Extending dynamic Bayesian networks for anomaly detection in complex logs

Extending Dynamic Bayesian Networks for Anomaly Detection in Complex Logs

Stephen Pauwels[1] and Toon Calders[1]

University of Antwerp, Middelheimlaan 1, Antwerp

Abstract. Checking various log files from different processes can be a tedious task as these logs contain lots of events, each with a (possibly large) number of attributes. We developed a way to automatically model log files and detect outlier traces in the data. For that we extend Dynamic Bayesian Networks to model the normal behavior found in log files. We introduce a new algorithm that is able to learn a model of a log file starting from the data itself. The model is capable of scoring traces even when new values or new combinations of values appear in the log file.

1 Introduction

We propose a way of detecting anomalous behavior in Business Processes (BPs). A BP is a series of structured activities in order to perform a task [1]. Such a sequence of events that together form an instantiation of a BP is called a trace of the business process. In order to monitor a BP, activities are logged in a log file. This file consists of different events and every line in the log file represents a single event. Often log files already indicate which events belong together in the same trace. If not we can apply a clustering algorithm as described in [2] for identifying the different traces.

Example 1. The log file in Table 1 is generated by a Business Process where an employee needs to log into a system to create a request. This request is then sent to his or her manager who can approve or reject the request. The log consists of 7 attributes: Time, EventID, Type, Activity, UserID, UserName and UserRole. We also keep track of the trace to which an event belongs. In total we have 4 users, each with a unique ID and Name. Every user has a role from a limited set of roles. For the sake of simplicity we have only captured a subset of all possible actions that can occur.

In the context of Business Processes, the detection of anomalous behavior is an important problem. Therefor, in this paper we describe an anomaly detection system that can find deviating traces. This is done by learning the structure and parameters of a model that reflects the normal behavior of a system. Our model takes all attributes and relations between attributes into account, in contrast to existing techniques [3]. Which provides us with a lot more useful information since log files created by an autonomous system often consist of many more

Time	eID	Type	Activity	UserID	UserName	UserRole	tID
0	0	User-Actions	Log in	001	User1	employee	1
1	1	User-Actions	Logged in	001	User1	employee	1
1	2	Request Permission	Create Request	001	User1	employee	1
2	3	Request Permission	Send Mail	001	User1	employee	1
3	4	User-Actions	Log in	001	User1	employee	2
4	5	User-Actions	Logged in	001	User1	employee	2
6	6	Request Permission	Create Request	001	User1	employee	2
7	7	Request Permission	Send Mail	001	User1	employee	2
8	8	Request Permission	Disapproved	002	User2	manager	2
9	9	User-Actions	Log in	003	User3	employee	3
10	10	User-Actions	Logged in	003	User3	employee	3
10	11	Request Permission	Create Request	003	User3	employee	3
11	12	Request Permission	Approved	002	User2	manager	1
12	13	Request Permission	Send Mail	003	User3	employee	3
17	14	Request Permission	Approved	004	User4	sales-manager	3
18	12	User-Actions	Log in	001	User1	manager	4
19	13	User-Actions	Logged in	001	User1	manager	4
20	14	Request Permission	Create Request	001	User1	manager	4
21	15	Request Permission	Approved	001	User1	manager	4
21	16	Request Permission	Send Mail	001	User1	manager	4

Table 1: Example Log file containing normal (black) and anomalous (red) traces

attributes. Attributes can influence each other within an event and between different events. Besides missing activities or a wrong ordering of activities there can be constraints on the activities enforcing that two activities must be performed by the same person or that a person needs to have a certain role to perform an action.

Diagrams like BPMN models are a great tool for human understanding of a Business Process. For applications such as anomaly detection, BPMN models are, however, insufficiently powerful as they lack the ability to easily express joint probability distributions and multiple attributes; they focus on a single perspective (i.e. the resource-activity perspective). Therefore, in order to take advantage of all possible relations between attributes in a log file we create a model based on Dynamic Bayesian Networks (DBNs) [4]. DBNs are an extension of Bayesian Networks that are able to incorporate discrete time. This model will link current events to their predecessors in order to find relations between these events rather than only relations within one event.

In this paper we identify and improve a number of shortcomings of DBNs when it comes to modeling the allowable sequences in a log:

- DBNs are not able to handle unseen values in an appropriate way for business process logs.
- The case where a value always occurs together with another value describes a common structure in log files. We can model these relations in a DBN but only implicit which may lead to less effective structures.

Therefor we extended the formalism of Dynamic Bayesian Networks to incorporate the aspects that are typical for log files. We will show that our extended Dynamic Bayesian Networks perform well for detecting anomalies.

The structure of our paper is as follows. Section 2 describes existing approaches to this (or similar) problems. Section 3 introduces the model for de-

scribing normal behavior in log files. We then use this model in Section 3.4 in order to discover anomalies in traces of events found in log files. The construction of the model is described in Section 4. We will evaluate our new method in Section 5.

2 Related Work

The problem we are interested in is that of finding anomalous sequences (traces) within a large database of discrete multivariate sequences. Different techniques have been proposed to solve this problem both in the anomaly detection field [5,6,7,8], as in the process mining field [9,10]. Some of these techniques use signatures of known anomalies that can occur in the system. It is clear that these systems cannot recognize a new type of anomaly and are too limited for our purpose. We are interested in techniques that build a model, such as Markov Chain, that represent normal behavior of a system.

A first type of algorithms works on a database of univariate sequences; i.e., they only take the activity into account. Bezerra et al. [9] investigated the detection of anomalies in a log file using existing Process Mining algorithms in order to build a model of the process. This model is then used to detect anomalous executions of this process. They only use information about the activities performed so they can use standard Process Mining techniques which do not take the extra attributes into account. Nolle et al. [6] propose an unsupervised anomaly detection method based on neural networks in noisy business process event logs. Using these neural networks makes it possible to reduce the impact of noise in the dataset, where other methods need a training dataset without anomalies as a reference. They do this by adding extra noise to the data before feeding the data into the neural network. Next the neural network is trained to reproduce its input. After the training phase, the network can be used to reproduce the traces from the same input log without the noise. Normal traces are expected to be reproduced with less errors than anomalous traces.

Other algorithms work on databases of multivariate sequences. Bertens [7] uses MDL to identify multivariate patterns that helps him detect and describe anomalies. A code table consisting of mappings between encodings and frequently occurring patterns is first generated by their algorithm called DITTO [11]. The anomaly score is defined by dividing the length of the encoded sequence given the code table on the whole dataset by the length of the sequence. Bohmer et al. [10] introduce a probabilistic model that is able to score events in the data. First a Basic Likelihood Graph is constructed where all activities are nodes and the edges between nodes indicate the probability that given the previous activity, a certain activity happens next. In the next phase this graph is extended by adding a *resource* and *weekday* between two activities that correspond to the resource that performed the previous action on a particular weekday. Using this graph it is possible to compute a baseline-score given the occurrence of a particular activity. This baseline-score is compared with the actual score given to an execution trace by the model. To score an actual trace Bohmer et al. use

	Univariate	Multivariate	Method
Our method		✓	Dynamic Bayesian Network
Ye [8]	✓		Markov Chains
Bezerra [9]	✓		Process Mining
Nolle [6]	✓		Neural Networks
Bertens [7]		✓	Minimum Description Length
Bohmer [10]		✓	Probabilistic Model

Table 2: Summary of Related Work in comparison with our proposed method

the data in the graph with the corresponding probabilities to get a score for the entire trace. Besides data present in the graph, the model is also able to deal with new values. However, they do not describe and test the use of more attributes in detail, but their model can be extended in a straightforward way to other attributes as well. A summary of the different techniques can be found in Table 2.

3 Extended Dynamic Bayesian Networks

In this section we will extend Dynamic Bayesian Networks to create a model which is more flexible and powerful when dealing with log files. Therefor we will first formally define a log file.

Definition 1. *We assume that $\mathcal{A} = \{A_1, \ldots, A_n\}$, an ordered set of attributes, is given as well as for each attribute A_i a set of allowed values $dom(A_i)$.*

An event description is a tuple $(a_1, \ldots, a_n)$ with $a_i \in dom(A_i)$; $desc.A_i$ denotes a_i. An event is a pair $(eID, desc)$ with eID a identifier and desc an event description. We use $e.A_i$ as a shorthand notation for $e.desc.A_i$.

A trace $t = \langle e^1, \ldots, e^i \rangle$ is a sequence of events. A log L is a set of traces, where events in the traces have different identifiers.

3.1 History and Context of an event

To be able to incorporate the timing aspect we introduce the *k-history* and *k-context* of an event.

Definition 2. *The k-history of an event e^i is defined as $\mathcal{H}_k(e^i) = x^k \cdot \ldots \cdot x^1$ with $\cdot$ the concatenation and where*

$$x^l = \begin{cases} e^{i-l}.desc & \text{if } i - l > 0 \\ (None, \ldots, None) & otherwise \end{cases}$$

None is a special dedicated value that should not occur in the log. We use $\mathcal{H}_k(e^i).A^l$ to denote the value of attribute A from the l-th event before e^i in the trace that is, $X^l.A$. When there is no ambiguity we will use the notation $\mathcal{H}(e^i).A^l$.

Definition 3. *The k-context of an event e is defined as* $\mathcal{C}_k(e) = (\mathcal{H}_k(e) \cdot e.desc)$ *we use the notations:*

$$\mathcal{C}_k(e).A := e.A$$
$$\mathcal{C}_k(e).A^l := \mathcal{H}_k(e).A^l$$

Example 2. For the log in Table 1, the *2-history* of the event with eID 3 is the tuple *(User-Actions, Logged in, 001, User1, employee, Request Permission, Create Request, 001, User1, employee).* The *2-context* of this event is the tuple *(User-Actions, Logged in, 001, User1, employee, Request Permission, Create Request, 001, User1, employee, Request Permission, Send Mail, 001, User1, employee).*

3.2 Conditional Probability Tables and Functional Dependencies

In Dynamic Bayesian Networks, the relations within the model are represented using Conditional Probability Tables (CPTs).

Definition 4. *A* $CPT(X|Y)$ *is a table where each row contains the conditional probability for a value of X given a combination of values of Y.*

The following example indicates the problems we have when using only CPTs for describing BP log files:

Example 3. Consider the situation where every User has a particular Role and certain activities can only be executed by certain roles. The attribute Role depends on the User and the Activity in this example. When building a single CPT we have to add a row for every possible combination of values for User and Activity, resulting in a large table with a lot of probabilities equal to 1. Also, when a new user is added to the system, all combinations with this user would have to be added to the CPT.

To avoid these problems we introduce a new type of relation: a Functional Dependency.

Definition 5. *Given a log L. A Functional Dependency* $A^{t_1} \rightarrow B^{t_2}$ *holds in L if for all events* $e, f \in L$ *holds that if* $\mathcal{C}(e).A^{t_1} = \mathcal{C}(f).A^{t_1} \neq None$*, then* $\mathcal{C}(e).B^{t_2} = \mathcal{C}(f).B^{t_2}$ *for attributes A and B and time steps* t_1 *and* t_2*.*

A Functional Dependency (FD) between attributes X and Y can be represented by a function $FD_{X \rightarrow Y} : a\_dom(X) \rightarrow a\_dom(Y)$, $FD_{X \rightarrow Y}(x) = y$, with x and y the respective values for attributes X and Y. $a\_dom(A)$ is defined as follows:

Definition 6. *Let L be a log over* $\mathcal{A}$ *and* $\{A_{i_1}, \ldots, A_{i_k}\} \subseteq \mathcal{A}(L)$*. We define the active domain* $a\_dom(A_{i1}, \ldots, A_{ik}) = \{(e.a_{i_1}, \ldots, e.a_{i_k}) | \exists t \in L : e \in t\}$ *as the set containing all values that occur in the log for the given attributes.*

Example 4. In the log in Table 1, $UserID \rightarrow UserRole$ is a Functional Dependency. Every value of UserID uniquely maps to a value of UserRole. A particular value in UserRole can however occur together with multiple values in UserID. We have the following mappings in our log:

$\{001 \mapsto employee,\ 002 \mapsto manager,\ 003 \mapsto employee,\ 004 \mapsto sales-manager\}$

It is possible to mimic this behavior by only using CPTs with all probabilities set to 1. Introducing Functional Dependencies, however, allows us to create easier models that can be used to express more general patterns.

A second major shortcoming of CPTs when dealing with log files is that only values that have occurred in the training dataset will be present in the tables. In a log file it might be normal for new users to appear without these events being anomalous. The model will assign a probability of 0 to these values. If the new value, however, satisfies all other relations then it is likely to be a correct event. Smoothing could be used, but may be inappropriate for attributes with frequent new values. The frequency of new values depends on the attribute itself, not on the log file.

Example 5. The attribute *Role* will never take a new value as these are fixed within the organization, while *UserName* can contain new values when a new user is added to the system.

3.3 Extending the Dynamic Bayesian Networks

Combining all these elements, we extend the definition of a DBN as follows:

Definition 7. *An extended DBN with memory k over $\mathcal{A}$ is a tuple:*

$$(G, FDR, \mathcal{CPT}, \mathcal{FD}, new\_value, new\_relation) :$$

- *$G(V, E)$ is a directed acyclic graph with $V = \mathcal{A}^k \cup \ldots \cup \mathcal{A}^1 \cup \mathcal{A}$ where $\mathcal{A}^i = \{A^i | A \in \mathcal{A}\}$ for $i = 1, \ldots, k$, and $E \subseteq V \times \mathcal{A}$.*
  $\mathcal{A}^i$ represents the attributes of the ith event before the current event.
  E expresses dependencies of the attributes of the current event on the other attributes in its context.
- *$FD \subseteq E$ denotes the set of dependencies that are functional.*
- *For each variable $A \in \mathcal{A}$, $Parents(A)$ denotes the set of variables $\{B \in V | (B, A) \in E \setminus FD\}$.*
- *$\mathcal{CPT}$ consists of a Conditional Probability Table $CPT(A|Parents(A))$ for each $A \in \mathcal{A}$*
- *$\mathcal{FD}$ consists of a Mapping $FD_{A \to B}$ for each $(A, B) \in FD$*
- *$new\_value(A)$ is a function representing $\mathcal{A} \to [0, 1]$*
- *$new\_relation(A)$ is a function representing $\mathcal{A} \to [0, 1]$*

Figure 1 shows a possible eDBN based on our example.

The probability distribution of an eDBN An eDBN represents a probability distribution over sequences as follows:

$$P(\langle e^1, \ldots, e^m \rangle) = \prod_{i=1}^{m} P(e^i | e^1, \ldots, e^{i-1}) \tag{1}$$

$$= \prod_{e \in t} P(e | \mathcal{H}_k(e)) \tag{2}$$

$$= \prod_{e \in t} \prod_{A \in \mathcal{A}} P(A = e.A | Parents(A) = \mathcal{C}_k(e)|_{Parents(A)}) \tag{3}$$

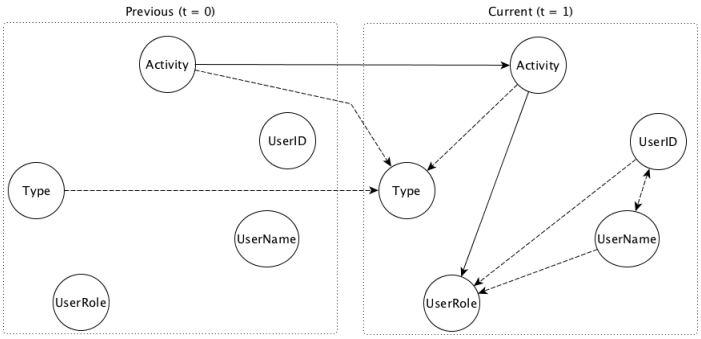


Fig. 1: eDBN with conditional (full) and functional dependencies (dotted)

The probability for an attribute in an event consists of three different parts. The first part checks for new values and is defined as:

$$value_A(x) = \begin{cases} 1 & \text{if } x \in a\_dom(A) \\ new\_value(A) & \text{otherwise} \end{cases} \tag{4}$$

The probability from the Conditional Probability Table is denoted as $CPT(x_i|Parents(X_i))$ and is defined as the probability for value x_i of event e given the parents of X_i when $\mathcal{C}_k(e)|_{Parents(X_i)}$ occurs in the training set or *new_relation(X_i)* otherwise. The probability for a Functional Dependency is expressed as follows:

$$FDM_{X,Y}(y|x) = \begin{cases} 1 & \text{if } FD_{X\rightarrow Y}(x) = y \text{ or } x \notin a\_dom(X) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

To incorporate all the new elements we introduced in our model we alter the way of determining the probability in contrast to original BNs.

$$P(e.A|Parents(A)) = value_A(e.A) \times CPT(e.A|Parents(A)) \times \prod_{(X,A)\in FDR} FDM_{X\rightarrow A}(e.A|\mathcal{C}_k(e).X))$$

Example 6. The probability for an event e in the model given in Figure 1 is equal to:

$$\begin{aligned} &value(Activity_1)CPT(Activity_1|Activity_0)value(Type_1)FDM(Type_1|Activity_0) \\ &\times FDM(Type_1|Activity_1)FDM(Type_1|Type_0)value(UserID_1)FDM(UserID_1|UserName_1) \\ &\times value(UserName_1)FDM(UserName_1|UserID_1)value(UserRole_1) \\ &\times CPT(UserRole_1|Activity_1)FDM(UserRole_1|UserID_1)FDM(UserRole_1|UserName_1) \end{aligned}$$

The value for the attribute $UserRole_1$ for the event with eID 2 is:

$$\begin{aligned} &value(UserRole_1)CPT(UserRole_1|Activity_1) \\ &\quad \times FDM(UserRole_1|UserID_1)FDM(UserRole_1|UserName_1) \\ &= 1 \times 0.5 \times 1 \times 1 \end{aligned}$$

3.4 Anomaly detection

To find anomalous sequences of events we use a score-based approach. The score is obtained by calculating the probability for a trace $\langle e^1, \ldots, e^n \rangle$ given a model m. We normalize the result using the n-th root, with n the number of events in the trace. This normalization makes sure that longer traces are not penalized.

$$Score(\langle e^1, \ldots, e^n \rangle) = \sqrt[n]{P(\langle e^1, \ldots, e^n \rangle)} \tag{6}$$

Sequences with a high score thus have a high probability of occurring and are most likely to represent normal behavior, whereas low scores indicate higher chances of being an anomaly. We return a sorted list of traces, sorted by their scores. The idea is that a user can only handle the first k anomalies detected. Since we can score any sequence of events, we do not have to wait for a complete trace before we can score it. The model can thus be used to detect anomalies in ongoing traces.

4 Learning the structure and parameters of the model

We build our model using a reference dataset containing only the normal execution of the process. Our experiments show that the performance of our algorithm is, however, not influenced when the dataset contains a small amount of noise. In order to incorporate the timing aspect we replace every event in the log with its *k-context*. We refer to this log as the *k-context log*.

We can use the k-context log as input for traditional Bayesian Network learning algorithms that have no specific knowledge about the different time steps to find the conditional probability tables. Afterwards we interpret the different attributes in their appropriate time slice. The complete algorithm for computing the structure can be found in Algorithm 11.

First the algorithm searches for Functional Dependencies in the data. In order to discover them, the Uncertainty Coefficient [12] is applied to the k-context log, which is defined as follows for the random variables X and Y:

$$U(X|Y) = \frac{I(X;Y)}{H(X)} \ ,$$

Function *LearnEDBN*
  Data: variables, FDThreshold
  Result: The learned eDBN
  V = $vars$
  FD = $\{X \to Y : X, Y \in V \mid U(X|Y) > FDThreshold\}$
  blacklist = $\{X \to Y : \forall X \in V_i, Y \in V_j \; with \; i \geq j > 0\}$
  whitelist = FD
  G(V, E) = LearnBayesianNetwork(variables = V, blacklist, whitelist)
  FDS = ConstructFunctionalDependencyFunctions(FD)
  CPT = ConstructConditionalProbabilitiesTables($E \setminus FD$)
  NV = $\{X \mapsto \frac{|a\_dom(X)|}{\mid L \mid} : \forall X \in V\}$
  NR = $\{X \mapsto \frac{|a\_dom(Parents(X))|}{\mid L \mid} : \forall X \in V\}$
  return *eDBN(G(V, E \ FD), FD, CPT, FDS, NV, NR)*

Algorithm 1: Algorithm for learning the structure and parameters of an Extended Dynamic Bayesian Network

with $H(X)$ the *entropy* [13] of X and I(X;Y) the *Mutual Information* [14] given as:

$$I(X;Y) = \sum_{y \in a\_dom(Y)} \sum_{x \in a\_dom(X)} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

$$H(X) = - \sum_{x \in a\_dom(X)} p(x) log(p(x))$$

The Uncertainty Coefficient is the normalized form of Mutual Information. It gives information about how much the values of an attribute depend on another attribute. We use it to determine what attributes are related to each other and how much they relate to each other. The measure ranges from 0 (no correlation between the two attributes) to 1 (completely correlated attributes, thus indicating the existence of a Functional Dependency) [15]. If $U(X|Y) > threshold$, we will assume that the FD $Y \to X$ holds. This threshold has to be chosen according to the amount of noise in the data. A higher threshold means a more strict Functional Dependency is used that is less able to cope with noise.

Example 7. When we calculate the Uncertainty Coefficient of *UserID* and *pUserID* we get: $U(UserID|pUserID) = \frac{I(UserID;pUserID)}{H(UserID)} = \frac{0.5597}{1.1369} = 0.4923$

For an attribute A in log L, $new\_value(A)$ and $new\_relation(A)$ are defined as follows:

$$new\_value(A) : \frac{|a\_dom(A)|}{\mid L \mid}, \quad new\_relation(A) : \frac{|a\_dom(Parents(A))|}{\mid L \mid}$$

Example 8. The New Value Rate of the *UserRole* is equal to $\frac{3}{15} = 0.2$. The rate for *Activity* is equal to $\frac{6}{15} = 0.4$. This indicates that new values are more likely to occur for the attribute *Activity* than for *UserRole* according to our data.

With a standard Bayesian Network learning algorithm we can discover the Conditional Dependencies present in the data. It is possible to use any learning algorithm that uses data to learn its structure. We choose to use a Greedy algorithm that finds a local optimum for the Akaike Information Criterion (AIC) [16].

The relations present in our model should only indicate a causality relation; events in the present cannot influence events in the past. Therefore edges that do not represent a causality relation are blacklisted. This blacklist is created by adding all edges that do not end in the *current* time step.

We do not want the algorithm to find edges already labeled as FDs, therefor we add these edges to a whitelist. The Bayesian Net learning algorithm should always include the edges from the whitelist in the model. This way the learning algorithm takes advantage of the information we already know about these FDs.

After running the greedy algorithm we have found the Conditional and Functional Dependencies that define the structures present in our data. We can then combine them into one single model. This gives us the structure of the eDBN-model. The next step in building the model is filling in all the different Conditional Probability Tables (CPTs) and constructing the Functional Dependency functions for all nodes.

5 Experiments

To properly test our newly proposed method we use two different datasets. The first dataset is a synthetically generated multi-dimensional dataset. The second is the BPI Challenge 2015 (BPIC15) [19] data. This data consists of applications for building permits in 5 Dutch municipalities, we refer to these as BPIC1 to BPIC5, Table 3 summarizes the data. We use this last dataset in two different forms: the dataset with anomalies introduced and the dataset with a reduced subset of attributes with anomalies included. We included the anomalies using a similar approach as described by Bohmer [10].

We use the synthetic dataset to test the overall performance of our algorithm, where we try different ratios of anomalies in both training and test set. Next we perform an in-depth comparison with the Likelihood Graphs proposed by Bohmer et. al [10] using the reduced subset of the BPIC15 data. Furthermore we compare our approach to a variety of algorithms available in the ELKI - tool [20], using both the synthetic data and the reduced BPIC15 data. The Area

Dataset	Number of traces	Average trace length	Number of Activities
BPIC1	1199	43.5	398
BPIC2	832	53.3	410
BPIC3	1409	42.3	383
BPIC4	1053	44.9	359
BPIC5	1156	51.0	389

Table 3: Description of the BPIC datasets

		Test set							
	% Anomalies	0.1	0.5	1.0	2.5	5.0	10.0	25.0	50.0
Training set	0.0	0.97	0.99	0.99	0.96	0.98	0.98	0.92	0.69
	0.5	0.99	0.99	0.93	0.96	0.98	0.99	0.92	0.70
	1.0	1.0	1.0	0.94	0.97	0.99	0.98	0.95	0.69
	2.5	1.0	1.0	0.95	0.99	0.99	0.98	0.92	0.68

Table 4: AUC values for different combinations of anomalies.

Under the Curve (AUC) is used to compare the algorithms. At the end we also perform a qualitative experiment showing that our method is also applicable for the analysis of log files to detect concept drift. All code used to perform the experiments and generate the datasets can be found on our GitHub repository[1].

5.1 Testing with synthetic data

We built a data generation tool that allows us to create log files containing different relations between events. In order to do so we first create a model of sequential activities with depending attributes. The model is based on a BP for shipping goods. Goods can have a value and an extra insurance can be taken. Goods with an extra insurance need a different workflow from goods without extra insurance. We create one model for normal execution and one model for anomalous execution, where we explicitly changed the order of events or use the wrong flow of events according to the insurance chosen. Next we introduce some extra attributes where some of these attribute depends on other attributes. For the anomalous traces we added random values on random places. We generated multiple set-ups with a variable number of anomalies in both training and test data. We added anomalies in our training data to check and show that our approach does not require a flawless log file as training data but is able to deal with a small amount of unexpected behavior in the data.

The AUC-scores for different amounts of anomalies in both training and test data can be found in Table 4. This test shows that our algorithm is able to find the relations mentioned in Section 3, even when the training set contains a small amount of noise or anomalies.

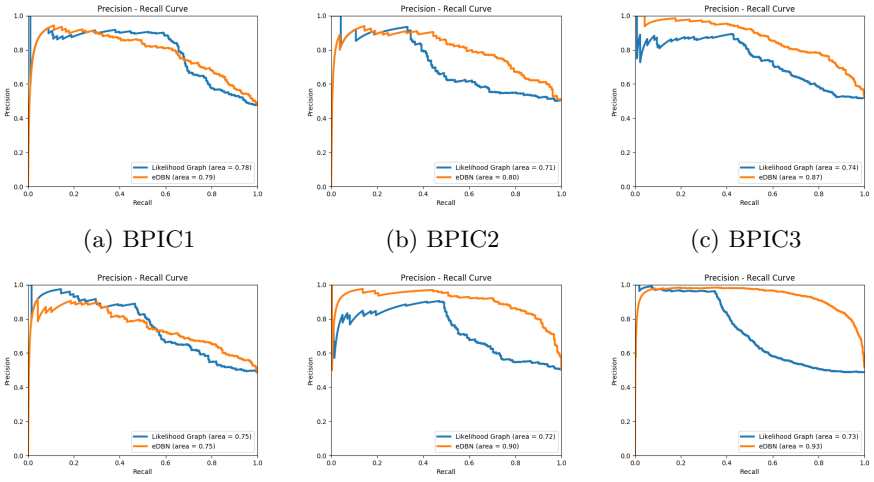
5.2 Comparison

Comparison with Likelihood Graphs In order to compare our approach to the solution presented by Bohmer we first implemented the algorithm found in [10]. Next we generated data as described by Bohmer et. al starting from the reduced BPIC data. Therefor we randomly split the original data in two equal data sets, one for training and one for testing. In the test data we introduced anomalies according to the description in Bohmer et al. The statistics for the generated files can be found in Table 5.

[1] https://github.com/StephenPauwels/edbn_ecmlpkdd

File	Training size	Test size	Number of anomalies in Test set
BPIC1	589	610	291 (47.7%)
BPIC2	408	423	214 (50.5%)
BPIC3	723	686	356 (51.8%)
BPIC4	522	530	257 (48.4%)
BPIC5	595	561	283 (50.4%)

Table 5: Number of traces present in the different log files.



(a) BPIC1 (b) BPIC2 (c) BPIC3

(d) BPIC4 (e) BPIC5 (f) Total BPIC

Fig. 2: Comparison of precision/recall graphs.

The Likelihood calculates the likelihood for the ongoing trace and compares this with a baseline score in order to indicate if a trace is an anomaly. Since our method works with giving scores and sorting all anomalies, we used the minimum of the difference between the Ongoing Likelihood and the Minimum Likelihood for the different activities within a trace in order to best capture the ideas of the Likelihood Graphs. The lower the difference the more likely it is that this trace contains an anomaly. We used the precision/recall curve to compare the two approaches. The results can be found in the graphs in Figure 2 for each of the five different municipalities. Since all five municipalities have different ways of performing the different processes we also created one file containing all data of all municipalities. Then we introduced anomalies in the same way as we did for the other files. This combined dataset allows us to test how well each approach can handle different processes in a single log file.

We can see that our method mostly outperforms the Likelihood Graph on the BPIC2015 data. When we look at the results for the combined log file, we can see that we clearly outperform the Likelihood Graph. So we can conclude

	AUC	
Method	Synth data	BPIC data
eDBN	1.00	0.84
Bohmer [10]	1.00	0.58
FastABOD [21]	0.50	0.56
LOF [22]	0.49	0.55
SOD [23]	0.53	0.60
Feature Bagging [24]	0.75	0.57
SimpleCOP [25]	0.53	0.60
LibSVMOneClassOutlierDetection [26]	0.51	0.46
COP [27]	0.81	0.63
DWOF [28]	0.51	0.44
OpticSOF [29]	0.53	0.46
LBABOD [21]	0.50	N/A
ALOCI [30]	N/A	0.86

Table 6: Overview of results for different Anomaly detection techniques.

that our model is capable of performing well even with multiple processes in the log file.

Comparison with other anomaly detection methods We also tested our method against other anomaly detection methods (not necessarily methods that take into account the sequential nature). We used the *k-context* format as input for all the algorithms. The best parameters were chosen after performing some experiments. We performed the experiments using the ELKI - tool [20]. Since none of these methods uses a different (clean) training dataset we used the same file to generate our model as to test the model. The results can be seen in Table 6.

6 Conclusion

In this paper we extended Dynamic Bayesian Networks in order to create a new model that allows us to better and in more detail describe the structure and properties of a log file generated by process-aware information systems. As standard DBNs have shortcomings for analyzing these logs we added some elements to cope with these shortcomings. We added Functional Dependencies for a better description of the structure of a log file. Since DBNs cannot cope with unseen values we also improved the way our model deals with these unseen values. Next we described our algorithm for creating models that reflect the multidimensional and sequential nature of log files. We conducted different types of experiments: the first experiment confirmed that our algorithm achieves high performance in different settings with different amounts of anomalies in both training and test sets. Next we compared our approach with existing solutions. These tests concluded that our model is able to better filter out anomalies.

In the future we would like to extend our model even further in order to incorporate the time aspect even better by introducing an extra timing element that is capable of dealing with duration of activities and time gaps between activities. We would also want to introduce continuous variables in our model.

References

1. Von Rosing, M., Von Scheel, H., Scheer, A.W.: The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM. Volume 1. Morgan Kaufmann (2014)
2. Pauwels, S., Calders, T.: Mining multi-dimensional complex log data. Benelearn (2016)
3. Van der Aalst, W.M., de Medeiros, A.K.A.: Process mining and security: Detecting anomalous process executions and checking process conformance. Electronic Notes in Theoretical Computer Science **121** (2005) 3–21
4. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach (3rd edition) (2009)
5. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection for discrete sequences: A survey. IEEE TKDE **24**(5) (2012) 823–839
6. Nolle, T., Seeliger, A., Mühlhäuser, M.: Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: International Conference on Discovery Science, Springer (2016) 442–456
7. Bertens, R.: Insight Information: from Abstract to Anomaly. Universiteit Utrecht (2017)
8. Ye, N., et al.: A markov chain model of temporal behavior for anomaly detection. In: Procs of the 2000 IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop. Volume 166., West Point, NY (2000) 169
9. Bezerra, F., Wainer, J., van der Aalst, W.M.: Anomaly detection using process mining. In: Enterprise, business-process and information systems modeling. Springer (2009) 149–161
10. Böhmer, K., Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer (2016) 80–98
11. Bertens, R., Vreeken, J., Siebes, A.: Keeping it short and simple: Summarising complex event sequences with multivariate patterns. arXiv preprint arXiv:1512.07056 (2015)
12. Press, W.H.: Numerical recipes 3rd edition: The art of scientific computing. Cambridge university press (2007)
13. Shannon, C.E.: A mathematical theory of communication. SIGMOBILE Mob. Comput. Commun. Rev. **5**(1) (January 2001) 3–55
14. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons (2012)
15. White, J., Steingold, S., Fournelle, C.: Performance metrics for group-detection algorithms. Proceedings of Interface 2004 (2004)
16. Akaike, H.: A new look at the statistical model identification. IEEE transactions on automatic control **19**(6) (1974) 716–723
17. Campos, L.M.d.: A scoring function for learning bayesian networks based on mutual information and conditional independence tests. Journal of Machine Learning Research **7**(Oct) (2006) 2149–2187

18. Margaritis, D.: Learning bayesian network model structure from data. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE (2003)
19. van Dongen, B.: (2015) bpi challenge 2015. eindhoven university of technology. dataset. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
20. Schubert, E., Koos, A., Emrich, T., Züfle, A., Schmid, K.A., Zimek, A.: A framework for clustering uncertain data. PVLDB **8**(12) (2015) 1976–1979
21. Kriegel, H.P., Zimek, A., et al.: Angle-based outlier detection in high-dimensional data. In: Procs of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2008) 444–452
22. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: ACM sigmod record. Volume 29., ACM (2000) 93–104
23. Kriegel, H.P., Kröger, P., Schubert, E., Zimek, A.: Outlier detection in axis-parallel subspaces of high dimensional data. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer (2009) 831–838
24. Lazarevic, A., Kumar, V.: Feature bagging for outlier detection. In: Procs of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, ACM (2005) 157–166
25. Zimek, A.: Correlation clustering. ACM SIGKDD Explorations Newsletter **11**(1) (2009) 53–54
26. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural computation **13**(7) (2001) 1443–1471
27. Kriegel, H.P., Kroger, P., Schubert, E., Zimek, A.: Outlier detection in arbitrarily oriented subspaces. In: Data Mining (ICDM), 2012 IEEE 12th International Conference on, IEEE (2012) 379–388
28. Momtaz, R., Mohssen, N., Gowayyed, M.A.: Dwof: A robust density-based outlier detection approach. In: Iberian Conference on Pattern Recognition and Image Analysis, Springer (2013) 517–525
29. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Optics-of: Identifying local outliers. In: European Conference on Principles of Data Mining and Knowledge Discovery, Springer (1999) 262–270
30. Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: Data Engineering, 2003. Proceedings. 19th International Conference on, IEEE (2003) 315–326
31. Van der Ham, U.: Benchmarking of five dutch municipalities with process mining techniques reveals opportunities for improvement. Business Process Intelligence Challenge (2015)