Predictive Business Process Monitoring with Structured and Unstructured Data

Irene Teinemaa[1,2(✉)], Marlon Dumas[1], Fabrizio Maria Maggi[1], and Chiara Di Francescomarino[3]

[1] University of Tartu, Tartu, Estonia
{irheta,marlon.dumas,f.m.maggi}@ut.ee
[2] STACC, Tartu, Estonia
irene.teinemaa@gmail.com
[3] FBK-IRST, Trento, Italy
dfmchiara@fbk.eu

Abstract. Predictive business process monitoring is concerned with continuously analyzing the events produced by the execution of a business process in order to predict as early as possible the outcome of each ongoing case thereof. Previous work has approached the problem of predictive process monitoring when the observed events carry structured data payloads consisting of attribute-value pairs. In practice, structured data often comes in conjunction with unstructured (textual) data such as emails or comments. This paper presents a predictive process monitoring framework that combines text mining with sequence classification techniques so as to handle both structured and unstructured event payloads. The framework has been evaluated with respect to accuracy, prediction earliness and efficiency on two real-life datasets.

Keywords: Process monitoring · Predictive monitoring · Text mining

1 Introduction

Business process monitoring is concerned with the analysis of events produced during the execution of a process in order to assess the fulfillment of its compliance requirements and performance objectives [7]. Monitoring can take place offline (e.g., based on periodically produced reports) or online via dashboards displaying the performance of currently running cases of a process [3].

Predictive business process monitoring [15] refers to a family of online process monitoring methods that seek to predict as early as possible the outcome of each case given its current (incomplete) execution trace and given a set of traces of previously completed cases. In this context, an outcome may be the fulfillment of a compliance rule, a performance objective (e.g., maximum allowed cycle time) or business goal, or any other characteristic of a case that can be determined upon its completion. For example, in a sales process, a possible outcome is the placement of a purchase order by a potential customer, whereas in a debt recovery process, a possible outcome is the receipt of a debt repayment.

© Springer International Publishing Switzerland 2016
M. La Rosa et al. (Eds.): BPM 2016, LNCS 9850, pp. 401–417, 2016.
DOI: 10.1007/978-3-319-45348-4_23

Existing approaches to predictive process monitoring [5,13,15,16] consider that a trace consists of a sequence of events with a structured data payload, such as a payload consisting of attribute-value pairs. For example, in a loan application process, one event could be the receipt of a new loan application. This event may carry structured data such as the name, date of birth and other personal details of the applicant, the type of loan requested, and the requested amount and valuation of the collateral. Each subsequent event in this process may then carry additional or updated data such as the credit score assigned to the applicant, the maximum loan amount allowed, the interest rate, etc.

In practice, not all data generated during the execution of a process is structured. For example, in said loan application process, the customer may include a free-text description of the purpose of the loan. Later, a customer service representative may attach to the case the text of an email exchanged with the customer regarding her employment details, while a credit officer may add a comment to the loan application following a conversation with the customer. Comments like these ones are common for example in application-to-approval processes, issue-to-resolution and claim-to-settlement processes, where the execution of the process involves unstructured interactions with the customer.

This paper studies the problem of jointly exploiting unstructured (free-text) and structured data for predictive process monitoring. The contribution is a predictive process monitoring framework that combines text mining techniques to extract features from textual payload, with (early) sequence classification techniques for structured data. The proposed framework is evaluated on two real-life datasets: a debt recovery process, where the outcomes are either a (partial) repayment or the referral of the case to an external agency for encashment, and a lead-to-contract process, where the outcome conveys whether or not a sales contract is signed with a potential customer.

The rest of the paper is structured as follows. Section 2 introduces the text mining techniques upon which our proposal builds. Section 3 presents the predictive process monitoring framework, while Sect. 4 presents the evaluation. Section 5 discusses related work while Sect. 6 summarizes the contribution and outlines future work directions.

2 Background: Text Mining

The central object in text mining is a *document* — a unit of textual data such as a comment or an e-mail. Natural language processing can be used to derive representative feature vectors for individual documents, which can thereupon be used in various (predictive) data mining tasks. In order to construct reasonable representations, the textual data should be preprocessed. Firstly, the text needs to be *tokenized* — segmented into meaningful pieces. In the simplest approach, text is split into tokens on the white space character. More sophisticated tokenization techniques can be used to obtain multi-word tokens (e.g., “New York”) or to separate words such as “it’s” into two tokens “it” and “is”.

Tokens can also be *normalized* so that tokens with small differences (e.g., “e-mail” and “email”) are equated. In addition, inflected forms of words can be grouped together using *stemming* or *lemmatization*. For instance, lemmatization can group words "good", "better", and "best" under a single *lemma*.

A document can be represented by using frequencies of single words as features. For example, the document "The fox jumps over the dog" is represented as {"the":2, "fox":1, "jumps":1, "over":1, "dog":1}. This representation ignores the order of words – a limitation that can be overcome by using sequences of two (*bigrams*), three (*trigrams*), or n (*n-grams*) contiguous words instead of or in addition to single words (*unigrams*). The bigrams in the above document are: {"the fox":1, "fox jumps":1, "jumps over":1, "over the":1, "the dog":1}. Features that are constructed based on words that occur in the document are called *terms*, while the corresponding representation is called *bag-of-n-grams* (BoNG).

Terms that occur frequently in a document collection are not representative of a particular document, yet they receive misleadingly high values in the basic BoNG model. This problem can be addressed by normalizing the term frequencies (tf) with the *inverse document frequencies* (idf) — the number of all documents divided by the number of documents that contain the term, scaled logarithmically. Thus, rare terms receive higher weights, while frequent words (like "with" or "the") receive lower weights. In text classification scenarios, weighing the term frequencies with *Naive Bayes* (NB) log count ratios may improve the accuracy of the predictions [19]. The BoNG model also suffers from high dimensionality, as each document is represented by as many features as the number of terms in the *vocabulary* (the set of all terms in the document collection). Common practice is to apply *feature selection* techniques, such as mutual information or Chi-square test, and retain only the most relevant terms.

Alternative approaches to the BoNG model are *continuous representations* of documents. These techniques represent text with real-valued low-dimensional feature vectors, where each feature is typically a *latent* variable — inferred from the observed variables. One such approach is *topic modeling*, which extracts abstract *topics* from a collection of documents. The most widely used topic modeling technique, Latent Dirichlet Allocation (LDA) [1], is a generative statistical model, which assumes that each document entails a mixture of topics and each word in the document is drawn from one of the underlying topics.

Continuous representations of words using neural network-based language models have also shown high performance in natural language processing tasks. These language models are trained to predict a missing word, given its *context* — words in the proximity of the word to be predicted. Techniques have been proposed that extend these approaches from word-level to sentence-, or document-level. For instance, *Paragraph Vector* (PV) [12] generates fixed-length feature representations for documents of variable length.

3 Framework

The proposed framework takes as input a set of traces and a labeling function that assigns a label (e.g., positive vs.negative) to each trace. Given this labeled set of traces, and the incomplete trace of a running case, it returns as output a prediction on the outcome (label) of the running case. Each trace consists of a sequence of events carrying a payload consisting of structured and unstructured data. For example, the following is a possible event (*Call*) in a debt collection process, carrying structured data (*revenue* and *debt_sum*) and unstructured data (the associated textual description).

Call {*revenue* : 34555, *debt_sum* : 500} {Please send a warning. 1234567: "Gave extension of 5 days and issued a warning about sending it to encashment. An encashment warning letter sent on the 06/10, 11:10 deadline."} (1)

The framework embodies two different components. An *offline component* uses historical traces to train classifiers that are used to make predictions about running cases through an *online component*. The following subsections explain each of these components in more detail.

3.1 Offline Component

Figure 1 illustrates the offline component of our proposed predictive monitoring framework. At the core of the framework, there are *text models* and *classifiers*. Both are trained using prefixes of historical cases. In particular, one text model and one classifier is trained for each possible *prefix length* (from 1 to m). From all prefixes of a certain length, unstructured sequences (sequences of events with their associated textual description) and structured sequences (sequences of events with their structured data payload) are extracted (*Extract sequences* in Fig. 1). A textual model is trained by using the unstructured data (*Construct text model* in Fig. 1). The purpose of a textual model is to transform a variable length textual description associated to an event into a fixed length numerical feature vector. Each textual description extracted from the considered prefixes is translated into a feature vector (*Extract textual features* in Fig. 1). A classifier is then trained by encoding each prefix as a *complex sequence*, combining (i) control flow, (ii) structured data payload, and (iii) features extracted from textual data. Therefore, the number of features depends on the prefix length k (from 1 to m) and thus different classifiers need to be trained for different prefix lengths. We now describe in more detail the main phases in the offline component.

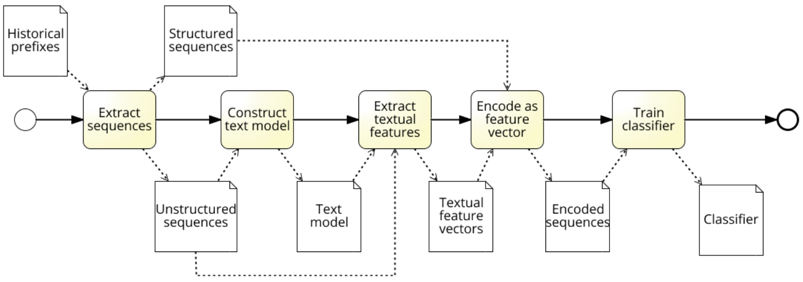


Fig. 1. The **offline** component of the proposed framework

Construct Text Models and Extract Textual Features: In the proposed framework, the text associated to each event is considered as a document and a feature vector is extracted from it. We compare 4 different techniques for extracting feature vectors from text: BoNG model with and without NB log count ratios, LDA topic modeling, and PV.

Before feature extraction, some preprocessing is done on the unstructured data. We start with tokenizing the documents, using simple white space tokenization. In the case of the running example (1), the tokenization produces a vector of tokens, e.g., "Please", "send", "a", "warning", ... Moreover, we generate equivalence classes for different types of numerals by replacing them with a corresponding tag (phone number, date, or other). For example, in (1), token "1234567" would be replaced by token "phone number", token "06/10" by "date" and token "11:10" by "time". Lastly, we *lemmatize* the text, i.e., we group together different inflected forms of a word and we refer to such a group with its base form or *lemma*. For example, in our running example, tokens "send", "sent" and "sending" will be clustered together into a "send" cluster (where "send" is the lemma), whereas "deadlin" is the base form of "deadline" and "deadlines".

In the following paragraphs, we illustrate in detail the techniques for extracting feature vectors from text we use in this paper.

Bag-of-n-grams (n, idf): This method is based on the BoNG model and takes as inputs two parameters: n, which is the maximum size of the n-grams; and *idf*, that is a boolean variable specifying whether the BoNG model is normalized with idf. In this method, the documents from historical prefixes are used to build a vocabulary of n-grams, $V(n)$. Given a vocabulary $V(n)$ of size $|V(n)| = v$, a document j is represented as a vector $\mathbf{d}^{(j)} = (g_{t_1}^{(j)}, g_{t_2}^{(j)}, ..., g_{t_v}^{(j)})$, where:

$$g_{t_i}^{(j)} = \begin{cases} tfidf(t_i{}^{(j)}) & \text{if } idf \\ f_{t_i}^{(j)} & \text{otherwise} \end{cases}$$

$f_{t_i}^{(j)}$ represents the frequency of n-gram t_i in document $\mathbf{d}^{(j)}$, i.e., $f_{t_i}^{(j)} = tf(t_i{}^{(j)})$.

For instance, in our running example (1), if $n = 1$, $idf = false$ and the vocabulary is $V(1) = \{about, agenc, collect, commun, date, deadlin, encash, extens, gave, issu, letter, number, phonenumb, pleas, send, time, warn, warning\}$, the vector encoding the textual description would be:

$$d^{(\bar{j})} = (1,0,0,0,0,1,1,1,1,1,1,1,1,1,3,1,1,3) \tag{2}$$

where the word "send" (and its variations) occur three times in the document, the word "about" occurs once and the word "agency" does not occur.

Naive Bayes log count ratios (n, α): In this method, features are still based on the BoNG model, but they are weighted with NB log count ratios, $\mathbf{d}^{(j)} = (f_{t_1}^{(j)} \cdot r_1, f_{t_2}^{(j)} \cdot r_2, ..., f_{t_v}^{(j)} \cdot r_v)$. The parameter α is a *smoothing parameter* for the weights [19], while n is, as in BoNG, the maximum size of the n-grams. For instance, if the vocabulary (and hence the term frequency) is the same as the one in (2) and the NB log count ratios vector is $\mathbf{r} = (0.85, 1.02, 0.76, 0.76, 1.52, 2.03, 1.19, 1.02, 0.45, 0.89, 1.02, 1.4, 1.39, 0.41, 1.02, 1.38, 1.27, 1.83)$, $\mathbf{d}^{(\bar{j})}$ would be:

$$\mathbf{d}^{(\bar{j})} = (0.85, 0, 0, 0, 1.52, 2.03, 1.19, 1.02, 0.45, 0.89, 1.02, 1.4, 1.39, 0.41, 3.06, 1.38, 1.27, 5.49) \quad (3)$$

Latent Dirichlet Allocation topic modeling (num_topics, idf): In this method, the text model is represented by *topics* covered by the documents. The method takes as input the *number of topics* to be obtained and, similarly to BoNG, a boolean parameter *idf* that determines whether the term frequencies should be weighted with *idf* before applying topic modeling. A topic is expressed as a probability distribution over words, where words that are characteristic to a particular topic possess higher values. Each document is represented as a probability vector over topics, $\mathbf{d}^{(j)} = (p_1^{(j)}, p_2^{(j)}, ..., p_c^{(j)})$, where c is the number of topics and $p_i^{(j)}$ is the probability that document j concerns topic p_i. For instance, if the three following topics have been identified by applying topic modeling to the textual descriptions of the historical unstructured data:

$$\begin{aligned}
&topic1 :(immediately : 0.4, phone : 0.2, pay : 0.1, ...) && (immediate\ \ payment)\\
&topic2 :(mobile : 0.3, answer : 0.2, switched : 0.15, off : 0.15, ...) && (not\ \ accessible\ \ by\ \ phone)\\
&topic3 :(send : 0.35, letter : 0.2, warning : 0.1, ...) && (warning\ \ letter\ \ sent)
\end{aligned}$$

(each topic can be abstracted by using textual descriptions like the ones reported on the right-hand side of the list of topics), the textual description in (1) will be represented as a vector of three items. Each item corresponds to the probability that the document concerns *topic*1, *topic*2 and *topic*3, respectively. In particular, the document is not very related to *topic*1, a bit more to *topic*2 and closer to the warning letter scenario. The resulting vector is:

$$d^{(\bar{j})} = (0.1, 0.2, 0.7) \quad (4)$$

Paragraph Vector (vector_size, window_size): In this method, not only terms but also the sequence of terms are exploited for the construction of the model. Namely, the method slides a window of size *window_size* over the documents, using each of such windows of words as the context. Once trained, the model is able to provide for each document a vector of features of a fixed length (specified by *vector_size*).

For the methods based on the BoNG model with and without NB log count ratios, before the textual features can be used for the complex sequence encoding, a *feature selection* step is required to reduce the number of features extracted. In particular, for the method based on the basic BoNG model the Chi-square test is used, while for the method based on the BoNG model with NB log count ratios the most discriminative features (i.e., the terms that achieve the top lowest and top highest NB log ratio scores) are selected. Both these feature selection techniques take as input the number of features to select, so that BoNG and NB also require such a number as additional input parameter.

Encode as Feature Vector: Our approach utilizes the index-based encoding for complex sequences [13]. This encoding scheme differentiates between static and dynamic (structured) data. Case attributes are static since they do not change as the case progresses. On the other hand, dynamic attributes may take new values during the execution of a case. Event attributes can hence be considered either as static (only the most recent value is used) or dynamic (the sequence of values up to a given point is used). Given a sequence σ_i of length k, with u static features $s_i^1, ..., s_i^u$, and r dynamic features $h_i^1, ..., h_i^r$, the index-based feature vector g_i of σ_i is:

$$g_i = (s_i^1, ..., s_i^u, event_{i1}, ..., event_{ik}, h_{i1}^1, ..., h_{ik}^1, ..., h_{i1}^r, ..., h_{ik}^r).$$

We enhance the index-based encoding with textual features by concatenating them to the feature vector. Textual data can be of both static and dynamic nature. When text contains static information, the derived v features $t_i^1, ..., t_i^v$ are added to the feature vector of σ_i as follows:

$$g_i = (s_i^1, ..., s_i^u, event_{i1}, ..., event_{ik}, h_{i1}^1, ..., h_{ik}^1, ..., h_{i1}^r, ..., h_{ik}^r, t_i^1, ..., t_i^v).$$

On the other hand, if textual data changes throughout the case, it should be handled in the same way as dynamic structured data:

$$g_i = (s_i^1, ..., s_i^u, event_{i1}, ..., event_{ik}, h_{i1}^1, ..., h_{ik}^1, ..., h_{i1}^r, ..., h_{ik}^r, t_{i1}^1, ..., t_{ik}^1, ..., t_{i1}^v, ..., t_{ik}^v).$$

For instance, if the case containing the event in the example (1) does not contain any static structured and unstructured data, using the topic model vector in (4), the complex sequence would be:

$$g^i = (..., call, ..., 34555, 500, ..., 0.1, 0.2, 0.7, ...) \tag{5}$$

Train Classifier: We use *random forest* [2] and *logistic regression* [9] to build the classifiers. Random forest has been shown to be a solid classifier in various problem settings, including credit scoring applications [14]. On the other hand, logistic regression, one among the most popular linear classifiers in text classification tasks, suites well to cases in which data are very sparse (this is the case when the BoNG model is used).

3.2 Online Component

The structure of the online component of our predictive monitoring framework is presented in Fig. 2. When predicting the outcome for a running case of prefix length k, the pre-built textual model and classifier for length k are retrieved and applied to the running case at hand. If the prefix length of the running case is larger than the maximum prefix length m used in the training process, only the first m events of the running case are used.

Threshold *minConf* is an input parameter of the framework. If the classifier returns a probability higher than minConf for the *positive* class, the framework outputs a *positive* prediction. If the probability is lower than the threshold, no

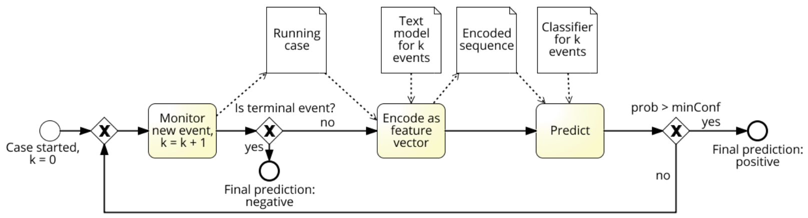


Fig. 2. The **online** component of the proposed framework

prediction is made and the framework continues to monitor the case. When the observed event is a terminal event, the final prediction is *negative*.

This setting, where the framework focuses only on making positive predictions, follows closely most real-life scenarios. Indeed, it is important for the stakeholders to filter the cases that may become deviant in the future, so that preventive actions can be taken. On the other hand, in cases that will likely have a normal outcome, no specific action is taken and they are allowed to continue in their own path. Still, our framework is easily extensible to early prediction of both positive and negative outcomes.

4 Evaluation

We have implemented the proposed methods in Python[1] and evaluated their performance on two datasets using an existing technique for predictive process monitoring with structured data as a baseline [13]. Below we describe the datasets, evaluation method and findings.

4.1 Datasets

We evaluated our framework on two real-life datasets pertaining to: (i) the debt recovery (DR) process of an Estonian company that provides credit management service for its customers (creditors), and (ii) the lead-to-contract (LtC) process of a due diligence service provider in Estonia.

The debt recovery process starts when the creditor transfers a delinquent debt to the company. This means that the debtor has already *defaulted* — failed to repay the debt to the creditor in due time. Usually, the collection specialist makes a phone call to the debtor. If the phone is not answered, an inquiry/reminder letter is sent. If the phone is answered, the debtor may provide an expected payment date, in which case no additional action is taken during the present week. Alternatively, the specialist and the debtor can agree on a payment schedule that outlines the repayments over a longer time period. If the collection specialist considers the case to be irreparable, she makes a suggestion to the creditor

[1] Scripts available at https://github.com/irhete/PredictiveMonitoringWithText.

about forwarding the debt to an outside debt collection agency (*send to encashment*) or about sending a warning letter to the debtor on the same matter. The final decision about issuing an encashment warning to the debtor and/or sending the debt to encashment is made by the creditor. If there is no advancement in collecting the debt after 7 days (e.g., the payment was not received on the provided date or the debtor has neither answered the phone nor the reminder letter), the procedure is repeated.

It is in the interest of the creditor to discover, as early as possible, cases that will not lead to any payment in a reasonable timeframe. The earlier the debt is recovered, the more value it entails for the creditor. Moreover, such cases are likely irreparable and could be sent to encashment without further delay. Therefore, our prediction goal is to determine cases where no payment is received within 8 weeks after the beginning of the debt recovery process.

The lead-to-contract process is logged through a customer relationship management system (CRM). The process begins when the sales manager selects companies as "cold leads" and loads them into CRM. Based on personal experience, the sales manager selects leads that qualify for an opportunity, or alternatively, makes a phonecall to the company in order to determine qualification. Then, when a case is in the *qualification stage*, a phonecall is initiated with the purpose of scheduling a meeting with the potential customer's representatives. If a meeting is scheduled, the opportunity enters the *presentation stage*. The goal of a sales person is to get the contract signed during the presentation. If she succeeds, the opportunity is marked as *won* and the case terminates. If the offer made during the meeting was acceptable, but the signing of the contract is postponed, the opportunity enters the *contract stage*. If the offer was not accepted during the meeting, an offer is sent via e-mail, and the opportunity moves to the *offer stage*. Any time during the process additional phonecalls can be made and follow-up meetings scheduled. When it becomes clear that the company is not interested in collaboration, the opportunity is marked as *lost*.

The number of potential customers is very high and it is not feasible for the sales people to deeply explore all of the possible leads. Thus, the process would benefit from a support system that estimates if an opportunity will likely end with a signed contract (opportunity won) or not (opportunity lost). If an opportunity is likely to be lost, the sales person can close it at an early stage (or assign it a lower priority), becoming able to focus on other leads with higher potential. Given this motivation, in the following experiments we aim at predicting, as early as possible, if an opportunity will be lost.

Table 1. Evaluation datasets

Data	# Normal cases	# Deviant cases	Avg. # words/doc	# Lemmas
DR	13608	417	11	11822
LtC	385	390	8	2588

In the debt recovery dataset, events are not explicitly logged. Instead, this information is captured in the *collector's notes*, which are written down in unstructured textual format. The collector's notes constitute a dynamic feature, which may describe the activity taken by the collection specialist, as well as the answer of the debtor and the assessment of the specialist. In the second dataset, the phonecall summaries are written down in unstructured format. The text in both datasets is written in Estonian language. Statistics about both datasets are given in Table 1.

Based on the structured data available, we identify 8 static and 69 dynamic features in the debt recovery dataset, and 3 static and 65 dynamic features in the lead-to-contract dataset. The static features are general statistics about the company, for instance, *the size of equity*, *the net profit*, and *field of activity*. The dynamic features in the first dataset are mostly related to the debt, e.g., *the number of days past due*, *the expected repayment amount until the next 7 days*, and *the sum of other debts of the debtor*. In the second dataset, the dynamic features include *activity name*, *resource*, and *expected revenue*. For both datasets, we use dynamic features that reflect the company's (either the debtor's or the potential customer's) risk score, calculated at 6 different months prior to the given event. Moreover, as the first dataset contains a considerable amount of missing values, additional 16 (static) features are added that express whether the value of a particular feature is present or missing. In the given datasets, we decide to use unstructured data as static information, i.e., to encode only the last available text, given a specific prefix length.

4.2 Research Questions and Evaluation Measures

In our evaluation, we address the following three research questions:

RQ1 Do the features derived from textual data (using different methods) increase the prediction *accuracy* of index-based sequence encoding?
RQ2 Do the features derived from textual data (using different methods) increase the prediction *earliness* of index-based sequence encoding?
RQ3 Is the proposed predictive monitoring framework *efficient*?

For evaluating prediction accuracy (**RQ1**) of our framework, we use *precision*, *recall*, and *F-score*, as suggested in [16]. We do not use *accuracy*, as it can lead to misleading results in case of imbalanced data [18]. Also, we do not report about *specificity*, as our main goal is to predict the positive class as accurately as possible. All metrics are based on the possible combinations of actual and predicted outcomes. True positives (TP) are positive cases, which are correctly predicted as positive. True negatives (TN) are negative cases, which are correctly predicted as negative. False positives (FP) are negative cases, which are incorrectly predicted as positives. False negatives (FN) are positive cases, which are incorrectly predicted as negatives. Given these notions, precision is defined as $TP/(TP+FP)$, recall as $TP/(TP+FN)$, and F-score as $2 \cdot precision \cdot recall/(precision + recall)$.

To answer **RQ2**, we measure the *earliness* of predictions [6]. Earliness is calculated for cases that are predicted as positive, as the ratio of *length of the* *case when the final prediction was made/total length of the case*. For instance, if the case was predicted as positive after 2 events, while the actual total length of the case was 8 events, $earliness = 0.25$. Low earliness values are better, as the aim of predictive monitoring is to provide predictions as early as possible.

Finally, the *computation time* is measured in order to estimate the efficiency of the framework (**RQ3**). For evaluating the offline component of the framework, we differentiate between the time for data processing (text model construction, textual feature extraction, and sequence encoding) and classifier training. Times are summed up over all prefix lengths, in order to evaluate the total time that is needed to set up the framework. For evaluating the online component, we combine the time for encoding the running case as a feature vector and the time for prediction. Times are averaged over the total number of processed events.

4.3 Evaluation Procedure

We split each dataset randomly in two parts, so that 4/5 of it is used for training the offline component, while the remaining 1/5 is used for testing the online component. For tuning the parameters of the text modeling methods, we perform a grid-search over all combinations of selected parameter values using 5-fold cross-validation on the training set. In the DC dataset, where only 3 % of cases are deviant, we use oversampling on the training data in order to alleviate the imbalance problem. The final Paragraph Vector models are trained for 10 epochs. The optimal parameters are chosen based on F-score, for each combination of text modeling method, classification method, and confidence threshold. The computation times are calculated as the average of 10 equivalent executions with $minConf = 0.6$. The probability estimates returned by the classifier are used as confidence values.

The optimal parameters found when using random forest and logistic regression are different. However, in the following, we discuss the values obtained using random forest only, since random forest performs better than logistic regression in all cases. We optimize the parameters described in Sect. 3 and use the default values for all the parameters not mentioned.

For the method based on the basic BoNG model, we explore 43 parameter settings (varying maximum *n-gram size*, *idf*, and *number of selected features*). In most cases, tf-idf weights perform slightly better than simple tf. Moreover, bigrams and trigrams gain similar performance, while both are better than unigrams. The best number of selected features stays between 100 and 1000. In the DR dataset, only 100 features are often sufficient to gain a good accuracy, while more features are needed in the LtC dataset (usually 750 or 1000).

For the method based on the BoNG model with NB log count ratios, we try 84 combinations of parameters (varying α, maximum *n-gram size* and *number of selected features*). Changing the α value has almost no effect on the results, usually a small value (either 0.01 or 0.1) is chosen. The best number of selected features tends to be higher than in the BoNG case, usually between 250 and 1000 features in the DR dataset and between 1000 and 5000 in the LtC dataset. In most cases, trigrams outperform bi- and unigrams.

In case of LDA (we vary the *number of topics* and *idf*), we try 6 different numbers of topics (12 combinations in total). In general, the larger the confidence, the higher the number of topics that achieves the best results. In the DR dataset, idf normalization does not improve the outcome, while changing the parameters has very little effect on the results in the LtC dataset.

For PV, we explore 91 combinations, varying the *size of the feature vector* and the *window size*. The best results are obtained with a small 10- or 25-dimensional vector. The optimal window size varies a lot across the experiments, but stays between 5 and 9, in general.

Experiments were run in Python 3.5 using scikit-learn (BoNG and classifiers), gensim (LDA and PV) and estnltk (lemmatization) libraries on a single core of a 64-bit 2.3 GHz AMD Opteron Processor 6376 with 378 GB of RAM.

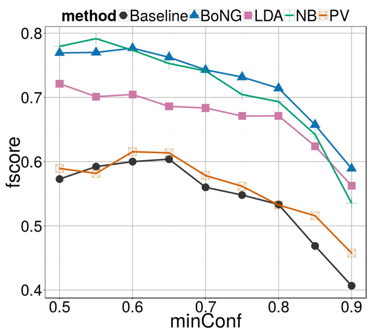
4.4 Results

The F-scores of the random forest classifiers are shown in Fig. 3a (debt recovery dataset) and c (lead-to-contract dataset). We observe that in both datasets, the methods that utilize unstructured data almost always outperform the baseline. In the DR dataset, BoNG and NB achieve considerably better results than the other methods, while in the LtC dataset, the best results are produced by LDA. Although the proportion of unstructured vs. structured data in the LtC dataset is much smaller than in the DR dataset, the improvement of the results is still substantial. The highest F-score in the DR dataset (0.791) is achieved by NB with $minConf = 0.55$, while LDA achieves F-score of 0.753 with $minConf = 0.65$ in the LtC dataset.

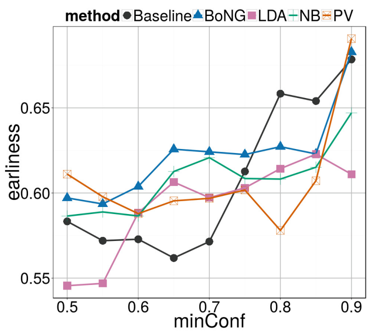
Figure 3b and d show the prediction earliness achieved with random forest. The model with the best F-score in the DR dataset tends to make predictions when 59 % of a case has finished, while the best model in the LtC dataset is predicted after 40 % of a case has been seen.

In order to further explore the importance of unstructured data in making predictions, we performed additional experiments using unstructured data only. In the DR dataset, the NB model achieves F-score of 0.66 (instead of 0.791 as in Fig. 3a), while in the LtC dataset, the LDA model reaches F-score of 0.70 (instead of 0.75 as in Fig. 3c). In both datasets, the model trained with only structured data (the baseline) outperforms all unstructured data models in terms of precision, while falls behind in terms of recall. Thus, unstructured features have some predictive power on their own, but in order to get the most out of the data, they should be combined with structured data. In addition, we observe that using the best model (NB, conf = 0.55) of the DR dataset, 3 out of the top 5 features ranked according to Gini impurity are derived from textual data. On the other hand, in the LtC dataset (LDA, conf = 0.65), the first 9 features according to importance are structured features. This implies that in best model of the LtC dataset, textual features are less relevant than structured features.

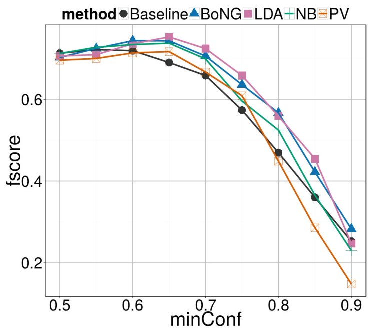
Table 2 reports the computation time required by the offline component for data processing and for classification, as well as the computation time required by the online component for providing a prediction with a minimum confidence



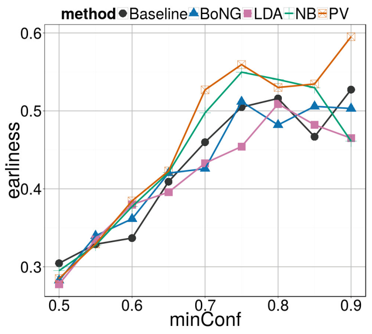
(a) F-score in DR dataset



(b) Earliness in DR dataset



(c) F-score in LtC dataset



(d) Earliness in LtC dataset

Fig. 3. Predictive monitoring results with random forest

of 0.6. The most expensive technique, in terms of computation time for setting up the offline component, is LDA, which requires more than 4 min for data processing in the DR dataset and 28 s in the LtC dataset (Table 2). The difference between the time required by the two datasets is likely due to their different size. In case of PV, the processing time of the offline component depends highly on the number of epochs used for training the paragraph vectors. In our experiments, we used 10 epochs, which results in relatively high processing time. BoNG is the most efficient method, taking only little over a second in the smaller dataset and over 5 s in the larger one. On the other hand, the current implementation of NB does not scale well as the size of the data increases.

Classifier training times remain between 24 and 83 s, depending on the dataset size and number of features. In the online component, all the methods are extremely fast in processing a running case (in the order of milliseconds per event). The slowest is LDA which takes 7 ms on average in the DR dataset.

Depending on the application, some additional time may be needed to prepare the data into a suitable format. Preprocessing the entire dataset took 2.3 min in case of DR and 14 s in case of LtC. The most time-consuming procedure was lemmatization that took 1.5 min in DR (12 s in LtC).

Table 2. Computation times, minConf = 0.6

	total_proc_offline (s)					total_cls_offline (s)					avg_online (ms)				
Data	Base	BoNG	NB	LDA	PV	Base	BoNG	NB	LDA	PV	Base	BoNG	NB	LDA	PV
DR	0.5	5.1	54.0	262	212	41.3	50.0	53.9	83.6	61.3	0.1	0.4	2.9	7.0	2.0
LtC	0.5	1.4	1.7	28.0	14.7	28.1	29.9	35.2	24.5	27.3	0.3	0.4	0.5	0.7	0.5

We also ran the same experiments with logistic regression instead of random forest. We omit these results since logistic regression performed worse in all cases. A possible explanation is that the dataset contains both sparse (textual features in case of BoNG and NB) and dense features (structured data payload), and the dense features carry substantial predictive power. Logistic regression is generally more suitable for sparse data.

4.5 Discussion

According to our results, BoNG performs well on both datasets over all confidence thresholds. This indicates that there exists a set of n-grams that carry enough information to classify cases. NB is able to outperform BoNG in a few cases, but the implementation is not as scalable.

In the LtC dataset, the best results are produced by LDA. The reason for this might be that LDA combines the information captured in textual data into topics, instead of using specific words. Thus, it is able to perform well even in the case of few available textual data, which is the case in the LtC dataset. Also, supported by previous studies where topic modeling methods have shown to perform well on short texts, such as tweets [10], LDA is less affected by the fact that individual notes in the LtC data set contain only 8 words on average.

A possible reason for PV performing worse than the other methods is that PV computes the feature vector for an unseen document via inference. Therefore, in order to produce reliable results, it requires a fairly large document collection for training. Moreover, the benefits of PV become more evident in heterogenous datasets, where a variety of words is used to express similar concepts.

One limitation of our evaluation is its low generalizability. While the evaluation datasets come from two real-life processes with different deviant case ratios (balanced vs. imbalanced), the textual notes in both datasets are written by members of a small team of debt recovery specialists and salespeople respectively. The observations might be different if these notes were written by a larger team or if they included emails sent by customers (higher heterogeneity). Also, the results may be affected by the amount of textual data available. Another limitation is the reduced set of classification algorithms employed (random forest and logistic regression). While these algorithms are representative and widely used in text mining, other classifiers might be equally or more suitable.

5 Related Work

Predictive monitoring is relevant in a range of domains where actors are interested in understanding the future performance of a system in order to take preventive measures. Predictive monitoring applications can be found in a wide range of settings, including for example industrial processes [11] and medical diagnosis [4]. One recurrent task addressed in this field is that of failure prediction [18] – i.e., detecting that a given type of failure will occur in the near-term.

While the predictive monitoring problems addressed in the above fields share common traits with the problem addressed in this paper, business process event logs have a specific characteristics that call for specialized predictive monitoring methods, chiefly: (i) business process event logs are structured into cases and each case can have a different outcome; hence, the problem is that of monitoring multiple concurrent streams of events rather than one; (ii) every event in a case refers to a given activity or external stimulus; (iii) every event has a payload; (iv) the payload may contain both structured data and text, and the structured part of the data includes both discrete and numerical attributes. In contrast, in other application domains [4,11,18], events in a given stream are generally of homogeneous types and carry numerical attributes (e.g., measurements taken by a device), this requiring a different type of techniques compared to predictive business process monitoring.

A range of methods have been proposed in the literature to deal with this specific combination of characteristics. These methods differ in terms of the object of prediction, the type of data employed, and the approach used for feature encoding. With respect to the former, some approaches focus on predicting time or other performance measures. For example, [17] uses stochastic Petri nets to predict the remaining execution time of a case, while [16] addresses the problem of predicting process performance violations in general and deadline violations in particular. Other approaches focus on predicting the outcome of a process, such as predicting failures or other types of negative outcomes (a.k.a. *deviance*). For example, [5] presents a technique to predict risks, while [15] focuses on predicting binary outcomes (normal vs. deviant cases).

Predictive process monitoring approaches also differ depending on the type of data they use. Some approaches only use control-flow data [16,17], others use control-flow and structured data [5,8,13,15]. When building predictive process monitoring models that take into account both control-flow and data payloads, a key issue is how to encode a given trace in the log (or a prefix thereof) as a feature vector. In this respect, a comparison feature encoding approaches is given in [13], which empirically shows that an index-based encoding approach provides higher performance.

None of the above studies have taken into account textual data. Yet, textual data is generated in a range of customer-facing processes and as shown in this paper, can enhance the performance of predictive process monitoring models.

6 Conclusion

We outlined a framework for predictive process monitoring that combines text mining methods to extract features from textual documents, with (early) sequence classification techniques designed for structured data. We studied different combinations of text mining and classification techniques and evaluated them on two datasets pertaining to a debt recovery process and a sales process.

In the reported evaluation, BoNG and NB, in combination with random forest, outperform other techniques when the amount of textual data is sufficiently large. In the presence of a smaller document collection, LDA exhibits better performance. An avenue for future work is to further validate these observations on other datasets exhibiting different characteristics, for example, datasets containing longer or more heterogeneous documents. Another future work avenue is to produce interpretable explanations of the predictions made, so that process workers and analysts can understand the reasons why a given case is likely to end up with a given outcome. Last but not least, we are planning to integrate our tool in the operational support of the process mining tool ProM to provide predictions starting from an online stream of events.

Acknowledgments. This research is funded by the EU FP7 Programme (project SO-PC-Pro) and by the Estonian Research Council and by ERDF via the Software Technology and Applications Competence Centre (STACC).

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
2. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
3. Castellanos, M., Casati, F., Dayal, U., Shan, M.: A comprehensive and automated approach to intelligent business processes execution analysis. Distrib. Parallel Databases **16**(3), 239–273 (2004)
4. Clifton, L.A., Clifton, D.A., Pimentel, M.A.F., Watkinson, P., Tarassenko, L.: Predictive monitoring of mobile patients by combining clinical observations with data from wearable sensors. IEEE J. Biomed. Health Inf. **18**(3), 722–730 (2014)
5. Conforti, R., de Leoni, M., Rosa, M.L., van der Aalst, W.M.P., ter Hofstede, A.H.M.: A recommendation system for predicting risks across multiple business process instances. Decis. Support Syst. **69**, 1–19 (2015)
6. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-Based Predictive Process Monitoring. arXiv preprint (2015)
7. Dumas, M., La Rosa, M., Mendling, J., Reijers, H.A.: Fundamentals of Business Process Management. Springer, Heidelberg (2013)
8. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012)
9. Freedman, D.: Statistical Models: Theory and Practice. Cambridge University Press, Cambridge (2005)

10. Hong, L., Davison, B.D.: Empirical study of topic modeling in Twitter. In: Proceedings of the First Workshop on Social Media Analytics, pp. 80–88. ACM (2010)
11. Juriceka, B.C., Seborga, D.E., Larimore, W.E.: Predictive monitoring for abnormal situation management. J. Process Control **11**(2), 111–128 (2001)
12. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053 (2014)
13. Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M.: Complex symbolic sequence encodings for predictive monitoring of business processes. In: Motahari-Nezhad, H.R., Recker, J., Weidlich, M. (eds.) BPM 2015. LNCS, vol. 9253, pp. 297–313. Springer, Switzerland (2015)
14. Lessmann, S., Baesens, B., Seow, H.V., Thomas, L.C.: Benchmarking state-of-the-art classification algorithms for credit scoring: an update of research. Eur. J. Oper. Res. **247**(1), 124–136 (2015)
15. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Heidelberg (2014)
16. Metzger, A., Leitner, P., Ivanovic, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., Pohl, K.: Comparing and combining predictive business process monitoring techniques. IEEE Trans. SMC **45**(2), 276–290 (2015)
17. Rogge-Solti, A., Weske, M.: Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays. In: Basu, S., Pautasso, C., Zhang, L., Fu, X. (eds.) ICSOC 2013. LNCS, vol. 8274, pp. 389–403. Springer, Heidelberg (2013)
18. Salfner, F., Lenk, M., Malek, M.: A survey of online failure prediction methods. ACM Comput. Surv. (CSUR) **42**(3), 10 (2010)
19. Wang, S., Manning, C.D.: Baselines and bigrams: simple, good sentiment and topic classification. In: Annual Meeting of the Association for Computational Linguistics, pp. 90–94 (2012)