
RWTH AACHEN UNIVERSITY

# Text-Aware Predictive Business Process Monitoring with LSTM Networks

## Master's Thesis

Author : **David Benedikt Georgi**

Supervisors : Dr.-Ing. M. Seran Uysal
M. Sc. Marco Pegoraro

Examiners : Prof. Dr. ir. Wil van der Aalst
Prof. Two

Registration date : YYYY-MM-DD

Submission date : YYYY-MM-DD

This work is submitted to the institute

**i9 Process and Data Science (PADS) Chair, RWTH Aachen University**

# Abstract

# Contents

# Chapter 1

# Introduction

## 1.1 Motivation

The rapid growth of data generated by large-scale information systems leads to new opportunities for society and businesses. In order to turn the massive amount of data into value, efficient techniques are needed, that are able to extract useful information. The generated value comes in different forms, for example visualizations, models, aggregated data or predictions can be of interest.

A remarkable subset of this data is described as *event data*, which is generated by *process-aware information systems* in order to manage, execute and monitor business processes [1]. With the non stopping rise of digitization of business processes more and more event data becomes utilizable, thus the potential value of this data is exploding.

The scientific engagement aiming to discover, analyze and improve real processes using event data led to *process mining*. Process mining bridges the gab between the data-driven characteristic of data science and the process-centric view of process science [2]. The ongoing success of progress mining in research has been transferred to businesses, that successfully offer or utilize this technology. Celonis, which is often considered as one of the biggest commercial providers of process mining, has been valued 2.5 billion dollar only 9 years after the company was founded [3].

Modern process mining software tends to focus on continuous analysis rather than the traditional offline and project-based approaches. These business process monitoring systems are a key success factor for many organizations, since they allow to understand and supervise all connected processes of a company in real-time as the data is flowing. The core idea of this approach is to automate process mining and keep a persistent connecting between the business process data and the analytical capabilities.

However, traditional process mining tends to be backward-looking [4], i.e. it rather focuses on answering the question "What did happen?", rather than "What will happen?" or even "What should be done?". Therefore, new techniques are required to add the forward-perspective to process mining software.

## 1.2 Problem Statement

Businesses can develop a competitive advantage, if their process mining software has predictive capabilities, that allow to predict the future of a running process instance. For example, if it is known beforehand that a running process instance will probably exceed its deadline, measures can be initiated before damage occurs. Furthermore, information about the next event or the future path of process instance can be of interest. In some scenarios, processes instances have an outcome like success/failure or accepted/declined that can be predicted.

Precisely, given an event log with past executions of a process and a running (i.e. not completed) process instance, we would like to answer the following questions:

- What will happen next?
- When will it happen?
- What is the most likely future path of the instance?
- When will the instance finish?
- What is the outcome of the instance?

## 1.3 Research Goals

This thesis aims to improve current state-of-art approaches for process prediction to improve the capabilities of process monitoring software. The main research goal is to design, implement and evaluate a predictive model for event data that is able to take advantage of additional attribute and textual data associated with each event. Since most current approach are not able to handle textual data, we would like to know to which extend textual data can improve the quality of process prediction. Furthermore, we want to evaluate different design choices for a text-aware process prediction model and point out potential trade-offs.

## 1.4 Contribution

## 1.5 Thesis Structure

This thesis is structured in seven chapters. Chapter 2 summarizes relevant scientific contributions which focus on the problem of prediction in process mining to give an overview of already available methods and their capabilities. In Chapter 3 the notations, definitions and concepts used in this contribution are introduced. A novel text-aware process prediction model is presented in Chapter 4. Moreover, the details regarding the implementation of the model are given in Chapter 5. In Chapter 6 the performance of the model is evaluated and compared to current state-of-the-art prediction methods. Finally, in Chapter 7 a conclusion about all findings and an outlook towards future potential research on process prediction is given.

# Chapter 2

# Related Work

The prediction of the future behavior of an process instance has been an important subfield in the process mining research, that aims to enhance process monitoring capabilities. Depending on the use case e.g. predicting time-related attributes, the future path or the the outcome of a case can be of interest. Most approaches presented in the literature either use machine learning based or process models based methods.

van Dongen et al. presented five different non-parametric regression predictors for forecasting the total cycle time of an unfinished case[5]. The estimates are based on activity occurrences, activity duration and attributes.

van der Aalst et al. proposed to build a transition system using a set, bag or sequence abstraction, which is annotated with time-related data in order to predict the remaining time of case [6]. The core idea of this approach is to replay unfinished cases on the learned transition system and compute the prediction using the annotated data.

Pandey et al. use a Hidden Markov Model to predict the remaining time of a case using the activity and timestamp data of an event log [7].

Rogge-Solti and Weske showed how a Stochastic Petri Net can be used to predict the remaining of a process instance. The model naturally supports parallelism in business processes and considers future events a expected to occur.

Ceci et al. presented an approach, where a sequence tree is learned in order relate a running traces to similar historical traces [9]. A decision tree is then used to predict the next activity and the remaining time of a case.

Teinemaa et al. applied text vectorization techniques like bag of n-grams (BoNG), Latent Dirichlet Allocation (LDA) and Paragraph Vectors (PV) to textual data of processes in order to predict a binary label describing the process outcome[10]. In this approach random forest and logistic regression classifiers for each prefix length of a trace are trained.

Most recently, several authors have applied recurrent neural networks in form of LSTM networks for process prediction. Evermann et al. encode events using an embedding matrix as it is known for word embeddings. The embedded events are then used as input for an LSTM network that predicts the next activity[11].

Tax et al. use an one-hot-encoding of the activity and the timestamp of an event to predict

the activity and timestamp of the next event. This is done by using a two-layered LSTM network[12].

The work by Navarin et al. adopts the idea of using an LSTM network [12] and extends the encoding to additional data attributes associated with each event[13] to predict the remaining time of an case.

Polato et al. presented a set of approaches that use support vector regression for remaining time prediction[14]. In this work the authors implement different encoding for events including simple one-hot-encoding and a more advanced state based encoding using transition systems. Furthermore, they enhance the approach in [6] by taking additional data attributes into account.

Teinemaa et al. reported an in-depth review and benchmark of outcome-oriented predictive process monitoring approaches. The study showed that aggregated encoding like counting frequencies of activities as most reliable encoding for outcome-prediction [15].

A comparison of suggested process prediction methods is presented in Table 2.1.

| **Approach** | **Year** | **Model(s)** | **Data-Aware** | **Text-Aware** | **Predictions** |
|---|---|---|---|---|---|
| van Dongen et al. [5] | 2008 | Regression | ✓ | - | Remaining time |
| van der Aalst et al. [6] | 2011 | Transition system | - | - | Remaining time |
| Pandey et al. [7] | 2011 | Hidden Markov | - | - | Remaining time |
| Rogge-Solti and Weske [8] | 2013 | Stochastic Petri Net | - | - | Remaining time |
| Ceci et al. [9] | 2014 | Sequence Tree<br>Decision Tree | ✓ | - | Next activity<br>Remaining time |
| Teinemaa et al. [10] | 2016 | Random Forest<br>Logistic regression | ✓ | ✓ | Case outcome |
| Evermann et al. [11] | 2016 | LSTM | - | - | Next activity |
| Tax et al. [12] | 2017 | LSTM | - | - | Next activity<br>Future path<br>Next event time<br>Remaining time |
| Navarin et al. [13] | 2017 | LSTM | ✓ | - | Remaining time |
| Polato et al. [14] | 2018 | Transition system<br>SVR | ✓ | - | Next activity<br>Future path<br>Remaining time |
| This approach | 2020 | LSTM | ✓ | ✓ | Next activity<br>Future path<br>Next event time<br>Remaining time<br>Case outcome |

Table 2.1: Comparison of process prediction methods.

# Chapter 3

# Preliminaries

In this section the basic concepts of process mining, long short-term memory networks, text mining as well as necessary formal definitions and notations are defined.

## 3.1 Processes and Process Mining

A *business process* is a collection of activities that are performed in a specific order to archive a goal [16]. A single execution of a process is a *case* or *process instance*, which is identified by a case ID. Each performed activity belongs to specific case and is completed at a certain time. A case can be e.g. a patient in a hospital, a customer of a company or an order and is usually identified by an ID, while the time is specified by a timestamp. The trinity of case, activity and timestamp is called event. An event can have more attributes like describing the resource, costs or transactional information of the event.

If the execution of a business process is logged by an information system, the resulting event data is called *event log*. Depending on the format of the event log, it can also contain additional data on case level. Typical formats for event logs, which are not part of an database, are comma-separated values (CSV) and eXtensible Event Stream (XES) [17]. A table-based representation of an artificial event log can be seen in Table 3.1.

Process mining is the discipline that covers all approaches aiming to generate value out of event data. As an umbrella term, Process Mining includes or utilizes concepts of Business Process Management, Data Mining, Business Process Intelligence, Big Data, Workflow Management, Business Activity Monitoring [2] as well as Machine Learning [18].

Process mining can be divided into a set of subdisciplines mainly process discovery, conformance checking, process enhancement and process analytics [19]. Process discovery aims to generate process models out of event data in order to understand a process and enable further analysis. Conformance checking is about comparing the intended and observed behavior of a process. On top of these diagnostic approaches, process enhancement deals with the improvement of processes.

Driven by the fast and ongoing development of quantitative prediction methods in data science and machine learning, prediction-based methods have been applied to event data leading to process prediction, a major subfield in process analytics. Forecasting the future

| ID | Activity | Timestamp | Resource | Cost | Comment |
|---|---|---|---|---|---|
| 0 | Register patient | 01.02.2020:14.12 | SYSTEM | 0 | - |
| | Consultation | 01.02.2020:14.34 | John Brown, MD | 24.32 | The patient reports persist nausea. |
| | Blood test | 01.02.2020:15.12 | Kim Smith | 14.23 | Tests: Complete blood cou |
| | Evaluate test result | 01.02.2020:16.35 | John Brown, MD | 38.67 | No abnormalities in the co blood count. |
| | Release patient | 01.02.2020:17.24 | SYSTEM | 0 | - |
| 1 | Register patient | 02.02.2020:08.20 | SYSTEM | 0 | - |
| | Consultation | 02.02.2020:14.12 | Jana Simpson, MD | 24.32 | Noticeable tachycardia. No are known. |
| | MRI | 02.02.2020:14.12 | Sara Taylor, MD | 352.87 | - |
| | Release patient | 02.02.2020:14.12 | SYSTEM | 0 | - |
| 2 | Register patient | 02.02.2020:09.08 | SYSTEM | 0 | - |
| | Consultation | 02.02.2020:09.14 | Jana Simpson, MD | 24.32 | The patient has severe leg due to a motorcycle accide |
| | Patient hospitalized | 02.02.2020:09.20 | Mike Johnson | 130.37 | - |
| ... | ... | ... | ... | ... | ... |

Table 3.1: Artifical event log of patient treatment in a hospital

of a running process instance is one of the main goals in process prediction and also the main topic of this thesis.

## 3.2 Basic notations, Sequences and Functions

The set $\mathbb{N}$ denotes the set of all natural numbers $\{1, 2, 3, \dots\}$, while $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$ denotes the set of natural numbers including 0. Given a set $A$, $A^n$ describes the set of all sequences $\langle a_1, a_2, \dots, a_n \rangle$ over $A$ of length $n$ with $a_i \in A$, $1 \leq i \leq n$. The set $A^0$ is defined as $\{\langle\rangle\}$, where $\langle\rangle$ is the empty sequence of length 0. The set of all possible sequences over $A$ is given with $A^* = \bigcup_{i \in \mathbb{N}_0} A^i$.

Sequences $\sigma_1$, $\sigma_2$ can be concatenated to $\sigma_1 \cdot \sigma_2$. Moreover, the $i$th element of a sequence $\sigma = \langle a_1, a_2, \dots, a_n \rangle$ is accessed by $\sigma(i) = a_i$ for $1 \leq i \leq n$. To derive a subsequence of $\sigma$ we write $\sigma(i, j) = \langle a_i, a_{i+1} \dots, a_j \rangle$ with $1 \leq i < j \leq n$. The length of a sequence is denoted by $|\sigma|$. If a sequence $\sigma$ contains an element $a_i$, we also write $a_i \in \sigma$.

A function $f \in A \to B$ can be lifted to sequences $\sigma$ over $A$ element-wise, precisely:

$$f(\sigma) = \begin{cases} \langle\rangle & \text{if } \sigma = \langle\rangle \\ \langle f(a_1), f(a_2), \dots, f(a_n) \rangle & \text{else} \end{cases}$$

## 3.3 Events, Traces, Event logs

**Definition 3.1.** An *event* is defined by tuple $e = (a, c, t, d_1, \ldots, d_m) \in \mathcal{C} \times \mathcal{A} \times \mathcal{T} \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_m = \mathcal{E}$ where $c \in \mathcal{C}$ is the case id, $a \in \mathcal{A}$ is the executed activity and $t \in \mathcal{T}$ is the timestamp of the event. Furthermore, each event contains a fixed number $m \in \mathbb{N}_0$ of additional attributes $d_1 \ldots d_m$ in their corresponding domains $\mathcal{D}_1, \ldots, \mathcal{D}_m$. In case that no additional attribute data is given ($m = 0$) the event space $\mathcal{E}$ (set of all possible events) is reduced to $\mathcal{C} \times \mathcal{A} \times \mathcal{T}$.

Each attribute $d \in \mathcal{D}$ of an event (including activity, timestamp and case ID) can be accessed by a projection function $\pi_D \in \mathcal{E} \to \mathcal{D}$. For example, the activity $a$ of an event $e$ is retrieved by $\pi_{\mathcal{A}}(e) = a$.

Throughout this thesis, $\mathcal{C} = \mathbb{N}_0$, $|\mathcal{A}| < \infty$ and $\mathcal{T} = \mathbb{R}$ is assumed, where $t \in \mathcal{T}$ is given in Unix time, precisely the number of seconds since 00:00:00 UTC on 1 January 1970 minus the applied leap seconds. Each additional attribute is assumed to be numerical, categorical or textual, i.e. $\mathcal{D}_i = \mathbb{R}$, $|\mathcal{D}_i| < \infty$ or $\mathcal{D}_i = \Sigma^*$ for $1 \leq i \leq m$ and some fixed Alphabet $\Sigma$.

**Definition 3.2.** A *trace* is a finite and non-empty sequence of events $\sigma = \langle e_1, e_2, \ldots \rangle \in \mathcal{E}^*$ with increasing timestamps, i.e. $\pi_{\mathcal{T}}(e_i) < \pi_{\mathcal{T}}(e_j)$ for $1 \leq i < j \leq |\sigma|$.

By lifting the projection functions to sequences a trace can be transformed into a sequence of attributes by applying the projection function to the trace. For example, $\pi_{\mathcal{A}}(\sigma)$ gives the sequence of the activities of the events in $\sigma$.

**Definition 3.3.** An *event log* $\mathbb{L} = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ is a set of traces, where each event of a trace is unique in the log and all events of a trace share a case IDs, which is unique per trace.

## 3.4 Text mining

Text mining describes all techniques to generate value out of unstructured or semi-structured textual data. It combines concepts of natural language processing, machine learning and data mining [20]. The base object in text mining is a *document* containing textual data. The text can be completed unstructured, i.e. it does not conform to a pre-defined data model, or semi-structured, like in an e-mail, where text information is assigned to sender, subject, message etc. A collection of documents is called *text corpus*, which forms the basis for many text mining techniques.

In order to derive a mathematical representation of the text data that can be interpreted by a computer, a text model has to be build using the text corpus. Popular text models are Bag-of-words, Bag-of-n-gram, Paragraph vector (a.k.a. Doc2Vec) [21] and Latent Dirichlet Allocation [22]. Most models require a text preprocessing step, where the text is cleaned from linguistic variation as well as meaningless words and symbols.

## 3.5 Long short-term memory networks

Long short-term memory (LSMT) is an advanced recurrent neural network architecture for sequential data originally presented by Hochreiter and Schmidhuber in 1997 [23]. This

approach addresses the well-known vanishing and exploding gradient problem [24] of traditional recurrent neural networks by introducing more complex LSTM cells as hidden units. The proposed architecture has been improved several times [25] [26] and considered as one of the most successful recurrent neural network models. Although LSTM networks have been available for a long time, the breakthrough of this technology is dated around 2016 after many success stories of LSTM in combination with large data sets and GPU hardware have been reported for sequence to sequence tasks like text translation [27].

Gated recurrent units (GRU) [28] are the competing gating mechanism by Cho et al. that have fewer parameters and perform similar to LSTM. However, more recent studies show, that LSTM outperforms GRU consistently in neural machine translation tasks [29].

A simple feedforward neural networks consists of an input layer, arbitrarily many hidden layers and an output layer, where each layer consists of cells that compute and output the weighted sum of the cells of the previous layer that has been passed to an activation function [30]. These networks can learn complex functions in supervised learning settings where input and output pattern are provided. The network computes a loss function for each training pattern and adjusts its weights using a back-propagation algorithm [31].

Recurrent neural networks extend traditional feed forward networks with backfeeding connections between hidden cells. This allows the network to keep a state across inputs and allows the neural network to process arbitrarily long sequences of input data while learning temporal dependencies.

In LTSM networks the hidden neurons are replaced with more complex LSTM cells. These cells uses as input the state $C_{t-1}$ and the output $h_{t-1}$ of the cell in the previous time step and the output of the previous layer. Each LSTM cell produces an output $h_t$ and state $C_t$.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.2}$$

$$\bar{C} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.3}$$

$$C_t = f_t \times C_{t-1} + i_t \times \bar{C}_t \tag{3.4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.5}$$

$$h_t = o_t \times \tanh(C_t) \tag{3.6}$$

# Chapter 4

# Text-Aware Process Prediction

Text-aware process prediction aims to utilize unstructured text information in event data to improve predictions for unfinished cases. While many prediction methods have been applied to event data, almost none of them are able to handle textual data. An exception is the approach presented in [10], where traces with text data are encoded as fixed vectors and a classifier is learned for each prefix length.

In this chapter a novel approach for process prediction is presented that is able to handle the control flow, additional numerical, categorical and textual data, captures temporal dependencies, seasonal variability and concept drifts using an event-wise encoding and a sequential prediction model (LSTM).

## 4.1 Overview

The proposed framework consists of a preprocessing, encoding and prediction model component, which operate in an offline and online phase. The encoding component distinguish between categorical or numerical data that can be encoded directly and textual data, which requires a textual model.

In the offline phase a historical event log with completed traces of a business process is used to train the prediction model. Given an event log $\mathbb{L} = \{\sigma_1, \ldots, \sigma_l\}$ with historical event data, the set of all prefix traces $\mathbb{L}_{\text{prefix}} = \{\sigma(1,k) \mid \forall \sigma \in \mathbb{L}, 1 \leq k \leq |\sigma|\}$ is computed. Each prefix with the desired target value corresponds to one training example for the prediction model. The total number of training examples that can be generated out of the log is $\sum_{\sigma \in \mathbb{L}} |\sigma|$, which is exactly the number of events in the log.

As target the activity and timestamp (relative to case start) of the next event as well as the outcome and cycle time (time between first and last event) of the case is chosen. For completed cases the next event's activity is an artificial "<Process End>" with the same timestamp as the final event.

The text data of the historical log is extracted to build a text corpus and construct a text model.

The prefix traces are encoded to sequences of fixed-length event vectors using the text model.

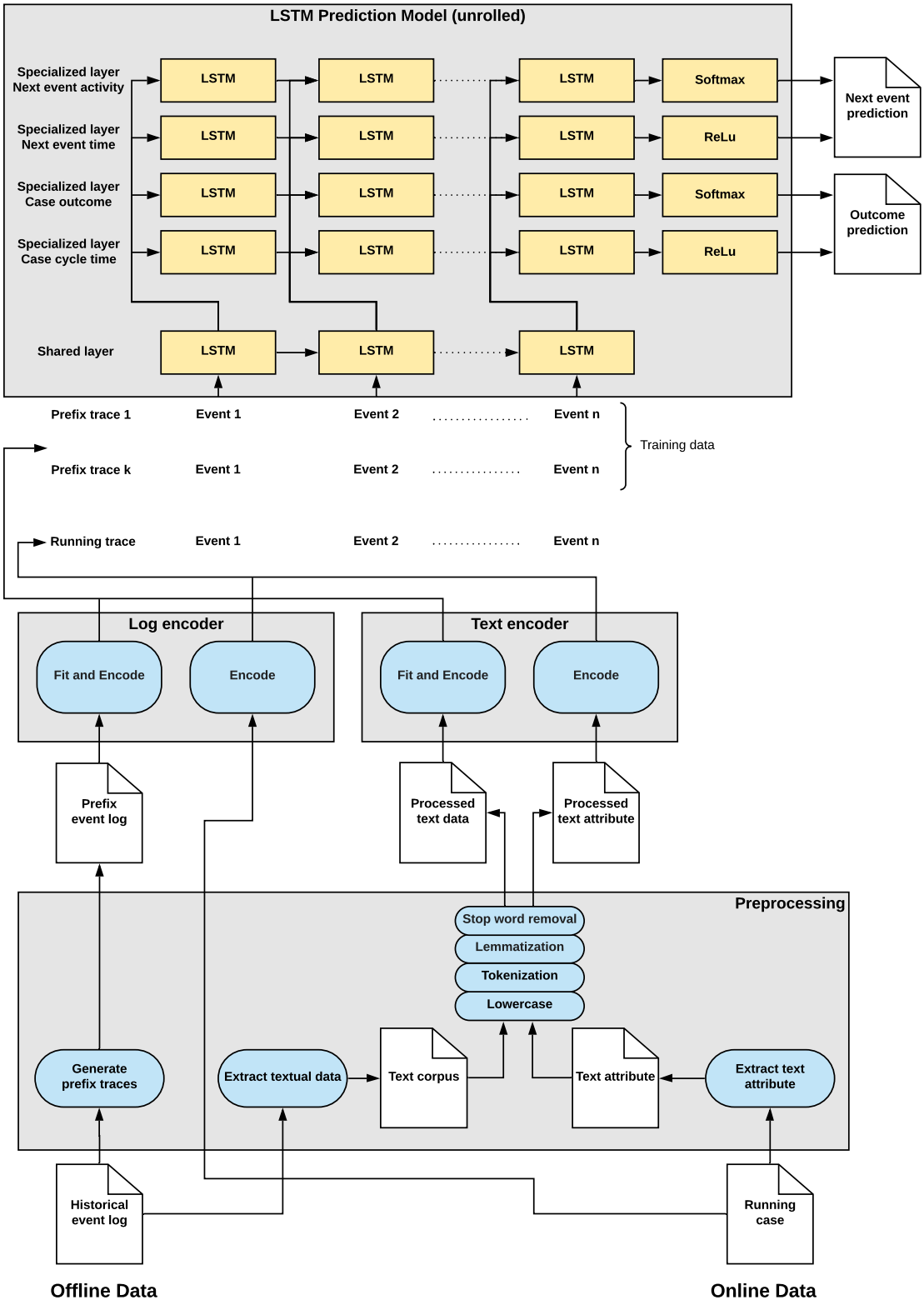


Figure 4.1: Framework

## 4.2 Log Encoding

Each event is encoded in a fixed-length vector using the event's activity, timestamp and additional categorical, numerical and textual attributes. The encoding of each attributes

$encode(e_i)$

### 4.2.1 Encoding of Activities and other Categorical Attributes

### 4.2.2 Encoding of Time and other numerical Attributes

For timestamp prediction of the next and final event of running process instance a set of time-based features is computed from the timestamp data in the event log. Given an event log $\mathbb{L}$ an event $e_i$ from a trace $\sigma = \langle e_1, \ldots, e_n \rangle$ the following time features are computed for the encoding of $e_i$:

| **Feature** | **Description** |
|---|---|
| $t_1 = \pi_{\mathcal{T}}(e_i) - \pi_{\mathcal{T}}(e_{i-1})$ | Seconds since previous event |
| $t_2 = \pi_{\mathcal{T}}(e_i) - \pi_{\mathcal{T}}(e_1)$ | Seconds since case start |
| $t_3 = \pi_{\mathcal{T}}(e_i) - \min\{\pi_{\mathcal{T}}(e_j), \forall e_j \in \sigma_k, \forall \sigma_k \in \mathbb{L}\}$ | Seconds since first recorded event |
| $t_4$ | Seconds since midnight |
| $t_5$ | Seconds since last Monday |
| $t_6$ | Seconds since last January 1 00:00 |

Using the time features a set of time-dependent trends can be captured and utilized for prediction. The features $t_1$ and $t_2$ give information about the time between events and the time of the event in the case. Using $t_3$ the absolute time position of an event in the data can be determined. This is important to detect concept drift in the process. The features $t_4, t_5$ and $t_6$ are used to capture daily, weekly or seasonal trends. For example, some activities might only be executed during office hours, before the weekend, during summer.

Each feature $t_1, \ldots t_6$ as well as all additional numerical attributes $d_i$ are scaled to the interval $[0, 1]$ to improve learning efficiency using min-max normalizing. The scaling for a numerical feature $x$ is realized with the transformation

$$\hat{x} = \frac{x - \min(x)}{\max(x) - \min(x)} \in [0, 1],$$

where $\min(x)$ is the lowest and $\max(x)$ is the highest value $x$ can take. If the limits are not bounded conceptually, the lowest or highest value of $x$ in the event log is used for scaling.

### 4.2.3 Text Encoding

In order to prepare the textual data of the event log for a prediction model, the texts have to be encoded in a "useful" numerical vector representation. Useful in that context means, that texts with similar semantic meanings should also have similar representations. Extracting the meaning of textual information remains a challenge even for humans, because textual data is unstructured, language dependent and domain specific. In addition,

| Step | Transformation | Document |
|---|---|---|
| 0 | Original | "The patient has been diagnosed with high blood pressure." |
| 1 | Lowercase | "the patient has been diagnosed with high blood pressure." |
| 2 | Tokenization | ["the", "patient", "has", "been", "diagnosed", "with", "high", "blood", "pressure", "."] |
| 3 | Lemmatization | ["the", "patient", "have", "be", "diagnose", "with", "high", "blood", "pressure", "."] |
| 4 | Stop word filtering | ["patient", "diagnose", "high", "blood", "pressure"] |

Table 4.1: Preprocessing transformation of an example document.

grammatical variations and the importance of context in language makes text mining even more difficult.

The text vectorization is realized in a 2-step procedure. First, all text data associated with the events in the corresponding textual attribute is collected in a so called *text corpus*. Each document in the text corpus is then preprocessed in order to filter out linguistic noise or useless information. Finally, the text corpus is used to build up a vocabulary and a text vectorization technique is applied to encode the text attribute to a fixed of vector.

#### 4.2.3.1 Text Preprocessing

In the preprocessing step each document is transformed by a processing pipeline which consists of the following four steps:

1. Letters are converted to lowercase
2. Document is tokenized by word
3. Each word is lemmatized
4. Stop words are filtered out

In the tokenenization step a document is split up in a list of words. Each word is then lemmatized, i.e. it is converted to its canonical form. The idea is to unify words that have a very similar meaning and filter out grammatical variations. For example, the words "go", "going", "goes", "gone" and "went" are all transformed to the basic form "go". Ultimately, all stop words are filtered out of each document. Stop words are words with low information value like "the", "a", "of" or "here".

Stop word lists are language dependent and can be more or less aggressive at filtering. Usually they contains articles, auxiliary verbs, prepositions and generic verbs like "be" and "have". In addition, punctuation marks or numerical information are excluded.

#### 4.2.3.2 Bag-of-N-Gram

#### 4.2.3.3 Paragraph Vector

## 4.3 Network

# Chapter 5

# Implementation

# Chapter 6

# Evaluation

# Chapter 7

# Conclusion

# Bibliography


[1] Wil M. P. van der Aalst. Process-aware information systems: Lessons to be learned from process mining. *Trans. Petri Nets Other Model. Concurr.*, 2:1–26, 2009. doi: 10.1007/978-3-642-00899-3_1. URL https://doi.org/10.1007/978-3-642-00899-3_1.

[2] Wil M. P. van der Aalst. *Process Mining - Data Science in Action, Second Edition*. Springer, 2016. ISBN 978-3-662-49850-7. doi: 10.1007/978-3-662-49851-4. URL https://doi.org/10.1007/978-3-662-49851-4.

[3] Ryan Browne. How three friends turned a college project into a $2.5 billion software unicorn, 2019. URL https://web.archive.org/web/20200125191917/https://www.cnbc.com/2019/11/21/celonis-raises-290m-series-c-funding-round-at-2point5b-valuation.html. Accessed = 2020-04-20.

[4] Wil M. P. van der Aalst. Process mining and simulation: a match made in heaven! In *Proceedings of the 50th Computer Simulation Conference, SummerSim 2018, Bordeaux, France, July 09-12, 2018*, pages 4:1–4:12. ACM, 2018. URL http://dl.acm.org/citation.cfm?id=3275386.

[5] Boudewijn F. van Dongen, R. A. Crooy, and Wil M. P. van der Aalst. Cycle time prediction: When will this case finally be finished? In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008, OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008, Monterrey, Mexico, November 9-14, 2008, Proceedings, Part I*, volume 5331 of *Lecture Notes in Computer Science*, pages 319–336. Springer, 2008. doi: 10.1007/978-3-540-88871-0_22. URL https://doi.org/10.1007/978-3-540-88871-0_22.

[6] Wil M. P. van der Aalst, M. H. Schonenberg, and Minseok Song. Time prediction based on process mining. *Inf. Syst.*, 36(2):450–475, 2011. doi: 10.1016/j.is.2010.09.001. URL https://doi.org/10.1016/j.is.2010.09.001.

[7] Suraj Pandey, Surya Nepal, and Shiping Chen. A test-bed for the evaluation of business process prediction techniques. In Dimitrios Georgakopoulos and James B. D. Joshi, editors, *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2011, Orlando, FL, USA, 15-18 October, 2011*, pages 382–391. ICST / IEEE, 2011. doi: 10.4108/icst.collaboratecom.2011.247129. URL https://doi.org/10.4108/icst.collaboratecom.2011.247129.

[8] Andreas Rogge-Solti and Mathias Weske. Prediction of remaining service execu-

tion time using stochastic petri nets with arbitrary firing delays. In Samik Basu, Cesare Pautasso, Liang Zhang, and Xiang Fu, editors, *Service-Oriented Computing - 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings*, volume 8274 of *Lecture Notes in Computer Science*, pages 389–403. Springer, 2013. doi: 10.1007/978-3-642-45005-1_27. URL https://doi.org/10.1007/978-3-642-45005-1_27.

[9] Michelangelo Ceci, Pasqua Fabiana Lanotte, Fabio Fumarola, Dario Pietro Cavallo, and Donato Malerba. Completion time and next activity prediction of processes using sequential pattern mining. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8777 of *Lecture Notes in Computer Science*, pages 49–61. Springer, 2014. doi: 10.1007/978-3-319-11812-3_5. URL https://doi.org/10.1007/978-3-319-11812-3_5.

[10] Irene Teinemaa, Marlon Dumas, Fabrizio Maria Maggi, and Chiara Di Francescomarino. Predictive business process monitoring with structured and unstructured data. In Marcello La Rosa, Peter Loos, and Oscar Pastor, editors, *Business Process Management - 14th International Conference, BPM 2016, Rio de Janeiro, Brazil, September 18-22, 2016. Proceedings*, volume 9850 of *Lecture Notes in Computer Science*, pages 401–417. Springer, 2016. doi: 10.1007/978-3-319-45348-4_23. URL https://doi.org/10.1007/978-3-319-45348-4_23.

[11] Joerg Evermann, Jana-Rebecca Rehse, and Peter Fettke. A deep learning approach for predicting process behaviour at runtime. In Marlon Dumas and Marcelo Fantinato, editors, *Business Process Management Workshops - BPM 2016 International Workshops, Rio de Janeiro, Brazil, September 19, 2016, Revised Papers*, volume 281 of *Lecture Notes in Business Information Processing*, pages 327–338, 2016. doi: 10.1007/978-3-319-58457-7_24. URL https://doi.org/10.1007/978-3-319-58457-7_24.

[12] Niek Tax, Ilya Verenich, Marcello La Rosa, and Marlon Dumas. Predictive business process monitoring with LSTM neural networks. In Eric Dubois and Klaus Pohl, editors, *Advanced Information Systems Engineering - 29th International Conference, CAiSE 2017, Essen, Germany, June 12-16, 2017, Proceedings*, volume 10253 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2017. doi: 10.1007/978-3-319-59536-8_30. URL https://doi.org/10.1007/978-3-319-59536-8_30.

[13] Nicolò Navarin, Beatrice Vincenzi, Mirko Polato, and Alessandro Sperduti. LSTM networks for data-aware remaining time prediction of business process instances. In *2017 IEEE Symposium Series on Computational Intelligence, SSCI 2017, Honolulu, HI, USA, November 27 - Dec. 1, 2017*, pages 1–7. IEEE, 2017. doi: 10.1109/SSCI.2017.8285184. URL https://doi.org/10.1109/SSCI.2017.8285184.

[14] Mirko Polato, Alessandro Sperduti, Andrea Burattin, and Massimiliano de Leoni. Time and activity sequence prediction of business process instances. *Computing*, 100 (9):1005–1031, 2018. doi: 10.1007/s00607-018-0593-x. URL https://doi.org/10.1007/s00607-018-0593-x.

[15] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, and Fabrizio Maria Maggi. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM*

*Trans. Knowl. Discov. Data*, 13(2):17:1–17:57, 2019. doi: 10.1145/3301300. URL https://doi.org/10.1145/3301300.

[16] Wil M. P. van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, R. P. Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos C. A. M. Buijs, Andrea Burattin, Josep Carmona, Malú Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn F. van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul, Frank van Geffen, Sukriti Goel, Christian W. Günther, Antonella Guzzo, Paul Harmon, Arthur H. M. ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maria Maggi, Donato Malerba, R. S. Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid R. Motahari Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith D. Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Thandar Wynn. Process mining manifesto. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*, volume 99 of *Lecture Notes in Business Information Processing*, pages 169–194. Springer, 2011. doi: 10.1007/978-3-642-28108-2_19. URL https://doi.org/10.1007/978-3-642-28108-2_19.

[17] Eric Verbeek, Joos C. A. M. Buijs, Boudewijn F. van Dongen, and Wil M. P. van der Aalst. XES tools. In Pnina Soffer and Erik Proper, editors, *Proceedings of the CAiSE Forum 2010, Hammamet, Tunisia, June 9-11, 2010*, volume 592 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010. URL http://ceur-ws.org/Vol-592/PaperDemo07.pdf.

[18] Fabian Veit, Jerome Geyer-Klingeberg, Julian Madrzak, Manuel Haug, and Jan Thomson. The proactive insights engine: Process mining meets machine learning and artificial intelligence. In Robert Clarisó, Henrik Leopold, Jan Mendling, Wil M. P. van der Aalst, Akhil Kumar, Brian T. Pentland, and Mathias Weske, editors, *Proceedings of the BPM Demo Track and BPM Dissertation Award co-located with 15th International Conference on Business Process Modeling (BPM 2017), Barcelona, Spain, September 13, 2017*, volume 1920 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-1920/BPM_2017_paper_192.pdf.

[19] Maikel L. van Eck, Xixi Lu, Sander J. J. Leemans, and Wil M. P. van der Aalst. PM ^2 : A process mining project methodology. In Jelena Zdravkovic, Marite Kirikova, and Paul Johannesson, editors, *Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings*, volume 9097 of *Lecture Notes in Computer Science*, pages 297–313. Springer, 2015. doi: 10.1007/978-3-319-19069-3_19. URL https://doi.org/10.1007/978-3-319-19069-3_19.

[20] Rada Mihalcea. *The Text Mining Handbook: Advanced Approaches to Analyz-*

*ing Unstructured Data* ronen feldman and james sanger (bar-ilan university and ABS ventures) cambridge, england: Cambridge university press, 2007, xii+410 pp; hardbound, ISBN 0-521-83657-3. *Comput. Linguistics*, 34(1):125–127, 2008. doi: 10.1162/coli.2008.34.1.125. URL https://doi.org/10.1162/coli.2008.34.1.125.

[21] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/le14.html.

[22] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003. URL http://jmlr.org/papers/v3/blei03a.html.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[24] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 1310–1318. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/pascanu13.html.

[25] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000. doi: 10.1162/089976600300015015. URL https://doi.org/10.1162/089976600300015015.

[26] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *IEEE Trans. Neural Networks Learn. Syst.*, 28(10):2222–2232, 2017. doi: 10.1109/TNNLS.2016.2582924. URL https://doi.org/10.1109/TNNLS.2016.2582924.

[27] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL http://arxiv.org/abs/1609.08144.

[28] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179.

[29] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc V. Le. Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906, 2017. URL `http://arxiv.org/abs/1703.03906`.

[30] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. URL `https://doi.org/10.1016/j.neunet.2014.09.003`.

[31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

# Acknowledgments