DOI: 10.17323/2587-814X.2024.3.41.55

Hidden Markov model: Method for building a business process model*

Artem Yu. Varnukhov
E-mail: varnuhov_ayu@usue.ru
Ural State University of Economics, Ekaterinburg, Russia

Abstract

More and more companies are influenced by the rapid development of technology (Industry 4.0/5.0 concept), are embracing digital transformation processes. The introduction of information systems makes it possible to accumulate a large amount of data about the company's activities. Study of such information expands the opportunities for applying a data-driven approach to business process management (BPM). Processing and studying data from event logs using process mining methods make it possible to build digital models of business processes which turn out to be a useful source of information when carrying out analysis, modeling and reengineering within the framework of the process approach. In this paper, we develop a method for building a business process model based on a hidden Markov model, taking into account the restrictions imposed by the subject area. The use of a hidden Markov model allows us to use the apparatus of probability theory and mathematical statistics to analyze business processes, as well as to solve classification and clustering problems. This article describes the capabilities of a data-driven approach to business process management and demonstrates examples of the practical application of the method to solve business challenges: drawing a dependency graph that can be used to identify discrepancies between actual and expected execution, as well as a method for predicting the outcome of a business process based on the sequence of observed events.

* The article is published with the support of the HSE University Partnership Programme

Keywords: business processes, hidden Markov models, process mining, business analysis, prediction, classification, data-driven approach, information systems, event logs

Citation: Varnukhov A.Yu. (2024) Hidden Markov model: Method for building a business process model. *Business Informatics,* vol. 18, no. 3, pp. 41–55. DOI: 10.17323/2587-814X.2024.3.41.55

Introduction

The development of the capabilities of modern information technologies stimulates enterprises in various fields to transfer their business processes from "analog" to digital form. There are many methodologies and techniques that enable us to model, reengineer, manage and monitor business processes [1], and they are constantly being improved. Quite often, modeling is performed "manually" with the involvement of appropriate business analysts and "in-house" experts who have specialized knowledge and expertise with respect to the phenomena being modeled. At the same time, in practice, the process of modeling and reengineering business processes turns out to be a non-trivial task even for experienced specialists [2]. For example, distortions occur due to the human factor, one's own position in the organization's structure and other typical issues characteristic of this modeling approach: idealization, choosing the wrong level of abstraction, or the inability to adequately reproduce the observed interaction [3]. As a result, the generated model may reflect only part of the existing "reality" and it is not functional enough so that ultimately it will have very limited value.

The implementation of automated information systems of various classes and functionality (ERP, CRM, ECM, etc.) leads to the concomitant accumulation of a large amount of useful information about the activities of the enterprise in a data warehouse [4]. Processing and subsequent analysis of data accumulated in the enterprise information systems make it possible to use a data-driven approach. Currently, research is being conducted in the field of process data mining [5, 6], creating digital twins [7], predictive and prescriptive analytics [8, 9], robotic process automation [10], and work is also being carried out on practical implementation in various industries [11, 12].

The purpose of this study is to develop a method for building a business process model based on a hidden Markov model, taking into account the characteristics of the subject area.

1. Application of a data-driven approach to business process management

The process approach allows us to present a company as a set of interconnected business processes, each process is considered as a valuable asset that ensures the delivery of the company's products and services to end consumers. The business process management (BPM) methodology defines the management life cycle, which typically consists of the following main stages: analysis, modeling, execution, monitoring, optimization and reengineering. To formulate the capabilities and context of using a data-driven approach within BPM, we can build a generalized management life cycle as shown in *Fig. 1*.

As can be seen from *Fig. 1*, process models are a source of information for analysis and optimization. They help us at the stage of information systems implementation and contribute to the management and control functions. Data accumulated in information systems can be used to make digital models that will allow for a better understanding of the organization's existing business processes. The use of a data-driven approach provides a capability to establish a

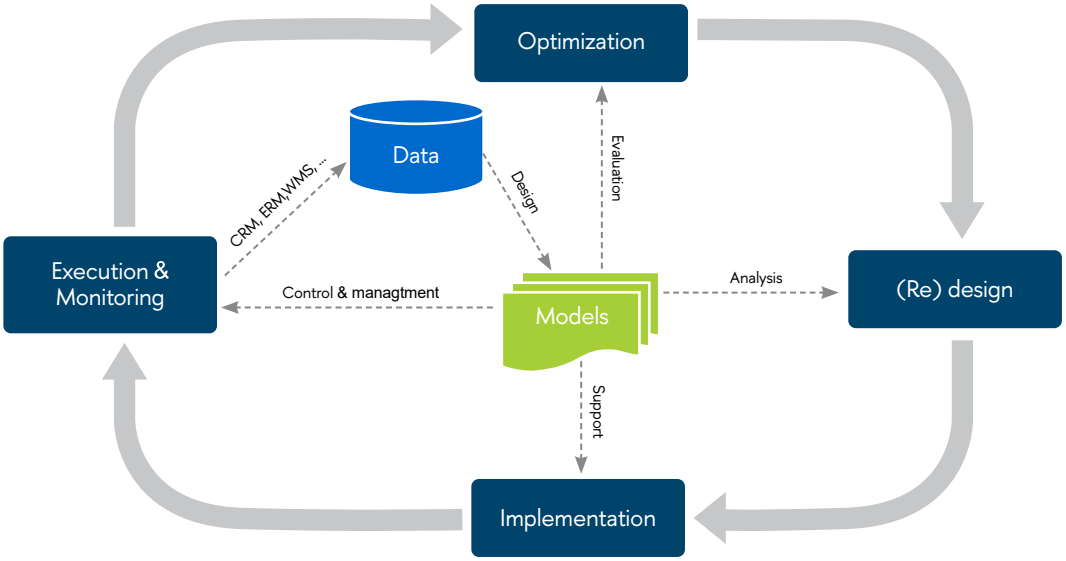

Fig. 1. Generalized BPM life cycle.

close relationship between existing processes and their representation in the form of models. We can single out a few key applications among many options. They include model identification, compliance testing, performance evaluation and process improvement. To identify models, data from event logs of information systems are examined and, using special methods of intelligent analysis, models are built without involving any a priori information. We can use the AS-IS models obtained in this way in further work when carrying out analysis, modeling and reengineering. It is worth noting that the design stage is crucial, since it provides a kind of "entry" point for all other tasks employing the identified digital models. To check compliance, a previously designed reference model of a business process and data from the event log are used: they are compared with each other to determine the degree of compliance. Such verification is useful for monitoring compliance with imposed rules and restrictions, detecting discrepancies between actual and expected performance, finding reasons for deviations, and so on. If we take into account the presence of a time component in the data, then using models it is possible to measure business process performance indicators, detect bottlenecks, evaluate the level of service, etc. For example, variant analysis will reveal differences in control flow and performance indicators between different organizations' departments. In addition to studying the control flow, we can expand the model by including an organizational component. This will allow us to take into account information about the process participants and their relationships. Thus, the use of a data-driven approach allows us to improve the quality and efficiency of business process management.

2. Statement of the problem for building a business process model

If a business process can be represented as a model, then a specific individual case implemented within this model can be described as its instance. An instance of a business process must be characterized by a certain set of sequential or parallel actions (activities) with the capability to determine the order in which they occur. Different instances shall be distinguishable from each other at least by the order of events. *Table 1* shows an example of an event log fragment obtained from the information system.

Table 1.

Event log fragment

Instance ID	Event ID	Timestamp		Event	Employee	…
1001	24837	24.08.2023	13:20	Receipt of request	Ivanov A.	…
1001	25123	25.08.2023	11:05	Availability check	Petrova I.	…
1001	26001	26.08.2023	09:15	Sending an invoice	Ivanov A.	…
1001	26560	27.08.2023	16:07	Shipment of goods	Sidorov V.	…
1002	24842	24.08.2023	14:27	Receipt of request	Ivanov A.	…
1002	24859	24.08.2023	16:20	Availability check	Petrova I.	…
1002	24892	24.08.2023	17:40	Refusal to deliver	Sobolev B.	…
…	…	…		…	…	…

It is assumed that all data in the log relates to one analyzed business process. Each line in the table contains the following mandatory attributes: Instance ID, Event and Timestamp. Multiple lines with the same Instance ID attribute value represent events that are associated with a single business process instance. The Event attribute contains the name of the event, which can be associated with some action (activity). The Timestamp attribute is used to chronologically arrange events within a single instance. The event log may contain other additional attributes (Employee, Cost, Customer, Office, etc.), which can be useful for monitoring a business process using machine learning [13]. For brevity, we will use a multiset, which will consist of chronologically ordered and grouped sequences of events according to the business process log. For example, for the data given in *Table 1*, we can record a multiset:

$$L = \{< a, b, c, d >^n, < a, b, e >^m, ...\}, \quad (1)$$

where L is a multiset in which each element contains an ordered sequence of events: a – "Receipt of request" event, b – "Availability check" event, c – "Sending an invoice" event, d – "Shipment of goods" event, e – "Refusal to deliver"; n and m are the number of times this ordered sequence occurred in the log.

Thus, we need to develop a method for building a business process model based on incoming input data in the form of multiset L.

3. Analysis of methods for building a business process model

"α-algorithm". It is quite simple and one of the first methods that allows us to recreate a business process model from an existing set of sequential events in the form of a Workflow-net (an individual case of a Petri net) [14]. To do this, the algorithm scans the log searching for a specific set of patterns: sequence, XOR-split and AND-split. Based on this, a matrix of "fingerprints" is recorded, enabling recognition of the existing relationships between events. The final model is built according to this matrix, taking into account the inference rules. We can name the following limi-

tations of the "α-algorithm": difficulties in processing noisy data, inability to recognize 1- or 2-step cycles and problems with local dependencies.

"Heuristic Miner". Unlike the "α-algorithm", it applies the idea of counting the frequencies of occurrence of events and reproduces the process model in the form of a Causal net [15, 16]. First, metrics are calculated that estimate the number of direct connections between each pair of events and measure the degree of their dependence. A dependency graph is drawn using (sequence, XOR, AND, and cycle) patterns based on the calculated metrics. The search for mergers and splits in the dependency graph can be performed by a sliding window over an event log of a given size or based on solving an optimization task in which the target function is the compliance degree of the model to the analyzed event log. The resulting process model in the form of Causal net can be converted to other required notations (BPMN, UML, EPC, WF-net, etc.). This method is less susceptible to noise in the data and eliminates many of the shortcomings of the "α-algorithm", but has problems handling non-local dependencies and detecting duplicate events, and also requires manual adjustment of cutoff threshold levels.

"Region-Based Miner". It is based on the application of the theory of regions and is built according to the assumption that state models can be transformed into Petri nets [17]. There are several approaches to implement this method. The first approach consists in defining a region as a set of states such that the actions in the state and transition model are consistent with this region. In this case, all events can be divided into "incoming", "outgoing" and "internal" in relation to this region. After dividing the regions according to these rules, each region can be associated with a specific position in the Petri net. The second approach uses a specially defined language model instead of a system of states and transitions [18]. The main idea of this approach is that removing a P_i position does not remove any behavioral pattern, but adding a new position may lead to the elimination of some possible behavior options. Advantages include the capability to handle more complex control flow structures. The weakness of this method is the inability to detect some types of process designs, issues with accuracy and generalization ability, and the difficulty of its practical implementation.

"Inductive Miner". It consists of three recursive steps: drawing an oriented graph, searching for a cut, and splitting log entries [19]. The method uses a pre-processed event log as input data. In the first step, the method transforms the data into an oriented graph in which each node corresponds to one event, and the arcs form transitions between events. After this, an attempt is made to detect places of possible cuts. If places of such cuts are detected, then the algorithm generates a cut operator and split segments. Based on the detected segments, the log is decomposed into smaller components. Each fragment thus obtained is then processed recursively until the base case is found: the fragment contains only one event. If, in the process of recursive descent, a fragment is encountered that is not reducible to the base case and at the same time does not have valid places for cutting, then the process of "falling through" is applied. The basic implementation of the method had difficulties detecting fixed-length cycles, handling rare events and limitations associated with the recursive nature of the design. However, further development of the method made it possible to overcome the primary disadvantages and provided the capability of scaling and using distributed computing [20].

The methods presented above allow us to build a business process model in various ways, however, it is of interest to study the capability of building a model that is based on probability assessment. As a basis, we could consider a Markov chain, but given the characteristics and nature of the source data, it would be more acceptable to assume that the events recorded in the log are only the external manifestation of some process hidden from the observer. To model such an assumption, we can consider a first-order hidden Markov model (hereinafter referred to as HMM). It is known that such models with hidden states are successfully applied for text processing tasks in natural languages [21], gesture identification [22], speech recognition [23], bioinformatics [24] and other fields. Based on the information presented in the analyzed sources, it can

be expected that the HMM use for business process analysis will make it possible not only to build a business process model, but also to solve the problem of classification and perform data clustering.

4. Suggested method for building a model

The HMM possesses multiple hidden states $S = \{s_1, s_2, s_3, ..., s_N\}$. Each hidden state can be associated with some other hidden states. A schematic representation of the model is shown in *Fig. 2*.

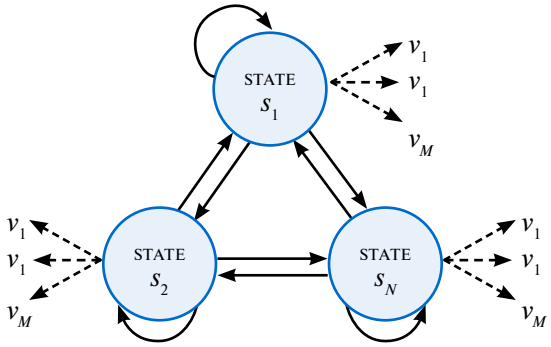

Fig. 2. The HMM schematic representation.

This paper considers a fully connected HMM structure, in which each hidden state s_k is associated with all hidden states different from it, as well as with itself. In addition to the hidden states, a finite alphabet of multiple observed events $V = \{v_1, v_2, v_3, ..., v_M\}$ is defined and each hidden state reproduces events from a given set V. At any individual time t, the model is in one of the hidden states:

$$\forall t : q_t \in S, 1 \le t \le T. \quad (2)$$

The HMM makes transitions between hidden states. So at time t, being in the hidden state q_t, the model will move to another state with a certain probability and at time $t + 1$ will be in the hidden state $q_{t+1} \in S$. In this paper we consider only discrete instants of time, while the current state and the chain of completed transitions between them are invisible to the observer. Being in some hidden state q_t, the model reproduces the event $o_t \in V$, which is visible to an external observer. The series of transitions between states and the events they reproduce as a result generates a sequence of observations $O = \{o_1, o_2, o_3, ..., o_T\}$. A schematic representation of the HMM operation is shown in *Fig. 3*.

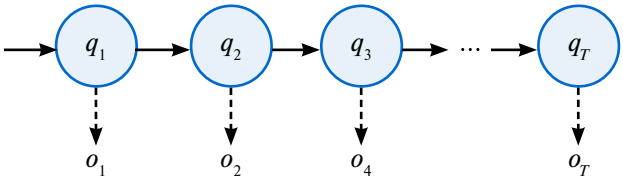

Fig. 3. Schematic representation of the HMM operation.

Since this work considers a first-order HMM, according to the Markov property we will assume that the probability of transition from one state to another is determined only by the previous state of the model:

$$P(q_t = s_i | q_1 = s_n, q_2 = s_m, \ldots, q_{t-1} = s_j) = \\ = P(q_t = s_i | q_{t-1} = s_j). \quad (3)$$

The second assumption will be that the probability of producing the observed event o_i depends only on the state in which the model is at a discrete instant of time t and does not depend on other states and observed events:

$$P\left(o_t = v_k | q_1 = s_n, \ldots, q_t = s_i, o_1 = v_m \ldots, o_{t-1} = v_j\right) = \\ = P\left(o_t = v_k | q_t = s_i\right). \quad (4)$$

Let us determine the initial distribution over the hidden states of the model, which specifies the probability that the model will be in a certain state at the first step:

$$\pi = \{\pi_i\}_{i=1}^N, \ \pi_i = P(q_1 = s_i), \sum_{i=1}^N \pi_i = 1. \quad (5)$$

Let us define the distribution of transition probabilities between hidden states as matrix $A = (a_{ij})$, where

$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i), 1 \le i, j \le N, \sum_{j=1}^N a_{ij} = 1. \quad (6)$$

We define the probability distribution of events occurring when the model is in some hidden state as matrix $B = (b_{ik})$, where

$$b_{ik} = P(o_t = v_k | q_t = s_i),\ 1 \le i \le N,\ 1 \le k \le M,$$
$$\sum_{k=1}^{M} b_{ik} = 1. \quad (7)$$

Based on the above, we define the hidden Markov model θ as

$$\theta = (S, V, A, B, \pi). \quad (8)$$

Let us assume that the original log is presented and some data pre-processing has been completed. Suppose multiset is

$$L = \{< a,e >^5,\ < a,b,c,e >^5, < a,c,b,e >^5,$$
$$< a,d,e >^{10}, < a,d,d,e >^5, < a,d,d,d,e >^1,$$
$$< a,b,c,d,e >^3, < a,b,d,c,e >^3, < a,c,b,d,e >^3,$$
$$< a,c,d,b,e >^3, < a,d,c,b,e >^3, < a,d,b,c,e >^3,$$
$$< e,b,c,m >^4, < e,c,b,m >^3\}. \quad (9)$$

Multiset L contains elements repeated several times, which represent individual instances of a business process performed at different time. It can be observed that some elements of the multiset (for example, <*a, b, c, e*> and <*a, c, b, e*>) contain almost identical sequences of events, except that the order in which events "*b*" and "*c*" follow is rearranged. An apparent rearrangement in the multiset may occur due to the fact that the registered events appear in the source log ordered by time stamp, but in reality, they represent subprocesses of a business process running in parallel. An example of such a situation is shown in *Fig. 4*.

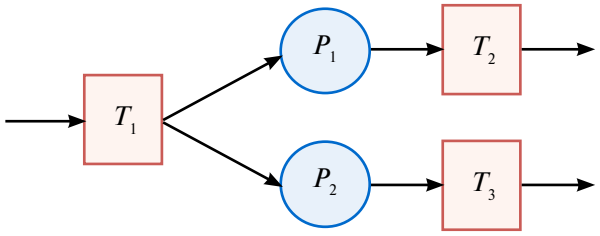

Fig. 4. Subprocesses run in parallel.

Due to the fact that such events are always pairwise and follow each other (albeit in a different order), and also taking into account the nature of the modeled subject area, we will assume that such rearranged events generate a logically single operation of the business process. Thus, we will refer similar pairwise permutations of events, which by their behavior form the logical AND operator of a business process, to the same hidden state of the model. Generally, a business process has one fixed start event at which its execution begins. A business process may have several end events due to the need to present different outcomes. Each such end event is logically final and therefore should not be divisible into several subprocesses. If subprocesses that run in parallel are not in the start or end events, then it should be assumed that there is some event after which the execution branches, as well as an event connecting the parallel execution. Thus, the logical AND operator of a business process shall be located between some start and end event in the observed sequence of events. To determine groups of events that form a set of logical operations AND of a business process, we choose unique elements according to rule (10) from multiset L:

$$FSET = \left\{ \begin{array}{l} \left(\sigma(i) = S_1, \sigma(i+1) = S_2, \sigma(i+2) = \right. \\ \left. = S_3, \sigma(i+3) = S_4\right): \\ \exists\left(\delta(l) = S_1, \delta(l+1) = S_3, \delta(l+2) = \right. \\ \left. = S_2, \delta(l+3) = S_4\right) \end{array} \right\}$$
$$\forall \sigma, \delta \in L : |\sigma| \ge 4,\ |\delta| \ge 4,$$
$$1 \le i \le |\sigma|,\ 1 \le l \le |\delta|. \quad (10)$$

That is, the members of the *FSET* set are ordered sequences of events, each of which satisfies the following conditions:

- is obtained from elements included in original multiset L;
- the sequence contains at least four consecutive events;
- there is another quadruplet with the same start event S_1 and end event S_4 in which places of events S_2 and S_3 are interchanged.

For the considered multiset from (9), set *FSET* will be generated consisting of the following elements:

$$FSET(L) = \{(a,b,c,e),(a,c,b,e),(a,b,c,d),(b,c,d,e),(a,b,d,c),(b,d,c,e),(a,c,b,d),(c,b,d,e),(a,c,d,b),(c,d,b,e),(a,d,c,b),(d,c,b,e),(a,d,b,c),(d,b,c,e),(e,b,c,m),(e,c,b,m)\}. \quad (11)$$

Each element of the set *FSET* represents the minimum acceptable part of a possible permutation. So, for example, to form a logical AND operator, composed of two subprocesses "*b*" and "*c*" run in parallel, which starts after the event "*a*" and ends with the event "*e*", it is necessary that the set *FSET* contains both parts of such a permutation (*a, b, c, e*) and (*a, c, b, e*). The elements of the set (*a, b, d, c*) and (*a, d, b, c*) also generate the minimal logical AND operator. However, it can be seen from (9) that the elements "*b*" and "*d*" are part of a larger logical AND operator, which also includes the event "*c*". Thus, it is necessary to define a procedure for growing longer logical AND operators, consisting of basic minimal parts. For this purpose, we define sets of start, permutable and end events:

$$FS = \{\sigma(1) : \sigma \in FSET\}. \quad (12)$$

$$PS = \{\sigma(2) : \sigma \in FSET\}. \quad (13)$$

$$ES = \{\sigma(4) : \sigma \in FSET\} \setminus PS. \quad (14)$$

To build up the maximum possible permutation, we take each start event from the set *FS* one by one and, going through the set of *FSET* elements, we will add each permutable symbol encountered in the second and third positions until we reach the element that contains the end event from *ES* in the last position. The procedure for display augmentation is shown in *Listing 1*.

As a result of the augmentation procedure, a multiset will be generated that contains elements with selected start and end states, as well as a set of permutable events between them. For example, a multiset will be formed from (9):

$$FPRM(L) = \{<e,\{b,c\},m>^2,<a,\{b,c\},e>^2,<a,\{b,c,d\},e>^6,<b,\{c,d\},e>^2,<c,\{b,d\},e>^2,<d,\{b,c\},e>^2\}. \quad (15)$$

The multiplicity of elements in multiset *FPERM* reflects the frequency with which a given permutation occurred in the source data. Let us assume that the source data contained a group of events "*b*", "*c*", "*d*", which forms the logical AND operator of the business process. Then the multiplicity of such a group with the same start and end events must be equal to six. Suppose that this group is missing one element, for example (*a, c, b, d, e*). In this case, the group of events should split into two logical AND operators forming parallel sub-

Input: *FS*: Start events
Input: *ES*: End events
Input: *FSET*: Set of data
Output: *FPERM*: Multiset of augmented permutations

```
1.  FPERM ← ∅
2.  for each start event SS in FS do
3.  ACCPERM ← ∅
4.  for each element σ in FSET do
5.      if σ(1) = SS or |ACCPERM| > 0 then
6.          ACCPERM ← ACCPERM ∪ σ(2) ∪ σ(3)
7.      if |ACCPERM| > 0 and σ(4) ∈ ES then
8.          FPERM ← FPERM ⊎ (SS, ACCPERM, σ(4))
9.          ACCPERM ← ∅
```

Listing 1. Augmentation procedure.

processes: the first group of events starts with event "*b*" and includes "*c*" and "*d*", the second group starts with event "*d*" and includes "*c*" and "*b*". It is also possible that some data may be lost during uploading or preprocessing. To take these situations into account, we introduce a control metric:

$$\text{MinLimit}(\sigma) = \left|\sigma(2)\right|! - \varepsilon,\ \sigma \in FPERM,\ \varepsilon \geq 0. \quad (16)$$

Metric (16) allows us to calculate the required number of identical elements, taking into account the capability of adjusting ε for missing or lost data. Let us create set of unique groups of events that form logical AND operators by including only elements whose multiplicity is not less than a given limit:

$$LPERM = \left\{\sigma : \sum_{s \in FPERM} \left[s = \sigma\right] \geq \text{MinLimit}(\sigma)\right\},$$
$$\sigma \in FPERM. \quad (17)$$

In addition to the augmented elements, the resulting set also contains parts of a larger permutation. To exclude such extra elements, we shall perform the check:

$$ANDGROUP = \left\{\sigma : \nexists \delta \,|\delta(2) = \right.$$
$$\left. = \left\{\sigma(1)\right\} \cup \sigma(2) \wedge \delta(3) = \sigma(3)\right\},\ \sigma, \delta \in LPERM. \quad (18)$$

Thus, the set *ANDGROUP* will contain only the necessary groups of events that represent subprocesses of the business process run in parallel. For example, we obtain the following logical AND operators from (9):

$$ANDGROUP(L) =$$
$$= \left\{(e,\{b,c\},m),\ (a,\{b,c\},e),\ (a,\ \{b,c,d\},e)\right\}. \quad (19)$$

From (8) it follows that to determine the HMM, it is necessary to specify sets of hidden states and events, as well as to determine transition and emission matrices, and probability vector characterizing the choice of the start state. For an arbitrary business process, none of these model parameters are known in advance, since there is only an observable sequence of events that is obtained from the input data. Thus, it is necessary to formulate a method that allows us to find parameters if only the sequence of observed events of a business process is known. This problem was described by Rabiner and is one of the three basic problems when working with HMM and at the same time the most challenging among them [25]. The complexity is conditioned by the lack of known analytical methods for solving the task enabling us to determine the model parameters for any finite sequence *O*. There are several approaches to solving it by reducing the problem to an optimization task for finding such model parameters θ that allow maximizing the probability $P(O \mid \theta)$. One such approach is the Baum–Welch algorithm, which is a type of EM (expectation-maximization) algorithm for computing maximum likelihood estimates. In general, this algorithm consists of two steps (E-step and M-step), which make it possible to iteratively recalculate the parameters θ and successively approach the locally maximum estimate at a certain *O*.

However, the classic implementation of the Baum–Welch algorithm does not take into account the features of the subject area and the specifics of the business processes functioning. Therefore, this paper proposes an improved modification of the algorithm for application to the studied task.

Let us define the set of the observed events *V* of the model as equal to the set of unique business process events from multiset *L*:

$$V = \bigcup_{\sigma \in L} \left\{\sigma(i),\ 1 \leq i \leq \left|\sigma\right|\right\}. \quad (20)$$

Since each element in (18) is a logical AND operator and any unique group of permutable events must be assigned to one hidden state of the model, we define a set of hidden states:

$$SPU = \left\{\sigma(2) :\ \sigma \in ANDGROUP\right\}. \quad (21)$$

$$SOU = V \setminus \bigcup \delta,\ \forall \delta \in SPU. \quad (22)$$

$$S = \left\{i : 1 \leq i \leq \left|SOU\right| + \left|SPU\right|\right\}. \quad (23)$$

The iterative nature of the implementation of the classical Baum–Welch algorithm allows matrices *A* and *B* to be specified with arbitrary values before starting its operation, since in the process of updating the param-

eters convergence to optimal values will be achieved. However, it is known that the HMM various organizational structures (ergodic, left-to-right, parallel left-to-right, etc.) can influence the nature of its behavior and the obtained outcome. Let each event from set V be consecutively numbered with natural number v_k from 1 to $|V|$. Let us define matrices A and B as follows:

$$a_{ij} = \frac{1}{|S|},\ 1 \le i, j \le |S|. \tag{24}$$

$$b_{ik} = \begin{cases} 1 : \forall i \in \left\{1 \le i \le |SOU|\right\} \wedge \forall k = \\ = v_k \in SOU \,|\forall i \ne j\ b_j \ne b_i \\ \dfrac{1}{|\sigma|} : \forall i \in \left\{|SOU| < i \le |S|\right\} \wedge \forall k = \\ = v_k \in \sigma \,|\forall i \ne j\ b_j \ne b_i, \forall \sigma \in SPU \\ 0 : \text{else}. \end{cases} \tag{25}$$

To set the initial distribution, we will assume that the start event of the business process is unique and for this event it is defined that $v_k = 1$:

$$\pi_i = \begin{cases} 1 : i = v_k \\ 0 : i \ne v_k. \end{cases} \tag{26}$$

If there are elements in multiset L that contain different start events, then it is always possible to add a new surrogate event to the beginning of all elements of the multiset in order to move to the unicity of the start event.

To reduce the number of operations and simplify calculations, the forward and backward pass method is used, which is based on the principles of dynamic programming. In this case, a matrix of intermediate values is formed, which makes it possible to estimate the probability at each step by summing up the calculations performed in the previous steps using auxiliary functions:

$$\alpha_t(i) = P(o_1, o_2, o_3, \ldots, o_t, q_t = s_i | \theta). \tag{27}$$

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, o_{t+3}, \ldots, o_T | q_t = s_i, \theta). \tag{28}$$

The classical implementation (27) and (28) does not take into account the model features associated with the specifics of the problem being studied. It is necessary to take into account the restrictions imposed on transitions between hidden states associated with logical AND operator. To do this, we define auxiliary functions as follows:

$$\varphi(i) = \{v_k \,|b_{ik} > 0,\ 1 \le k \le |V|\}. \tag{29}$$

$$HFW_t(i) = \begin{cases} |\varphi(i)| \times \prod_{j=0}^{|\varphi(i)|-1} \left[o_{t+j} \in \varphi(i)\right] : T > t + |\varphi(i)| \\ 0 : T \le t + |\varphi(i)|. \end{cases} \tag{30}$$

Function (30) specifies the estimate that will be used to select the most appropriate hidden state when calculating α and β. For such a state, the value of function (30) will be maximum:

$$HFW_t^{\max} = \underset{1 \le i \le N}{\operatorname{argmax}}\, HFW_t(i). \tag{31}$$

Then we will obtain:

$$\alpha_1(i) = \pi_i b_i(o_1). \tag{32}$$

$$\alpha_{t+1}(j) = \begin{cases} b_j\left(o_{t+1}\right) \sum_{i=1}^{N} \alpha_t(i) a_{ij} : \Big(j = HFW_{t+1}^{\max} \wedge \\ \wedge o_{t+1} \in PS \wedge o_t \notin PS \Big) \vee (o_{t+1} \notin PS) \\ b_j\left(o_{t+1}\right) \alpha_t(j) a_{jj} : o_{t+1} \in PS. \end{cases} \tag{33}$$

This definition (33) enables us to limit the transitions space between hidden states of the model, which contain a group of events that form a logical AND operator. In a similar way we define:

$$HBW_t(i) = \begin{cases} |\varphi(i)| \times \prod_{j=0}^{|\varphi(i)|-1} \left[o_{t-j} \in \varphi(i)\right] : t - |\varphi(i)| \ge 0 \\ 0 : t - |\varphi(i)| < 0 \end{cases} \tag{34}$$

$$HBW_t^{\max} = \underset{1 \le i \le N}{\operatorname{argmax}}\, HBW_t(i). \tag{35}$$

$$\beta_T(i) = 1. \tag{36}$$

$$\beta_t(i)=\begin{cases}\sum_{j=1}^{N}\beta_{t+1}(j)a_{ij}b_j(o_{t+1}):\left(i=HBW_t^{\max}\wedge\right.\\ \left.\wedge o_t\in PS\wedge o_{t+1}\notin PS\right)\vee(o_t\notin PS)\\ \beta_{t+1}(i)\,b_i(o_{t+1})\,a_{ii}:o_t\in PS\\ 0:o_t\in PS\wedge o_{t+1}\notin PS\wedge i\neq HBW_t^{\max}.\end{cases}\quad(37)$$

Since for a business process the end event means its final completion and the impossibility of transition to any other states, as well as for compliance with (7), we will assume that such hidden states should transform into themselves forming a loop. Let us define ξ and γ as follows:

$$\xi_t(i,j)=\begin{cases}\dfrac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}:1\le t\le T-1\\ 1:t=T\wedge i=j\wedge\alpha_{T-1}(i)>0.\end{cases}\quad(38)$$

$$\gamma_t(i)=\frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}.\quad(39)$$

Considering that updating the coefficients shall be performed across all elements of multiset L, we will use:

$$a_{ij}^{*}=\frac{\sum_{\sigma\in L}\sum_{t=1}^{T}\xi_t(i,j)}{\sum_{\sigma\in L}\sum_{t=1}^{T}\gamma_t(i)}.\quad(40)$$

$$b_{ik}^{*}=\frac{\sum_{\sigma\in L}\sum_{t=1}^{T}\left[o_t=v_k\right]\gamma_t(i)}{\sum_{\sigma\in L}\sum_{t=1}^{T}\gamma_t(i)}.\quad(41)$$

Iterative execution of E and M steps of the algorithm is carried out up to its convergence or until a specified limit of repetitions is reached.

5. Options for using the method to solve business tasks

5.1. Predicting the outcome of a business process and finding deviations

Let us assume that there are many instances of a business process which are divided according to some parameter into several non-crossing groups G_1, G_2, G_3, ..., G_N. For example, within the framework of the "sale of goods" business process, we can split its instances according to the transaction outcome. In this case, the following groups can be generated: "refused to purchase", "postponed the purchase" and "the purchase is successful". Each such group corresponds to its own multiset L_1, L_2, L_3, ..., L_N. Using the proposed method described above, we will build N hidden Markov models using these multisets as a training sample. As a result, θ_n will correspond to each L_n. Let us use the forward-backward pass algorithm and define (42):

$$\alpha_t(j)=\begin{cases}\pi_j b_j(o_1):t=1\\ b_j(o_t)\sum_{i=1}^{N}\alpha_{t-1}(i)a_{ij}:1<t\le T.\end{cases}\quad(42)$$

For a new instance of the business process O_x, using the built HMMs, we can predict its membership in one of the previously generated groups. The group G_n will be the target group for which the estimate $P(O_x\mid\theta_n)$ is the highest:

$$F(O_x)=\arg\max_{\theta}P(O_x\mid\theta)=\arg\max_{\theta}\sum_{i=1}^{N}\alpha_T(i).\quad(43)$$

As a result, instance O_x is most likely to have an outcome that matches group G_n. Such a prediction can also be obtained for incomplete instances of a business process, that is, for those cases when there is only part of the O_x sequence. Having the capacity to obtain such an evaluation, one can solve various practical tasks. So, for example, for the business process of selling a product, you can analyze transactions that are at some intermediate stage to predict a possible outcome. If a high probability of an undesirable outcome is determined, then corrective actions can be developed for such transactions aimed at correcting the path. In addition, having a reference model of a business process as input and data from the respective event log, we can, having received an evaluation of the membership of each sequence, identify deviant instances for the purpose to further examine the causes and make management decisions.

5.2. Representation of a business process as a dependency graph

Suppose that for a certain process, an event log is recorded and processed, on the basis of which multiset L is formed:

$$L = \{< a,z >^5, < a,e,b,c,z >^5, < a,e,c,b,z >^3, < a,b,c,d,z >^5, < a,b,d,c,z >^5, < a,c,b,d,z >^5, < a,c,d,b,z >^5, < a,d,c,b,z >^5, < a,d,b,c,z >^5, < a,l,z >^{10}, < a,l,l,z >^5, < a,l,l,l,z >^1\}. \quad (44)$$

If we build the HMM for the multiset (44) using the proposed method described above, we obtain the following matrices A and B:

$$A = \begin{pmatrix} 0 & 0 & 0.51 & 0.14 & 0.27 & 0.08 \\ 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.67 & 0 & 0 & 0.33 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.3 & 0\ 7 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & 0.33 & 0.33 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

We assume that hidden states for which $a_{ii} > 0$ and containing several events in matrix B correspond to the logical AND operator, and those having only one event generate a cycle. The dependency graph drawm for the multiset (44) is shown in *Fig. 5*.

When needed, this dependency graph can be converted into other representations: BPMN, Petri nets, Casual Net, etc. The resulting dependency graph can be used to study the actual execution of a business process, conduct a comparative analysis of implementation options between different structural units, search for deviations and identify their causes. If we supplement the model with data from the event log about the execution time of basic operations, we can calculate various performance indicators (processing and idle time, duration and effective time of one cycle, etc.). In addition, event logs may contain information about participants, costs incurred and resources used which will allow the model to be scaled to analyze other aspects of the business process.

Conclusion

The data-driven approach is not an alternative to traditional modeling using analysts and domain experts. However, the use of this approach makes it possible to improve the quality of analysis, modeling, design and reengineering of business processes through the study of actual data that have been accumulated in the enterprise's information systems. Detection of non-obvious connections, as well as the capability of

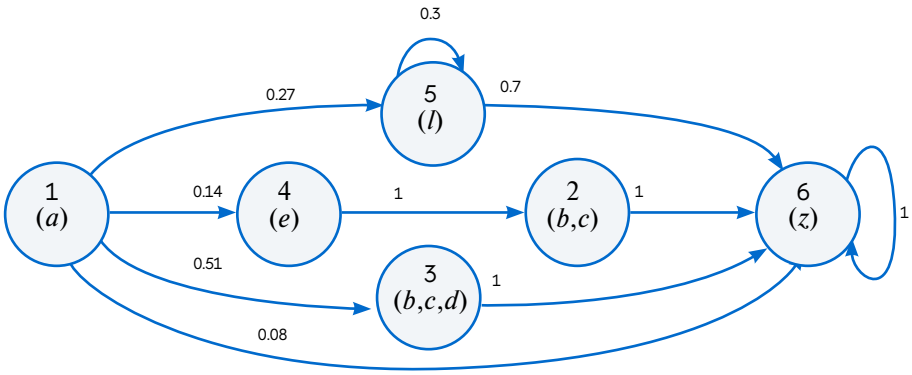

Fig. 5. Business process dependency graph.

impartial analysis, regardless of the value judgment of the process participants, help to minimize the likelihood of distortions and erroneous conclusions. The model so built can be used to monitor the execution of specific instances of a business process, identify deviations or abnormal behavior, and will also provide support for the implementation of key performance indicators (KPIs) in the company's activities, both at the level of individual employees and entire departments.

Unlike other algorithms described in this paper, the proposed method is based on a hidden Markov model, which allows the use of the apparatus of probability theory and mathematical statistics. In particular, a method for obtaining an evaluation of a business process future outcome is demonstrated, which enables implementation of proactive management influence in order to adjust the expected result. In addition, using HMM, you can perform clustering of business process instances and solve the classification task.

The identified shortcomings include: the lack of a guaranteed occurrence of all events that generate the logical AND operator (when using the model as a generator), as well as a narrow horizon for accounting the dependencies (due to the first-order assumption).

As a direction for the development of the method, it is advisable to consider the multi-level hierarchical organization of the model, the introduction of ensemble methods of machine learning and use of higher-order HMMs. ■

References

1. Lizano-Mora H., Palos-Sánchez P.R., Aguayo-Camacho M. (2021) The evolution of business process management: A bibliometric analysis. *IEEE Access*, vol. 9, pp. 51088–51105. https://doi.org/10.1109/ACCESS.2021.3066340

2. Fetais A., Abdella G.M., Al-Khalifa K.N., Hamouda A.M. (2022) Business process re-engineering: A literature review-based analysis of implementation measures. *Information*, vol.13, no. 4, 185. https://doi.org/10.3390/info13040185

3. Rosemann M. (2006) Potential pitfalls of process modeling: part A. *Business Process Management Journal*, vol. 12, no. 2, pp. 249–254. https://doi.org/10.1108/14637150610657567

4. Nambiar A., Mundra D. (2022) An overview of data warehouse and data lake in modern enterprise data management. *Big Data and Cognitive Computing*, vol. 6, no. 4, 132. https://doi.org/10.3390/bdcc6040132

5. Pegoraro M., van der Aalst W.M.P. (2019). Mining uncertain event data in process mining. *2019 International Conference on Process Mining (ICPM)*, pp. 89–96. https://doi.org/10.1109/ICPM.2019.00023

6. Andrews R., van Dun C.G.J., Wynn M.T., Kratsch W., Röglinger M.K.E., ter Hofstede A.H.M. (2020) Quality-informed semi-automated event log generation for process mining. *Decision Support Systems*, vol. 132, 113265. https://doi.org/10.1016/j.dss.2020.113265

7. Park G., van der Aalst W.M.P. (2021) Realizing a digital twin of an organization using action-oriented process mining. *3rd International Conference on Process Mining (ICPM)*, pp. 104–111. https://doi.org/10.1109/ICPM53251.2021.9576846

8. Kratsch W., Manderscheid J., Röglinger M. et al. (2021) Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. *Business & Information Systems Engineering*, vol. 63, pp. 261–276. https://doi.org/10.1007/s12599-020-00645-0

9. Teinemaa I., Dumas M., Rosa M.L., Maggi F.M. (2019) Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 2, pp. 1–57. https://doi.org/10.1145/3301300

10. Leno V., Polyvyanyy A., Dumas M., La Rosa M., Maggi F.M. (2021) Robotic process mining: vision and challenges. *Business & Information Systems Engineering*, vol. 63, pp. 301–314. https://doi.org/10.1007/s12599-020-00641-4

11. Munoz-Gama J., Martin N., Fernandez-Llatas C. et al. (2022) Process mining for healthcare: Characteristics and challenges. *Journal of Biomedical Informatics*, vol. 127, 103994. https://doi.org/10.1016/j.jbi.2022.103994

12. Grisold T., Mendling J., Otto M., vom Brocke J. (2021) Adoption, use and management of process mining in practice. *Business Process Management Journal*, vol. 27, no. 2, pp. 369–387. https://doi.org/10.1108/BPMJ-03-2020-0112

13. Mehdiyev N., Fettke P. (2021) Explainable artificial intelligence for process mining: A general overview and application of a novel local explanation approach for predictive process monitoring. *Interpretable Artificial Intelligence: A Perspective of Granular Computing*, *Springer*, pp. 1–28. https://doi.org/10.1007/978-3-030-64949-4_1

14. van der Aalst W.M.P., Weijters T., Maruster L. (2004) Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 9, pp. 1128–1142. https://doi.org/10.1109/TKDE.2004.47

15. Mannhardt F., de Leoni M., Reijers H.A. (2017) Heuristic mining revamped: an interactive, data-aware, and conformance-aware miner. *15th International Conference on Business Process Management (BPM 2017)*, pp. 1–5.

16. van der Aalst W.M.P., Adriansyah A., van Dongen B. (2011) Causal Nets: A modeling language tailored towards Process Discovery. *International Conference on Concurrency Theory*, pp. 28–42. https://doi.org/10.1007/978-3-642-23217-6_3

17. van Dongen B.F., Busi N., Pinna G.M., van der Aalst W.M.P. (2007) An iterative algorithm for applying the theory of regions in process mining. Proceedings of the *Workshop on Formal Approaches to Business Processes and Web Services (FABPWS'07)*, pp. 36–55.

18. Bergenthum R., Desel J., Lorenz R., Mauser S. (2007) Process mining based on regions of languages. *Business Process Management: 5th International Conference (BPM 2007), Brisbane, Australia, September 24–28, 2007*, pp. 375–383. https://doi.org/10.1007/978-3-540-75183-0_27

19. Leemans S.J.J., Fahland D., van der Aalst W.M.P. (2013) Discovering block-structured process models from event logs: A constructive approach. *Application and Theory of Petri Nets and Concurrency: 34th International Conference* (*PETRI NETS 2013)*, *Milan, Italy, June 24–28, 2013*, pp. 311–329. https://doi.org/10.1007/978-3-642-38697-8_17

20. Leemans S.J.J., Fahland D., van der Aalst W.M.P. (2015) Scalable process discovery with guarantees. *Enterprise, Business-Process and Information Systems Modeling (BPMDS EMMSAD 2015). Lecture Notes in Business Information Processing,* vol. 214. Springer, Cham, pp. 85–101. https://doi.org/10.1007/978-3-319-19237-6_6

21. Pande S.D., Kanna R.K., Qureshi I. (2022) Natural language processing based on name entity with n-gram classifier machine learning process through ge-based hidden Markov model. *Machine Learning Applications in Engineering Education and Management*, vol. 2, no. 1, pp. 30–39.

22. Sagayam K.M., Hemanth D.J. (2019) A probabilistic model for state sequence analysis in hidden Markov model for hand gesture recognition. *Computational Intelligence*, vol. 35, no. 1, pp. 59–81. https://doi.org/10.1111/coin.12188

23. Srivastava R.K., Pandey D. (2022) Speech recognition using HMM and Soft Computing. *Materials Today: Proceedings*, vol. 51, pp. 1878–1883. https://doi.org/10.1016/j.matpr.2021.10.097

24. Du J., Wang C., Wang L. et al. (2023) Automatic block-wise genotype-phenotype association detection based on hidden Markov model. *BMC Bioinformatics*, vol. 24, article 138. https://doi.org/10.1186/s12859-023-05265-5

25. Rabiner L.R. (1990) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286. https://doi.org/10.1109/5.18626

About the author

Artem Yu. Varnukhov

Assistant, Department of Business Informatics, Ural State University of Economics, 62, 8 Marta Str., Yekaterinburg 620144, Russia

E-mail: varnuhov_ayu@usue.ru