' average performance). This can be explained by the fact that with *DS2*, each task can have an average of seven potential next tasks, whereas this figure is nearly 30 for *DS1*. Thus, the probability of picking the right next task reduces as the set size increases. The results highlight the difficulty of handling complex data sets. When the data is not too diverse (i.e. *DS2*), MSA (fifth order) obtains the highest number of correct predictions with a result of 63%. Compared with other benchmarks, MSA results are better than a fifth-order Markov model that achieves 57%, an All Kth ($K = 5$) Markov model that achieves 60% and an HMM that achieves 43%. Please note that we did not pick the most

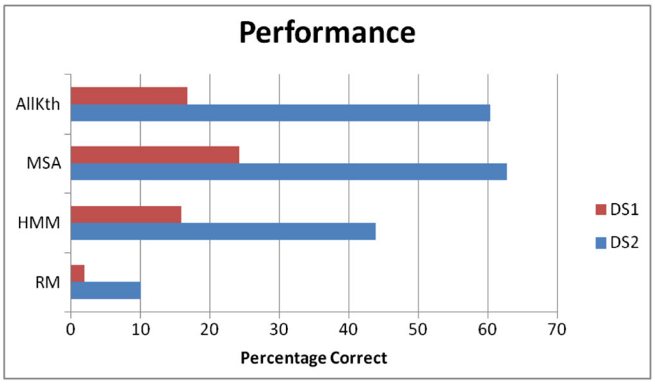

Figure 5: *Percentage of correct predictions of different models on DS1 and DS2 data sets.*

accurate MSA and All *K*th Markov model to compare with the HMM and the RM. The reason is that All *K*th Markov models and MSAs only have an advantage with higher orders. In order to see how well they cope with default predictions, we chose a fifth order MSA and an All fifth Markov model.

When the data is more fragmented as in the case of *DS1*, the effectiveness of our model is reduced. However, even though the performance on *DS1* is not as good as the performance on *DS2*, our model still performs an important role considering the requirement of selecting a single correct task from 30 potential candidates on average. Even in this case, we still manage to outperform the HMM result by 20% and also outperform the All fifth Markov model result by 3%.

4.3. *Sequential KNN experiments*

4.3.1. Data preparation We carried out a number of experiments based on records from two real processes (*DS3* and *DS4*). As we aim to predict the process outcome, it is necessary to label the outcome as either success or failure, and this needs to be carried out before the proposed method can be applied. There is, however, an issue in our available process data. Records that are labelled with outcomes are not available for many processes. For some processes, it is not always trivial to define process success or failure.

To this end, we have to look for different suitable criteria to classify process instances into success/failure for different data sets. One such criterion is the duration used to complete a process instance. That is, to label the process instance as a success if the time duration is under a chosen threshold, otherwise it is labelled as a failure. In the case of *DS4*, however, 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, then that process instance is classified as success, otherwise it is classified as failure.

4.3.2. Results To evaluate KnsSA, 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 *A*, *B*, *D* and *A*, *A*, *C*, 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. The distance for each category is computed as the difference in the number of occurrences of the corresponding task. Then, the sum over all categories is taken in order to obtain the total distance between any two given sequences. For instance, the two sequences given previously consist of four categories *A*, *B*, *C* and *D*. The distance for 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 $d = d_A + d_B + d_C + d_D = 4$.

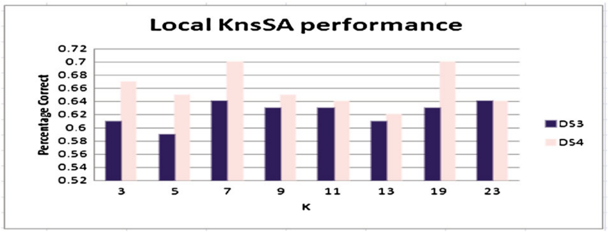

Figure 6: *Percentage of correct predictions of local KnsSA using data sets DS3 and DS4.*

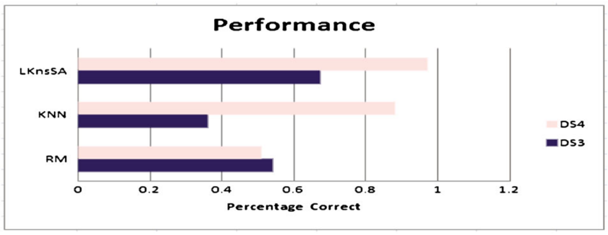

Figure 7: *Percentage of correct predictions of different models on data sets DS3 and DS4.*

For the proposed models, we investigate the effect of *K* as it is important to obtain the reasonable number of similar sequences. As the labels are 0 and 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 *DS3* and *DS4* 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 applied to data sets *DS3* and *DS4* are presented in the Figure 6.

The performances of the original KNN, random guess and the proposed models, local KnsSAs, applied to the two data sets are presented in Figure 7.

The results show that the proposed model outperforms both benchmarking models original KNN and random guess. This also implies that the temporal characteristics of the data are important for predicting the process outcome.

5. Conclusions

In this paper, we have introduced several models for predicting the next step in business processes in a telecommunication services context where no formal description of the processes exists or the real process significantly deviates from the designed path prototype when being applied in real environments and under realistic constraints. Specifically, we first applied Markov and hidden Markov models to generate probability matrices for moving from one event to the next and then using them to provide predictions. Next, we have proposed a novel predictive model (MSA) that is an extension of Markov models employing sequence alignment. In order to analyse its effectiveness, we have empirically evaluated MSA against two data sets from a major telecommunication company with varying degrees of diversification. The results have clearly demonstrated that MSAs outperform both Markov models and HMMs by at least 10%. In addition, we have also shown that higher order Markov models outperform a first order Markov model. The default prediction module we introduced significantly improves the corresponding original Markov model result when the order of the model is high (starting from the fifth order Markov model). Nonetheless, although the improvement provided by the new default prediction module is quite significant, higher order MSAs do not outperform second order MSA in the case of the data set *DS1* and third order MSA in the case of the data set *DS2*. Another contribution of the paper is that we propose a new sequential KNN for symbolic sequences. This proposed method outperforms the benchmarking models and proves that sequential characteristics of the data play an important role in predicting the process outcome. These experimental results motivate us to use sequence alignment technique as a similarity measure to group sequences that have similar characteristics and then tackle them separately. Our future research will look into using sequence alignment and *K*-means clustering to cluster data into *K* groups and then treat each group with suitable approaches.

References

VAN DER AALST, W. (2011) Process Mining: Discovery, Conformance and Enhancement of Business Processes, New York Icn: Springer-Verlag.

BAUM, L.E., T. PETRIE, G. SOULES and N. WEISS (1970) A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *The Annals of Mathematical Statistics*, **41**, 164–171.

BERRY, M.J.A. and G.S. LINOFF (2004) Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management, New York: Wiley.

COX, L.A. JR. and D.A. POPKEN (2002) A hybrid system-identification method for forecasting telecommunications product demands, *International Journal of Forecasting*, **18**, 647–671.

DESHPANDE, M. and G. KARYPIS (2004) Selective Markov models for predicting web page accesses, *ACM Transactions on Internet Technology (TOIT)*, **4**, 163–184.

EASTWOOD, M. and B. GABRYS (2009) A non-sequential representation of sequential data for churn prediction, in *Proceedings of the KES2009 Conference*, Santiago, Chile.

EIRINAKI, M., M. VAZIRGIANNIS and D. KAPOGIANNIS (2005) Web path recommendations based on page ranking and Markov models, *WIDM*'05, 2–9.

NEEDLEMAN, S.D. and C.D. WUNSCH (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins, *Journal of Molecular Biology*, **48**, 443–453.

NETZER, O., J. LATTIN and V.S. SRINIVASAN (2007) A hidden Markov model of customer relationship dynamics, *Marketing Science*, **27**, 185–204.

PAPOULIS, A. and S.U. PILLAI (1991) Probability, Random Variables and Stochastic Processes, New York: McGraw-Hill.

PITKOW, J. and P. PIROLLI (1999) Mining longest repeating subsequences to predict world wide web surfing, in *The 2nd USENIX Symposium on Internet Technologies & Systems*, Colorado, USA, 139–150.

RUTA, D. and B. MAJEED (2011) Business process forecasting in telecommunication industry, in *GCC Conference and Exhibition (GCC), 2011 IEEE*, Dubai, United Arab Emirates, 389–392.

RUTA, D., D. NAUCK and B. AZVINE (2006) K nearest sequence method and its application to churn prediction, in Intelligent Data Engineering and Automated Learning IDEAL 2006, volume **4224** of Lecture Notes in Computer Science, Corchado, E., Yin, H., Botti, V. and Fyfe, C. (eds). Berlin: Springer, 207–215.

SMITH, T.F. and M.S. WATERMAN (1981) Identification of common molecular subsequences, *Journal of Molecular Biology*, **147**, 195–197.

TAYLOR, P., M. LEIDA and B.A. MAJEED (2012) Case study in process mining in a multinational enterprise, in Lecture Notes in Business Information Processing, Berlin: Springer.

THEDE, S. and M. HAPPER (1999) A second-order hidden Markov model for part-of-speech tagging, in *The 37th Annual Meeting of the ACL*, Maryland, USA, 175–182.

WATERMAN, M. (1994) Estimating statistical significance of sequence alignments, *Philosophical Transactions of the Royal Society of London, Series B: Biological Sciences*, **344**, 383–390.

The authors

Mai Le

Mai Le is currently a research assistant at Bristol University. She is working on a collaborative project between Bristol University and BT's Research and Innovation Division. Her research focuses on data analytics and visual analytics that enable autonomous systems to make decisions in unusual situations based on human judgment and on data from previous similar situations. Mai obtained her MSc in optimisation and modelling from the University of Paris 6, in 2007. She is completing her PhD at Bournemouth University in mathematical modelling and predictive analysis of temporal sequential data.

Bogdan Gabrys

Bogdan Gabrys received an MSc in electronics and telecommunication from Silesian Technical University, Poland, in 1994, and a PhD in computer science from Nottingham Trent University, UK, in 1998. After many years of working at different Universities, he moved to the Bournemouth University in UK in January 2003, where he holds a Chair in Computational Intelligence position and acts as the founding director of an interdisciplinary Data Science Institute. His current research interests lay in a broad area of intelligent, complex adaptive systems and include a wide range of machine learning, biologically/nature inspired learning and hybrid intelligent techniques encompassing data and information fusion, learning and adaptation methods, multiple classifier and prediction systems, processing and modelling of uncertainty in predictive modelling, pattern recognition, diagnostic analysis and decision support systems.

Detlef Nauck

Detlef Nauck is chief research scientist for Data Science with BT's Research and Innovation Division located at Adastral Park, Ipswich, UK. He is leading a group of international scientists working on Intelligent Data Analysis and Autonomic Systems. Detlef has been working on data analytics projects across a number of areas such as customer service analytics, real-time business intelligence, business process analytics and network analytics. His current work focusses on introducing autonomic principles like self-learning, self-healing and self-optimisation into business processes and network management. Detlef is a visiting professor at Bournemouth University and a private docent at the Otto-von-Guericke University of Magdeburg, Germany. Detlef holds an MSc (1990) and a PhD (1994) in computer science both from the University of Braunschweig, Germany. He also holds a Habilitation (postdoctoral degree) in computer science from the Otto-von-Guericke University of Magdeburg, Germany (2000).