A next click recommender system for web-based service analytics with context-aware LSTMs

Sven Weinzierl
FAU Erlangen-Nürnberg
sven.weinzierl@fau.de

Matthias Stierle
FAU Erlangen-Nürnberg
matthias.stierle@fau.de

Sandra Zilker
FAU Erlangen-Nürnberg
sandra.zilker@fau.de

Martin Matzner
FAU Erlangen-Nürnberg
martin.matzner@fau.de

Abstract

Software companies that offer web-based services instead of local installations can record the user's interactions with the system from a distance. This data can be analyzed and subsequently improved or extended. A recommender system that guides users through a business process by suggesting next clicks can help to improve user satisfaction, and hence service quality and can reduce support costs. We present a technique for a next click recommender system. Our approach is adapted from the predictive process monitoring domain that is based on long short-term memory (LSTM) neural networks. We compare three different configurations of the LSTM technique: LSTM without context, LSTM with context, and LSTM with embedded context. The technique was evaluated with a real-life data set from a financial software provider. We used a hidden Markov model (HMM) as the baseline. The configuration LSTM with embedded context achieved a significantly higher accuracy and the lowest standard deviation.

1. Introduction

A key focus of services is the perceived quality by the users, which ultimately determines the service quality [1]. For software services, end-users are mostly looking for functionality and non-functional aspects like stability, performance, but most importantly, usability [2]. While the latter is mostly given for modern software services primarily in the consumer business, enterprise software services are often still driven by functionality and are facing challenges in adapting user interfaces due to the large existing user base. Therefore, solutions for improving usability are required that extend existing user interfaces.

Fromm et al. consider *service usage analytics* as the most mature type of service analytics as it requires advanced analytical capabilities as well as customer data which used to be hard to obtain [3]. In recent years, an increasing number of software providers offered their products as a cloud-based service instead of a locally installed software, which enables them to track data about the usage of their services [4].

With the ongoing popularity of machine learning and artificial intelligence, a multitude of techniques and approaches were created that can also be applied to service usage analytics. In order to support users with handling complex and less intuitive software, next likely click items (e.g., buttons or text fields) could be suggested by a recommender system. Recommender systems are software tools and techniques providing suggestions for items (e.g., next click items[1]) to be of use to a user [5].

An exemplary use case would be the processing of a sales order depicted in Figure 1. After order entry, the user has to decide how to process it further. The interface contains an overwhelming amount of information and offers several options for the next clicks, like checking the credibility of the customer or confirming the order to the customer. In a separate box, a recommendation is shown. It suggests the click, which is most likely performed by users at this stage.

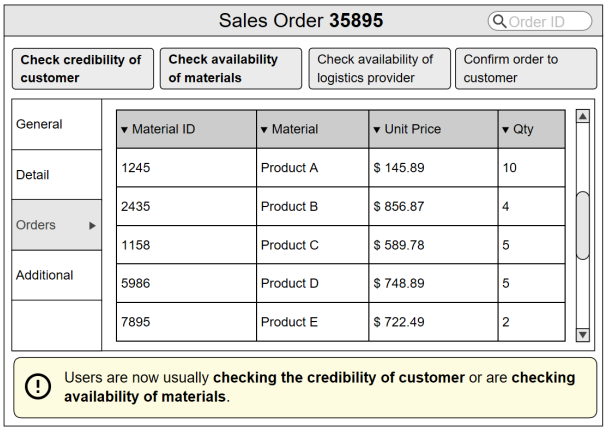

Figure 1. Exemplary use case of a recommender system for a sales order process.

[1]Note we use the term "clicks" for "click items".

URI: https://hdl.handle.net/10125/63929
978-0-9981331-3-3

However, for such a recommender system to be useful, the quality of the predictions has to be high. Given the great variety in user interactions, including session duration for web applications, this can be a challenging task that cannot be solved by applying basic statistical or time series methods.

We suggest the use of techniques from the domain of predictive process monitoring which are built with the design goal of considering the sequential flow of the incoming process data (*process awareness* [6]). To learn predictive models for the next click prediction, we use long short-term memory (LSTM) neural networks [7] and fed the context attributes as well as the process flow attributes in the form of a vector-oriented representation to the LSTMs. Further, the context of a business process consists of context attributes that characterize the environment in which the process is running [8]. For example, the domain of a web service. Regarding the context, we make use of *paragraph2vec* embedding to capture dependencies between the context attributes. According to De Koninck et al., this has never been done for a LSTM-based technique from predictive process monitoring [9]. Against this background, our research goal is to design a *technique for a next click recommender system with context-aware LSTMs.* We contribute to both academia and practice in two ways by providing a context-aware technique for the next click prediction that is evaluated with a real-life web usage log. First, the technique is designed to work with three different configurations, of which one is embedding-based. Second, we present a new use case for analytics-based services with the characteristics displayed in Table 1.

2. Related work

In this paper, we mainly deal with two streams of research that are currently converging: web usage and process mining. Process mining is a technology for analyzing business processes from process data [11]. This data, which is usually referred to as event logs, is commonly obtained from enterprise information systems like SAP ERP. Web usage mining is a research field that emerged at the end of the 90s driven by the increased number of websites and in particular, webshops [12].

We focus on how to apply techniques from the domain of process mining to clickstream data. As said, the strong point of process mining techniques is their process-awareness. That means recognizing the sequential dependencies between events in the data that belong to the same instance (i.e., a customer visit on a website). In web usage mining, techniques for predicting usage paths for users are surprisingly sparse. This becomes evident when looking at the predictive analytic capabilities of the most prominent commercial systems. For example, the path analysis tool from Google analytics[2] or from Adobe analytics[3] allow only a purely descriptive analysis of the paths and therefore these tools have a deficit in predicting a user's future software behavior [13, 14].

There have been several works applying process discovery - that is, discovering process models from data - to clickstream data [15, 16]. In this work, however, we will adopt a technique from predictive process monitoring which is a subbranch from process mining [6]. It deals with making predictions about the next process steps or other process outcomes like duration from process data. Again, a key differentiator from other techniques is the process-awareness meaning that the predictive model "exploits an explicit representation of the process model to make the prediction" [6].

Recommender systems in the domain of web usage mining are usually focused on improving customer conversion by suggesting items such as products or content (e.g., videos or songs) [17]. In general, these items are not sequential. For realizing such a recommender system, there are different kinds of approaches like collaborative or content-based filtering [18]. However, because of the sequential nature of recommending the next clicks, these approaches are not applicable.

The first work to use recurrent neural networks (RNNs) for predicting next clicked items mainly evolves around the issue of missing user-profiles [19] which is not a common issue within the context of enterprise clickstream data, and they do not apply LSTMs. Pardos et al. also make use of LSTMs, but they only predict the next URL and not individual clicks or interactions [20].

3. Research method

This research follows the design science paradigm, and the structure of this paper is based on the design science publication schema that Gregor and Hevner propose [21]. Our primary purpose is to design a technique for a next click recommender system with context-aware LSTMs which is our artifact. The artifact consists of two components: an offline and an online component.

We first presented related work on web usage mining and process mining for problem identification. Then we describe the design of the artifact, dealing with

[2] https://support.google.com/analytics/answer/1713056?hl=en (retrieved on: 31.08.2019).

[3] https://docs.adobe.com/content/help/en/analytics/analyze/ad-hoc-analysis/c-reports-paths.html (retrieved on: 31.08.2019).

Table 1. Classification of our case following the taxonomy (based on Hunke et al. [10]).

Dimension	Characteristics					E/N
Data Generator	Customer	Non-customer		Processes	Objects	N
Data Target	Internal			External		N
Data Origin	Customer	Non-customer	Processes	Objects	Environment	N
Analytics Type	Descriptive	Diagnostic	Predictive	Prescriptive		E
Portfolio Integration	Stand-alone solution	Wrapped around product		Wrapped around service		E
Service User Role	Recipient	Provider		Interactor		E

an LSTM-based approach for a recommender system. Based on that, we consider three different configurations consisting of the two components. Next, we evaluate the artifact in regards to its predictive quality with a real-life web usage log from a financial service provider. We benchmark the obtained results with a hidden Markov model (HMM). Finally, we sum up our contributions, outline limitations, and describe future research needs.

4. Towards an LSTM-based next click predictor

In this section, we describe an approach for the next click prediction in the context of the web usage mining domain. The approach is composed of an offline and an online component. In the offline component, an LSTM-based predictor is learned from a web usage log. In the online component, the learned LSTM-based predictor is applied to predict the next click of a running web session. Before we specify these two components, we define the underlying problem.

4.1. Problem statement

The next click recommendation system receives as input a web usage log $\mathcal{L}$[4], which is a multi-set of web sessions $\mathcal{L} = \{ws_1, ws_2, \ldots, ws_l\}$, where $l = |\mathcal{L}|$. A web session represents the execution of a web service and can be depicted as a sequence of clicks with additional attributes. Each click of a web session and its attributes represent a feature vector:

$$ws_i = \langle\langle c_1^{(i)}, a_{1,1}^{(i)}, \ldots, a_{1,m_1}^{(i)}\rangle, \langle c_2^{(i)}, a_{2,1}^{(i)}, \ldots, a_{2,m_2}^{(i)}\rangle, \ldots, \langle c_n^{(i)}, a_{n,1}^{(i)}, \ldots, a_{n,m_n}^{(i)}\rangle\rangle, \quad (1)$$

where $c_j^{(i)}$ for $1 \leq j \leq n$ represents the n clicks of web session ws_i and $a_{j,m_j}^{(i)}$ the attributes associated to click j. The number of attributes associated with a click and the number of clicks per web session can vary. Context elements are attributes in this formalization. Every web session has a *prefix* (length k of a web session) and a *suffix* (remaining part of a web session). For example, the prefix of length 3 of the web session w_1 (disregarding context attributes) $\langle c_1, c_2, c_3, c_4\rangle$ is $\langle c_1, c_2, c_3\rangle$ and the suffix is $\langle c_4\rangle$. With regard to the next click prediction, the length of the suffix is always 1. For describing the artifact in the next chapter, consider the web session ws_1 with prefix length 3 as an initial example, where the clicks within a web session are ordered by their associated time step attribute.

$$ws_1 = \langle(\text{"Click 1"}), (\text{"Click 3"}), (\text{"Click 7"})\rangle. \quad (2)$$

To complete this example, the suffix $\langle c_4\rangle$ of ws_1 has the attribute value "Click 10".

4.2. Offline component: three configurations to learn an LSTM

Figure 2 depicts a schema of the three different configurations to learn an LSTM-based predictor for the next click prediction. In the first configuration (*LSTM without context*), only the click sequence of the web usage log (control-flow information) is considered for learning, whereas in the second (*LSTM with context*) and in the third approach (*LSTM with embedded context*) additional context information is used for learning an LSTM. Each configuration consists of two steps - a pre-processing step of the web usage log depending on the received input data and an ensuing learning step.

4.2.1. Pre-processing The three configurations considered in this paper are built upon each other. Therefore, we start by explaining the pre-processing step for the simplest one, the LSTM without context.

LSTM without context. In this configuration, the offline component receives as input a web usage log $\mathcal{L}$, where each ws is only described by its click attributes. Values of the click attribute are transformed into numeric ones because an LSTM performs numerical calculations (in most cases with stochastic gradient

[4]We use our own representation for describing the problem statement.

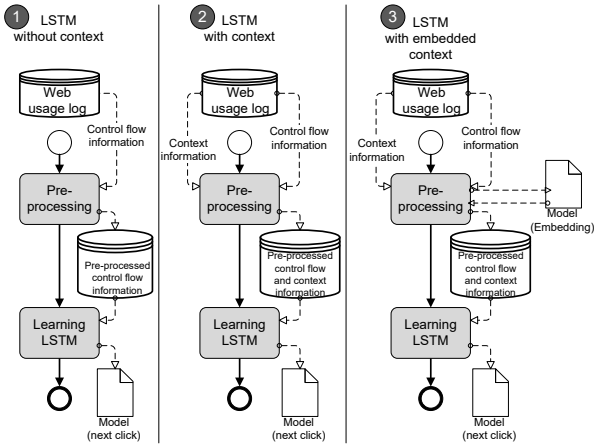

Figure 2. Three configurations of the offline component.

descent) during the learning phase. To achieve that, we apply a one-hot encoding for the click attribute values. Based on our exemplary web session ws_1, we have four different click values and the one-hot encoding for the first click attribute value "Click 1" is (0,0,0,1). After the numeric transformation, ws_1 from (2) assumes the representation in (3).

$$ws_1 = \langle (0,0,0,1), (0,0,1,0), (0,1,0,0) \rangle. \quad (3)$$

Each prefix of the pre-processed web session ws_1, as represented in (3), is then transferred to the LSTM.

LSTM with context. In the second configuration, the offline component receives as input a web usage log $\mathcal{L}$ with context attributes. As the click values in the previous section, the values of the context attributes must also be transformed into numeric ones. We apply an integer-mapping for context attributes with discrete values. The first value of such an attribute is mapped to 0, the second one is mapped to 2, and so on. Missing values are replaced by a dummy value. Further, not to implicitly give one attribute more relevance than another, we applied for each numerical context attribute with a value range not equal to $[0, 1]$ a min-max normalization to grant equality among the attributes. Missing values of a numerical attribute are previously replaced by the mean of the existing values.

In order to continue with our exemplary web session ws_1, we add the context attributes *User* with the values "User A", "User B" and "User C" for the three clicks and *Browser of User* with the three values "IE, "IE" and "Chrome" to it. After the numerical transformation of the context attributes, w_1 from (3) is represented as in (4), where the values of the context attributes are highlighted in bold.

$$\begin{aligned} ws_1 = \langle &(0, 0, 0, 1, \mathbf{0}, \mathbf{0}), \\ &(0, 0, 1, 0, \mathbf{1}, \mathbf{0}), \\ &(0, 1, 0, 0, \mathbf{2}, \mathbf{1}) \rangle. \end{aligned} \quad (4)$$

Finally, each prefix of the pre-processed web session ws_1, as represented in (4), is then transferred to the LSTM.

LSTM with embedded context. In the last configuration, the offline component receives as input a web usage log $\mathcal{L}$ with embedded context attributes. An embedding of the context attributes enables to learn and finally represent in the form of a low-level dimensional vector semantic similarities between these attributes. To embed the values of more than one context attributes per click, we use the *paragrah2vec* embedding technique *distributed bag of words* [22]. Note each context attribute from the previous configuration can be used for this embedding technique. Further, this technique uses a shallow neural network that performs a stochastic gradient descent to learn a vector representation. In detail, at each iteration of the stochastic gradient descent, a window is sampled, then a random context attribute value is sampled from the window, and a classification task is formed given the paragraph vector (all context attribute values for a click). If the learning converges, attribute values with similar meaning are mapped to a similar position in the vector space. After the embedding of the context attributes to a vector with size 4, w_1 from (4) is represented as in (5), where the values of the context attributes are highlighted in bold.

$$\begin{aligned} ws_1 = \langle &(0, 0, 0, 1, \mathbf{0.09}, \mathbf{0.12}, \mathbf{-0.13}, \mathbf{-0.15}), \\ &(0, 0, 1, 0, \mathbf{0.17}, \mathbf{0.18}, \mathbf{-0.01}, \mathbf{-0.03}), \\ &(0, 1, 0, 0, \mathbf{0.30}, \mathbf{0.31}, \mathbf{0.27}, \mathbf{0.35}) \rangle. \end{aligned} \quad (5)$$

Based on the representation of ws_1 in (5), values of context attributes for "Click 1" are more similar to the values of the context attributes for "Click 3". Therefore, we suppose a higher dependence between "Click 1" and "Click 3" and a lower dependence between "Click 1" and "Click 7" from a context perspective. Eventually, each prefix of the pre-processed web session ws_1, as represented in (5), is then fed to the LSTM.

4.2.2. Learning an LSTM In the context of web usage mining, we use the basic architecture variant of LSTMs [7] (called vanilla LSTMs) for learning the next click predictor for four reasons. First, LSTMs are recurrent neural networks (RNNs) that are designed to

handle temporal dependencies in sequential prediction problems [23]. Second, LSTMs are resistant to the vanishing or exploding gradient problem in the case of long sequences [24]. Third, vanilla LSTMs learn predictive models with good predictive quality on different data sets [25]. Fourth, first works in the field of predictive process monitoring have shown (e.g., [26, 27]) that vanilla LSTMs can successfully learn predictive models from context data. At this point, it should be mentioned that other LSTM architecture variants exist that are geared towards (at least to a certain degree) other criteria (e.g., the gated recurrent unit [28] with a simpler RNN architecture for reaching a shorter training time). In addition, since the focus of our work is on predictive business process monitoring as a sub-field of process mining [29] techniques from other fields (e.g., sequential pattern mining [30]) are not considered.

Further, a vanilla LSTM is an extension of the vanilla RNN. The vanilla RNN is the most fundamental architecture type of RNNs. In such an RNN, each hidden layer consists of recurrent artificial neurons (RNN units), i.e., neurons with a feedback connection. This connection enables RNNs to store representations of past clicks. RNNs can be unfolded, i.e., there are equivalent copies of the network, one copy for each time slice and each copy passing a message to the next one. Figure 3 depicts an RNN with one unit and its unfolded version.

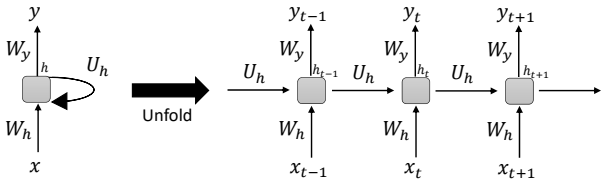

Figure 3. Vanilla RNN with one RNN unit (based on LeCun et al. [23]).

More formally, at each time slice t, an RNN unit takes the vector of its previous hidden state h_{t-1} as well as its current input vector x_t as input and then outputs the vector of a new hidden state h_t. The hidden state vector h_t can be calculated as $h_t = \sigma_h(U_h h_{t-1} + W_h x_t + b_h)$, where σ_h is the activation function (often a sigmoid function or other nonlinear function like hyperbolic tangent or rectified linear function) applied element-wise, W_h and U_h are the weight vectors and b_h is the bias. Based on h_t, a third weight vector W_y and a possible activation function σ_y calculate the output vector y_t as $\sigma_y(W_y h_t + b_y)$. The weight vectors W_h,U_h and W_y as well as the biases b_h and b_y are parameters to be learned.

However, vanilla RNNs can not learn from long sequences, because of the vanishing or exploding gradient problem [24]. To remedy this problem, vanilla LSTM cells introduce next to the state vector h_t an internal memory in the form of a vector (c_t) and different gates to control this memory vector. In general, c_t and h_t can be calculated as:

$$\begin{aligned} c_t &= f_t * c_{t-1} + i_t * g_t, \\ h_t &= o_t * \tanh(c_t), \end{aligned} \tag{6}$$

where, f_t, i_t, o_t and $g_t \in (0, 1)$ are the gates and $*$ describes the point-wise multiplication. f_t (forget gate) determines how much of the previous memory is kept, i_t (input gate) controls how much new information is stored into memory, g_t (gate gate or candidate memory) defines how much information is stored into memory and o_t (output gate) determines how much information is read out of the memory. Typically, the gates are parameterized as:

$$\begin{aligned} f_t &= sigmoid(U_f * h_{t-1} + W_f * x_t + b_f), \\ i_t &= sigmoid(U_i * h_{t-1} + W_i * x_t + b_i), \\ o_t &= sigmoid(U_o * h_{t-1} + W_o * x_t + b_o), \\ g_t &= sigmoid(U_g * h_{t-1} + W_g * x_t + b_g), \end{aligned} \tag{7}$$

where the weights $U_{f,i,o,g}$ and $W_{f,i,o,g}$ as well as the biases $b_{f,i,o,g}$ are parameters to be learned.

4.3. Online component: apply an LSTM for the next click prediction

The online component receives a running web session, takes as input the learned model from the offline component, and outputs a prediction of the next click. Figure 4 depicts the schema of the online component for the three configurations proposed in chapter 4.2.

For a running web session with a length > 1, the two steps - pre-processing and prediction - are performed according to the respective configuration. In line with Tax et al., we only consider running web sessions with a length of > 1 for the next click prediction, since for a running web session with a length of 1 there is insufficient data available to base the prediction upon [31]. The pre-processing for each configuration takes place as already described in the offline component. Subsequently, in the prediction step, a pre-processed web session is fed to the LSTM model, and the LSTM model outputs a probability distribution over all possible next clicks in the form of an output vector o_t, where t is the current time step. Finally, the click with the highest probability in o_t is determined as the next click.

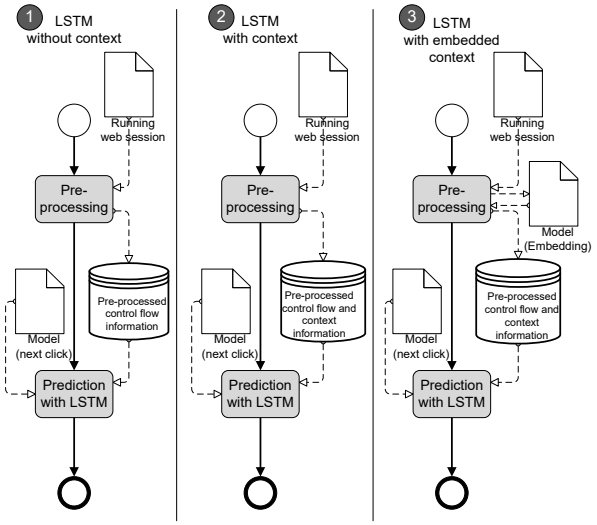

Figure 4. The configurations of the online component.

5. Evaluation

In this section, we provide an evaluation of the three proposed configurations in terms of predictive quality for the next click prediction. We used a real-life web usage log from a software service provider in the finance sector.

5.1. Baseline

To evaluate the usefulness of our technique, we compare the results with an HMM. HMMs are popular for sequence prediction, e.g., in the field of grammatical inference. For the calculation of our baseline, we make use of an implementation developed for next event prediction with dynamic Bayesian networks [32] which can be found on GitHub[5].

5.2. Web usage log

For the evaluation, we used a real-life web usage log from a financial service provider. In the case study, the usage behavior of the customers was tracked. Therefore, the log consists of a case ID, click ID, time stamp, and six context attributes. The characteristics of these variables are shown in Table 2. The data set includes 2,142 sequences with a maximum length of 247 clicks. Figure 5 shows the distribution of the sequence lengths. It can be seen that the web usage log includes a lot of shorter sequences than longer ones, with a mean of eleven clicks and a 90% quantile of 93 clicks.

[5]The source code of our baseline is available on GitHub at https://github.com/fau-is/coppa-matlab.

Table 2. Characteristics of the web usage log.

# sequences	2,142
sequence lengths	1-247
median sequence length	11
# click types	161
# context attributes	6
context attribute sizes	[16, 44, 120, 222, 154, 46]

The web usage log covers 161 different possible click types. Whereas, 23 different click types perform 90% of all executed clicks. This imbalance is due to the fact that the evaluation is based on a real-life web usage log. Here certain types of clicks, e.g., ”Open Browser” or ”Close Browser” are made in almost every sequence of the process.

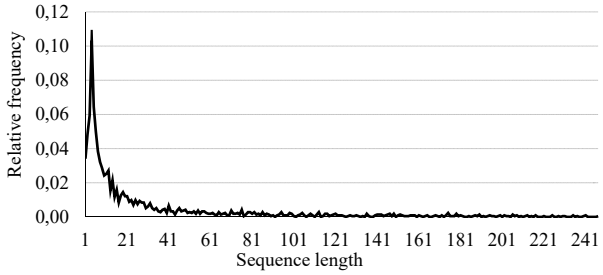

Figure 5. Distribution of sequence lengths.

The data contains cycles, i.e., recurrent clicks, created if the user clicks on one button multiple times in a row. This can be on purpose or when required by the process, e.g., pressing the save-button after entering date in each field of a form. On the other hand, recurrent clicks can be unintentionally when double-clicking a button. These recurrent clicks have a different timestamp, different context attributes, or both.

5.3. Evaluation procedure

To evaluate how well the three configurations of our approach work, we compared them on the real-life web usage log described in the previous section.

First, we used a ten-fold cross-validation with a web-session-based random sampling for each configuration as well as for the baseline to avoid an overfitted learning by the learning algorithms. Note in predictive process monitoring, there are two essential sampling techniques case-based and event-based sampling [33]. In this paper, we applied the case-based sampling technique for web sessions since we want to preserve the sequential structure concerning the LSTM. Also, this technique is more realistic in terms of the next click prediction task. Furthermore, in each of the ten iterations non-overlapping 10% of the web

usage log data are used for testing (application of the online component to predict the next click). The remaining 90% of each iteration are used for training (application of the offline component to learn the next click predictor).

In the training phase of the LSTMs batch normalization with a batch size of 256 was applied, where gradients are updated after each 256th web session prefix of the training set. For the learning phase of the LSTM, we set the hyperparameters *number of epochs* to 100 and *number of hidden layers* to 1. With regard to the single LSTM layer, the activation function *tanh* is set to ensure convergence at long web sessions, and a drop-out rate of 0.2 is configured to avoid overfitting. Finally, the *Nadam* [34] optimization algorithm was used in each LSTM model with *categorial cross-entropy loss*. For other parameters of this optimization algorithm (e.g., learning rate), the default values were used. Further, the shallow neural network used for training the embeddings of the context attributes has also configurable hyperparameters. For this neural network we set the *number of epochs* to 10, the *learning rate* to 0.025 with a decrease of 0.0025 per epoch, the *size of output feature vector* to 16, the *window size* to 5, and the *minimum count* to 1.

After learning the predictors, we used the online component on the test set for predicting the next click. We used the *accuracy* as an evaluation measure since we are interested in the proportion of correct predictions. For that, we calculated the *mean* as well as the *standard deviation* over the ten obtained accuracy values.

Experiments were run on a workstation with 12 CPUs and 64 GB RAM. In detail, the approaches were fully implemented in Python 3.7. Tensorflow (GPU) 1.13.1 [35] was used for constructing all LSTM models and embeddings were implemented based on Gensim [36]. Finally, the source code of the LSTMs is available on GitHub[6].

5.4. Results

We calculated both the average accuracy of the ten folds and the standard deviation for all approaches. The results of our evaluation can be found in Table 3.

Compared to the HMM, we could achieve much higher values for the average accuracy and much lower values for the standard deviation for each LSTM configuration. The configuration with context could achieve higher average accuracy and a lower standard deviation than the configuration without context. However, for the LSTM with an embedded context, we could achieve the highest average accuracy with an especially low standard deviation. Figure 6 shows the accuracy value for each fold per configuration. The LSTM with context information reaches a higher accuracy for nine out of ten folds. However, with the LSTM and embedded context information, we got the highest values for nine out of ten folds.

[6]The source code of the LSTMs is available on GitHub at https://github.com/fau-is/hicss2020-service-analytics.

Table 3. Average accuracy and standard deviation of the ten folds for each approach.

	Avg. acc.	Std. dev.
HMM (25 hidden states)	0.351207	0.020517
LSTM	0.603680	0.004378
LSTM + context	0.607522	0.004975
LSTM + embedded context	0.615279	0.003913

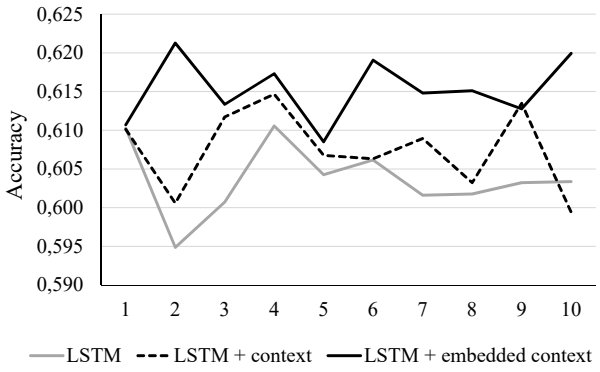

Figure 6. Accuracy value for each fold per configuration.

In line with Tama and Comuzzi [33], we conducted a significance test to analyze whether the accuracy values for the three configurations are significantly different from each other. Note the baseline was not included in this significance test since it is obvious that each LSTM-based configuration outperforms the baseline. First, we show with a Friedman test [37] that there is a significant difference between the ten accuracy values for each of the three configurations on a 1% significance level. This means that the accuracy values of at least two configurations are significantly different from each other. To find out what these ones are, we conducted a Nemenyi test [38]. This pair-by-pair comparison showed that the LSTM and the LSTM with embedded context is the only pair of configurations that results in significantly different accuracy levels on a 1% significance level.

5.5. Discussion

First, we showed with our results that an LSTM-based technique for predicting next clicks

achieves a much higher accuracy than an HMM (baseline). A possible reason could be the length of the web sessions and the long-term dependencies between clicks. HMMs are not suitable in this case due to the Markov property (i.e., a future state only depends on the current state), but RNNs provide the necessary long-term memory capabilities [39]. Especially, as stated in section 3.2.2, LSTMs can learn from long web sessions because of their internal memory as well as their different gates that control this memory.

Further, context information plays an important role in predicting the next clicks in web usage mining. In the field of predictive process monitoring, van der Aalst et al. found that it could be insufficient to base predictive process monitoring on pure process data [40]. Márquez-Chamorro et al. argue that context information can improve the predictive quality as it adds valuable information to the predictive model [6]. In the field of web usage mining, it is certainly more complex to predict the next clicks than predicting events in predictive process monitoring. This is due to the fact that a web session usually represents a longer sequence of clicks (a stream of clicks) and has a larger solution space in terms of different click types. The used web usage log in this paper has a maximum sequence length of 247 and 161 unique click types. For instance, the popular bpi2012 (subprocess work item) event log [41] in predictive process monitoring (e.g., used in [31]) has a maximum sequence length of 74 and 6 unique event types. Therefore, context information has high relevance for web usage mining.

However, given our results, the proposed technique with a context-aware configuration for predicting next clicks does not bring a significantly higher accuracy at first glance compared to the configuration without context information. This may be because the available context attributes are not meaningful enough in describing the web sessions of the web usage log. We suppose that only a small part of the existing web session context is described by these attributes, and most of the web session context information is hidden. Furthermore, only a specific subset of the given context attributes could be most significant for describing the web sessions of the web usage log.

Comparing the obtained results of the two context-aware configurations, we can observe that the kind of context representation influences the predictive quality of an LSTM. In the first context-aware configuration of our technique, we numerically transformed the context information with the use of the data mining techniques integer-mapping and normalization. To encode categorical attributes, we preferred an integer-mapping over a low-level encoding with a high expressiveness. For example, a one-hot encoding would lead to many columns and therefore, an inefficient computation if the context attributes have a lot of values. However, with this configuration, the average accuracy is higher than the configuration without context information, but rather limited in comparison to the configuration with embedded context information.

In the second context-aware configuration of our technique, we embed the context information on click level with a *paragraph2vec* technique to capture information about the dependencies between the context attributes. With this configuration, we could achieve the highest accuracy. Beyond our chosen embedding technique, there are further and maybe more expanding approaches for an embedding on click level. One example is an embedding of the context attributes together with control-flow information (means the click information itself and its timestamp). Another possibility could be a hierarchical approach where first the representation of two separate vectors (one for context information and another for control-flow information) is learned, and then based on these a third one is learned representing both types of information. We do not consider these broadening approaches in this paper because our focus is initially on the representation of the context itself.

6. Conclusion and future work

As outlined in the related work section, the two research streams process mining and web usage mining are currently converging. Thus, we presented a new use case for an analytics-based service with a technique touching both domains. For this use case, we adopted an LSTM-based technique from predictive process monitoring (a sub-field of process mining) to predict the next click of running web sessions in the context of web usage mining. For this technique, the three configurations *LSTM without context*, *LSTM with context*, and *LSTM with embedded context* were considered. According to De Koninck et al., *paragraph* embeddings have not yet been evaluated for a LSTM-based technique in the field of predictive process monitoring [9]. Therefore, this paper provides the first contribution in this direction. We evaluated our proposed technique with a real-life web usage log provided by a financial service provider. Based on that, we could show that all of the three configurations performed better than our baseline (HMM) in regard to accuracy and standard deviation. A higher prediction accuracy translates to improved perceived service quality for the users of the recommender

system. Among the three configurations, the LSTM with embedded context achieves a significantly higher accuracy and the lowest standard deviation since it exploits the dependencies between the considered context attributes. Our work includes four main limitations. First, we were only provided with one real-life web usage log from the already mentioned financial provider to evaluate our approach. Second, we did not implement the recommender system as such into the software interface, and therefore, we could not evaluate the actual usefulness of the recommendations for end-users. Such an implementation would allow us to make statements about user satisfaction and the impact on the perceived service quality. Third, we did not perform a hyperparameter optimization (e.g., a grid search) for the LSTM, because this would go beyond the scope of this work. Last but not least, we neglected the fact that web interfaces can change quickly compared to desktop software interfaces. We did not consider how concept drift might impact the quality of our predictions and subsequently of the service [42].

For future research, we plan to evaluate our technique with additional real-life web usage logs with more context attributes (maybe) provided by different companies. Another avenue for future research is to investigate other techniques for encoding categorical attributes that are both more advanced than an integer-mapping and less sparse than a one-hot encoding (e.g., binary encoding) [43]. Finally, we plan to find a way to split long web sessions into logically meaningful sub web sessions. For example, a long web session can include a multitude of different transactions where each transaction is represented by a cluster of clicks that belongs to the same transaction.

Acknowledgements

This project is funded by the German Federal Ministry of Education and Research (BMBF) within the framework programme *Software Campus* (www.softwarecampus.de) under the number 01IS17045. We furthermore want to thank DATEV eG for providing us with data.

References

[1] A. Parasuraman, V. A. Zeithaml, and L. L. Berry, "Servqual: A multiple-item scale for measuring consumer perc," *Journal of Retailing*, vol. 64, no. 1, p. 12, 1988.

[2] ISO/IEC, "ISO/IEC 25010 System and software quality models," tech. rep., ISO/IEC, 2010.

[3] H. Fromm, F. Habryn, and G. Satzger, *Service Analytics: Leveraging Data Across Enterprise Boundaries for Competitive Advantage*. Springer Berlin Heidelberg, 2012.

[4] A. Taivalsaari and T. Mikkonen, "The Web as a Software Platform: Ten Years Later," in *Proceedings of the 13th International Conference on Web Information Systems and Technologies (WEBIST)*, pp. 41–50, 2017.

[5] F. Ricci, L. Rokach, and B. Shapira, "Introduction to Recommender Systems Handbook," in *Recommender Systems Handbook*, pp. 1–35, Springer, Boston, MA, 2011.

[6] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortás, "Predictive Monitoring of Business Processes: A Survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 962–977, 2018.

[7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] T. da Cunha Mattos, F. Santoro, K. Revoredo, and V. Nunes, "A formal representation for context-aware business processes," *Computers in Industry*, vol. 65, pp. 1193–1214, 10 2014.

[9] P. De Koninck, S. vanden Broucke, and J. De Weerdt, "act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes.," in *Proceedings of the 16th International Conference on Business Process Management (BPM)*, pp. 305–321, Springer International Publishing, 2018.

[10] F. Hunke, C. Engel, R. Schritz, and P. Ebel, "Understanding the Anatomy of Analytics-Based Services A Taxonomy to Conceptualize the Use of Data and Analytics in Services," in *Proceedings of the 27th European Conference on Information Systems (ECIS)*, AIS eLibrary, Stockholm, 2019.

[11] W. van der Aalst, *Process Mining: Data Science in Action*. Springer Berlin Heidelberg, 2nd ed., 2016.

[12] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: information and pattern discovery on the World Wide Web," in *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE Comput. Soc, 1997.

[13] Z. Liu, Y. Wang, M. Dontcheva, M. Hoffman, S. Walker, and A. Wilson, "Patterns and sequences: Interactive exploration of clickstreams to understand common visitor paths," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 321–330, 2016.

[14] D. A. Filvà, M. J. C. Guerrero, and M. A. Forment, "Google analytics for time behavior measurement in Moodle," in *Proceedings of the 9th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1–6, IEEE, 2014.

[15] N. Poggi, V. Muthusamy, D. Carrera, and R. Khalaf, "Business process mining from e-commerce web logs," in *Proceedings of the 11th International Conference on Business process management (BPM)*, pp. 65–80, Springer International Publishing, 2013.

[16] Q. A. Liang, J.-Y. Chung, S. Miller, and Y. Ouyang, "Service pattern discovery of web service mining in web service registry-repository," in *Proceedings of the 2006 IEEE International Conference on e-Business Engineering (ICEBE)*, pp. 286–293, IEEE, 2006.

[17] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender System Application Developments: A Survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[18] R. Burke, "Hybrid Web Recommender Systems," in *The adaptive web*, pp. 377–408, Springer, Berlin, Heidelberg, 2007.

[19] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems (ACM RecSys)*, pp. 241–248, ACM, 2016.

[20] Z. A. Pardos, S. Tang, D. Davis, and C. V. Le, "Enabling Real-Time Adaptivity in MOOCs with a Personalized Next-Step Recommendation Framework," in *Proceedings of the 4th ACM Conference on Learning @ Scale (L@S)*, pp. 23–32, ACM, 2017.

[21] S. Gregor and A. R. Hevner, "Positioning and Presenting Design Science Research for Maximum Impact," *MIS Quarterly*, vol. 37, no. 2, pp. 337–355, 2013.

[22] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pp. 1188–1196, 2014.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[24] Y. Bengio, P. Simard, P. Frasconi, *et al.*, "Learning Long-Term Dependencies with Gradient Descent is Difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[25] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[26] S. Schönig, R. Jasinski, L. Ackermann, and S. Jablonski, "Deep Learning Process Prediction with Discrete and Continuous Data Features.," in *Proceedings of th 13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, pp. 314–319, 2018.

[27] N. Navarin, B. Vincenzi, M. Polato, and A. Sperduti, "LSTM Networks for Data-Aware Remaining Time Prediction of Business Process Instances," in *Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–7, IEEE, 2017.

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," *arXiv preprint arXiv:1406.1078*, 2014.

[29] C. Di Francescomarino, C. Ghidini, F. M. Maggi, and F. Milani, "Predictive Process Monitoring Methods: Which One Suits Me Best?," in *Proceedings of the 16th International Conference on Business Process Management (BPM)*, pp. 462–479, Springer International Publishing, 2018.

[30] C. Chand, A. Thakkar, and A. Ganatra, "Sequential Pattern Mining: Survey and Current Research Challenges," *International Journal of Soft Computing and Engineering (IJSCE)*, vol. 2, no. 1, 2012.

[31] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive Business Process Monitoring With LSTM Neural Networks," in *Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAiSE)*, pp. 477–492, Springer International Publishing, 2017.

[32] J. Brunk, K. Revoredo, M. Stierle, M. Matzner, P. Delfmann, and J. Becker, "Prediction of Business Process Instances with Dynamic Bayesian Networks.," in *Proceedings of the 8th International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA)*, pp. 50–54, 2018.

[33] B. A. Tama and M. Comuzzi, "An empirical comparison of classification techniques for next event prediction using business process event logs," *Expert Systems with Applications*, vol. 129, pp. 233–245, 2019.

[34] T. Dozat, "Incorporating Nesterov Momentum Into Adam," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

[35] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A System for Large-Scale Machine Learning," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.

[36] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, ELRA, 5 2010.

[37] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[38] P. Nemenyi, *Distribution-free Multiple Comparisons*. Princeton University, 1963.

[39] Y. Bengio and P. Frasconi, "Diffusion of context and credit information in Markovian models," *Journal of Artificial Intelligence Research*, vol. 3, pp. 249–270, 1995.

[40] W. van der Aalst, M. H. Schonenberg, and M. Song, "Time Prediction Based on Process Mining," *Information systems*, vol. 36, no. 2, pp. 450–475, 2011.

[41] Van Dongen, B.F., *BPI Challenge 2012*. Eindhoven University of Technology, 2012.

[42] L. Baier, N. Kühl, and G. Satzger, "How to Cope with Change?-Preserving Validity of Predictive Services over Time," in *Proceedings of the 52nd Hawaii International Conference on System Sciences (HICSS)*, pp. 1085–1094, 2019.

[43] K. Potdar, T. S. Pardawala, and C. D. Pai, "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *International Journal of Computer Applications*, vol. 175, no. 4, pp. 7–9, 2017.