HAW HAMBURG

MASTERTHESIS
Tasmin Herrmann

Deep Learning based Buy Predictions with Sequence Models

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Tasmin Herrmann

Deep Learning basierte Kaufprognose mit Sequenz Modellen

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Marina Tropmann-Frick
Fachlicher Betreuer: Dr. Sascha Lange

Eingereicht am: 10. Oktober 2019

Acknowledgement

Throughout the writing of this thesis, I have received a great deal of support and assistance.
I would like to express my sincere gratitude to my supervisor, Dr. Sascha Lange of the PSIORI GmbH, who assisted me in the formulation of the question and the solution approaches of the thesis.
I would also like to thank PSIORI GmbH overall for its support throughout my master thesis.
Furthermore, I am grateful to my supervisors, Prof. Dr. Kai von Luck and Prof. Dr. Marina Tropmann-Frick, who discussed the structure of the thesis with me and accompanied me during its development.
This work was created within the context of the machine learning working group ML AG at HAW Hamburg.
I would also like to acknowledge Prof. Dr. Kai von Luck and M.Sc. Tobias Eichler, who actively supported the working group, encouraged an exchange between the students and supervised our projects. I would especially like to thank Matthias Nitsche, Stephan Halbritter and Timo Lange for the professional exchange in this group as well as their work on the systems to provide a platform for machine learning projects. I must also recognize Henrik Wortmann from the /* CREATIVE SPACE FOR TECHNICAL INNOVATIONS */ laboratory at HAW Hamburg, who has driven the development of an infrastructure for machine learning projects at the HAW Hamburg alongside Tobias Eichler, in addition to all members of the ML AG who are not mentioned by name for their exchange and participation, which contributed to this work.
Finally, I would like to thank my family and my boyfriend, Stephan Krugmann, for their support and faith in me.

Tasmin Herrmann

Thema der Arbeit

Deep Learning basierte Kaufprognose mit Sequenz Modellen

Stichworte

Kaufprognose, elektonischer Handel, rekurrente neuronale Netze

Kurzzusammenfassung

Die Identifizierung, ob ein angeklickter Artikel in einer E-Commerce Sitzung mit einem Kauf endet, ist ein aktuelles Forschungsthema. Diese Aufgabe wurde auch bei der RecSys Challenge im Jahr 2015 gestellt. Der Wettbewerb wurde mit einem zweistufigen Ansatz von Romov und Sokolov gewonnen. Sie erkannten zuerst die Käufer mit Hilfe von Sitzungsdaten und bestimmten dann die gekauften Artikel auf den erkannten Kaufsitzungen. In den Merkmalen ihrer Modelle gibt es jedoch keine Informationen über die genauen Klickseqeunzen der betrachteten Artikel in einer Sitzung. Daher untersuche ich in dieser Arbeit die Frage, ob der Kauf von Produkten von der Klicksequenz der betrachteten Artikel in einer Sitzung abhängt. Zu diesem Zweck trainiere ich Sequenzmodelle auf den Klicks der Sitzungen mit Käufen, um zu prüfen, ob sich die gekauften Artikel mit Hilfe dieser vorhersagen lassen. Hier verwende ich die beiden verschiedenen Sequenzmodelle *Sequence Classification* und *Sequence Labeling*. Ich kombiniere meine Sequenzmodelle mit den Modellen von Romov und Sokolov, um zu untersuchen, ob sich ihr Ansatz mit Sequenzinformationen verbessert. Meine Untersuchungen haben gezeigt, dass der Kauf eines Artikels von der Klickfolge der angesehenen Artikel abhängt und zu einer Verbesserung der Vorhersageergebnisse eines merkmalsbasierten Ansatzes wie der von Romov und Sokolov beitragen kann. Zudem wurde gezeigt, dass für den Ansatz die Datenvorbereitung weniger zeitaufwendig ist als für einen merkmalsbasierten Ansatz.

Tasmin Herrmann

Title of Thesis
Deep Learning based Buy Predictions with Sequence Models

Keywords
Buy prediction, e-commerce, recurrent neural networks

Abstract
Identifying whether a clicked item in an e-commerce session will end in a purchase is a current research inquiry. This task was also set at the RecSys Challenge in 2015. The competition was won with a two-stage approach by Romov and Sokolov. They first recognized the buyers with the help of session data and then determined the purchased items on the recognized purchase sessions. However, in the features of their models, there is no information about the exact click sequences of the items that were viewed in a session. Therefore, in this paper, I examine the question of whether the purchase of products depends on the click stream of the items that are viewed in a session. For this purpose, I train sequence models on the item clicks of the sessions with purchases to check whether they can help predict purchased items. Here, I use the two sequence models of sequence classification and sequence labeling. I combine my sequence models with those of Romov and Sokolov to determine if their approach improves with sequence information. This research demonstrates that buying an item depends on the click order of the items that are viewed and can improve the predictive results of a feature-based approach, such as that of Romov and Sokolov. In addition, the research illustrates that data preparation is less time-consuming for this approach than for a feature-based approach.

Contents

List of Figures

List of Tables

Abbreviations

AUC area under the curve.

BRNN bidirectional recurrent neural network.

CRF conditional random field.

ELMo embeddings from language models.

FN false negatives.

FP false positives.

FPR false positive rate.

GRU gated recurrent unit.

LSTM long short-term memory.

MLP multilayer perceptron.

NLP natural language processing.

RNN recurrent neural network.

TN true negatives.

TP true positives.

TPR true positive rate.

1 Introduction

Electronic commerce (e-commerce) is the activity of buying or selling products via online services or the Internet. Revenues in the e-commerce market has amounted to about €1,795,989 million in 2019 and is forecasted to reach a market volume of €2,527,739 million in 2023 according to the forecast. This prediction corresponds to annual revenue growth of 8.9% [38].

Marketing also plays a major role in this market, which attaches importance to personalization of the shopping experience and the recommendation of individual products to each user. These elements are not always feasible with long-term collected data, such as user profiles or information about products, as new users constantly enter online shops for which no data are stored yet. In addition, the General Data Protection Regulation (GDPR) restricts the use or storage of personal data of users. In addition, user interests that cannot be represented by long-term profiles also change at short notice. In such cases, session-based information is used for personalization. One important topic in session-based personalization is the session-based recommendation of items. However, this aspect is only one use case for buy prediction. Some approaches engage with session-based recommendations of items that are facilitated by deep learning.

In 2015, the company YOOCHOOSE,[1] which develops recommendations systems and other personalization software, collaborated with ACM to provide a dataset with click streams from an online shop as part of the RecSys Challenge. This challenge requested that they identify purchase sessions and recognize the purchased items. The motivation for the competition was the importance of the information that people buy and whether they will buy at all. This information can be used to determine which items to recommend and how to lead the user to become a buyer, such as through targeted advertising and discounts.

The solution that won the competition developed two models: purchase detection, which identifies the buyers; and item detection, which identifies the bought items from the purchase sessions. For these models, they built large feature sets that describe a session, which were then used to train models and predict buys.

[1] https://www.yoochoose.com

1.1 Research Questions

The feature sets of the winning solution do not contain any information about the actual item click stream. There have also been published papers that successfully give item recommendations with the click stream of the dataset. The assumption follows that the previous clicks influence not only the next clicked item but also the purchased item. With this assumption, I propose the following hypotheses::

1. The buying behavior will be reflected in the item click streams and this information can be used to predict item purchases.
2. The prediction of item purchases improves if one adds the item click stream information to the existing model (if it did not previously contain such information).
3. The combined buy prediction from a purchase detection and item detection model improves when item click stream information is included in item detection.
4. The preparation of the input data is less time-consuming for sequence-based models than for state-based models.

I assess these four hypotheses in this thesis. To this end, I aim to predict purchases of items with recurrent neural networks (RNNs). These models make predictions on the basis of sequential data. I seek to combine these RNNs with the item detection model of Romov and Sokolov to enhance the ability of that model to forecast item purchases. The resulting item detection model is then be combined with the purchase detection model to determine if the overall buy prediction improves.

1.2 Outline

To evaluate the hypotheses, I first present the dataset in more detail in Chapter 2. In addition, I describe the approach of Romov and Sokolov as well as the approaches with RNNs on the dataset to recommender systems. Subsequently, I explain my previous experiments. In chapter 3 the methods used for the experiments follow. In this chapter, I elaborate on the relevant algorithms and evaluation metrics of the experiments. In Chapter 4, I describe the reproduction of the models of Romov and Sokolov and compare my results to those they have described. The reproduction of the models is necessary to link them with my own models. The experiments then follow to investigate the hypotheses. The models are described, and the results are discussed. Finally, the conclusion section briefly summarizes the results and delivers an outlook on the topic.

2 Buy predictions

This chapter addresses the buy prediction and the dataset considered with previous approaches. Chapter 3 offers further details on the methods and basics.

Buy prediction belongs to predictive analytics. Predictive analytics [29] includes a variety of statistical and analytical methods that are employed to develop models that predict future events or behaviors. The shape of these predictive models depends on the behavior or event that they predict. However, most predictive models calculate a score with the likelihood that the target behavior or event will occur. In the buy prediction context, the model predicts a buy action of an item with the session data.

Data mining [9] can be used to create a predictive model by identifying patterns, trends and relationships among the data. These techniques are based on statistical methods, such as regression or time-series models, and can provide non-obvious knowledge. In this thesis, I want to apply data mining techniques to develop predictive models for buy prediction and to analyze the data to gain knowledge about users and their sessions. Other application examples include fraud detection in credit institutes [4] and predictive maintenance [25]. Predictive maintenance connects information from different devices and machines in real time to improve maintenance processes by facility managers.

Buy prediction is assigned to the context of recommender systems [34]. Recommender systems attempt to predict the preference that a user would give to an item. This information can be used to offer suggestions to the user. Collaborative filtering [33] and content-based filtering [23] are the most frequently used types. The former searches users with the same pattern of behavior as the target user and uses their information to predict the behavior of the target user, while the latter searches items that are similar to those that the target user likes. The buy prediction aims to anticipate not only interest in an item but also the act of purchasing an item. There are several ongoing inquiries in buy prediction research. However, I first introduce the dataset before summarizing the current research on buy prediction.

2.1 Dataset

The dataset that is considered in this paper was published in the context of the RecSys Challenge in 2015.[1] This competition takes place within the ACM RecSys Challenge conference and concerns recommender systems. In 2015, buy predictions were the topic of the challenge, and the goal was to predict whether the user would make a purchase or not and, if so, which items the user would buy.

The dataset was provided by YOOCHOOSE GmbH, which offers personalization solutions for online shops. There are two relevant datasets for this thesis: the click dataset, *yoochoose-clicks.dat*; and the buy dataset, *yoochoose-buys.dat*. The clicks contain all click information about a user's session on an e-commerce website, while the buy data contain the corresponding purchases for the sessions.

Session ID	Timestamp	Item ID	Category
1	2014-04-07T10:51:09.277Z	214536502	0
1	2014-04-07T10:54:09.868Z	214536500	0
1	2014-04-07T10:54:46.998Z	214536506	0
1	2014-04-07T10:57:00.306Z	214577561	0
2	2014-04-07T13:56:37.614Z	214662742	0
2	2014-04-07T13:57:19.373Z	214662742	0
2	2014-04-07T13:58:37.446Z	214825110	0
2	2014-04-07T13:59:50.710Z	214757390	0

Table 2.1: Examples of the click dataset

In Table 2.1 presents some example lines of the dataset. A click consists of a session ID, a timestamp, an item ID and a category. A row in the dataset describes a click, and a session consists of one or more clicks.

The session ID identifies the particular session, and the timestamp indicates the time at which an item was clicked. Meanwhile, the item ID specifies the item that was clicked in this action, and the category is the context in which the item was clicked

The data file has a size of 1.5 GB and contains 33,003,944 clicks. Table 2.1 contains examples of clicks. There are 9,249,729 unique sessions in the dataset and each session ID has 3.6 clicks on average. The maximum value of clicks in a session is 200, which seems to be a hard limit given that 26 IDs reached this number of clicks. The *yoochoose-clicks.dat* has data from 2014-04-01 to 2014-09-30. The most clicks occurred in August.

[1] https://recsys.acm.org/recsys15/challenge/

The users viewed 52,739 different items in this period of time. In addition, the item with item ID 643078800 was viewed 147,419 times and was thus the most frequently considered item. There are 5,657 items that were only viewed once.

The category "S" is the most common category. It occurred 10,769,610 times, which translates to 33% of all clicks. The category "S" indicates a special offer, while a number between 1 and 12 is a real category identifier. They are the most prevalent after the category "S," and each accounts for up to 5% of all clicks. The brands then follow. If the context was a brand, then the value is an 8- to 10-digit number. The value is defined by the context of the click, although the meaning of this is not explained.

Session ID	Timestamp	Item ID	Price	Quantity
420374	2014-04-06T18:44:58.314Z	214537888	12462	1
420374	2014-04-06T18:44:58.325Z	214537850	10471	1
281626	2014-04-06T09:40:13.032Z	214535653	1883	1
420368	2014-04-04T06:13:28.848Z	214530572	6073	1
420368	2014-04-04T06:13:28.858Z	214835025	2617	1
140806	2014-04-07T09:22:28.132Z	214668193	523	1
140806	2014-04-07T09:22:28.176Z	214587399	1046	1
140806	2014-04-07T09:22:28.219Z	214586690	837	1

Table 2.2: Examples of the buy dataset

Table 2.2 offers some examples of the buy session data. A column in the dataset represents one buy in a session. A buy consists of a session ID, a timestamp, an item ID, a price and a quantity.

The file has 1,150,753 buys and 509,696 buyers (sessions). On average, buyers purchased 2.26 items each. The buyer with most purchases bought 144 distinct items. While 20.1% of the buyers bought one item, 11.5% bought two distinct items in a session. The *yoochoose-buys.dat* dataset was available for the same time period as the *yoochooseclicks.dat* dataset. The most distinct items were purchased in August. The item with item ID 643078800 was bought most often in the sessions and was the most frequently clicked item. It was purchased in 15,203 sessions, which represents 3% of all buy sessions.

The price describes the sale price of the item at the time of its purchase. The minimum value is 0, which represents a missing value. The maximum value is 334,998. However, there is no information about the currency. The average price is 1,423.

The quantity describes the number of items with one ID that were bought in a session. The minimum value is 0, which signifies a missing value. The maximum value is 30, and the average quantity is 0.6. The average is less than 1 because there are many missing values. If the data with missing values were removed, then the average would increase. However, this information is not used for the sequence models, and the data are still considered.

In the *yoochoose-clicks.dat* and *yoochoose-buys.dat* datasets, the missing values were cleansed and replaced by 0. I did not find any missing values in session ID, item ID or timestamp. There are 49.5% missing values for category and 53.0% missing values for price and quantity. I did not detect any additional errors or data anomalies.

The challenge is to predict buys with the scarce information that is available. Almost half of the data are missing from the click category, and the piece and price information are missing for half of the purchase data. Therefore, it is difficult to anticipate without the context underlying the category. Here, one can use the item click sequences as a type of context through a sequence model to obtain the purchase predictions.

Unfortunately, as external actors, we received no insight into which articles correspond to the item IDs or which number represents which category. Nevertheless, this knowledge is not necessary for the modeling, as the computer does not require it. Still, as humans, we can interpret this information; therefore, it is useful to check the plausibility and gain knowledge about the buy predictions.

2.2 Ensemble Learning based Approaches

As part of the RecSys Challenge 2015, several solutions were presented that use ensemble learning to achieve the buy prediction.

The winning solution of the RecSys Challenge employs ensemble models to predict purchases. For the solution, Romov and Sokolov [32] utilized two-stage classification, numerous categorical values and strong classifiers that were trained with gradient boosting and whose thresholds were optimized on the competition score.

I describe this solution in detail in Chapter 4 since I reproduce this approach to combine it with the sequence models that I subsequently develop.

For the challenge, Cohen et al. [6] presented two approaches with ensemble learning models. One approach, which is similar to that of Romov and Sokolov, involves an ensemble learning model of two trees. The other approach uses a tree to directly predict the items that are purchased. Cohen et al. used *Weka*'s [2] *REPTree* implementation to build the models.

2.3 Deep Learning based Approaches

Hidasi et al. [17] were the first researchers to apply recurrent neural networks (RNNs) to recommender systems. Their motivation was that recommendations in real life often have brief, session-based data, such as those in the YOOCHOOSE dataset, instead of long user histories. Thus, matrix factorization approaches are often not accurate. The solution is to recommend similar items.

Hidasi et al. [17] have proposed an RNN-based approach for session-based recommendations to model the whole as a superior alternative. They have focused their model on the top items in which a user might be interested and trained RNNs with ranking loss functions. Thus, the output of their network is the predicted preference of the items (i.e. the likelihood of being the next in the session for each item).

They used Recall@20 and mean reciprocal rank (MRR@20) as model metrics. Recall@20 is the proportion of cases with the desired item among the top-20 items in all test cases, while MRR@20 is the average of the reciprocal ranks of the desired items. The most highly performing models gained 20–30% on these scores in comparison to their implemented baselines on the RecSys Challenge 2015 dataset.

Hidasi et al. [17] have illustrated the possibility to identify user interests through item-sequence-based models. This result raises the question of whether it is possible to not only offer recommendations but also use the information for the purchase prediction.

Tan et al. [40] have further studied RNN-based models for session-based recommendations after the successful model of Hidasi et al. [17]. To improve the model's performance, they employed a variety of techniques, including data augmentation, model pretraining and generalized distillation. They have introduced a novel alternative model that directly predicts item embeddings to reduce the output dimensions. Experiments on the RecSys Challenge 2015 dataset have revealed relative improvements of 12.8% and 14.8% over the results of the RNN models of Hidasi et al. [17] on the Recall@20 and MRR@20 metrics.

Villatel et al. [43] have evaluated RNN-based models on short-term and long-term recommendation tasks. Their experimental results suggest that RNNs are capable of predicting both immediate and distant user interactions. They found that the most highly performing configuration was a stacked RNN with layer normalization and tied item embeddings. Their top model in the RecSys Challenge 2015 dataset had a Recall@20 score that was superior to those of the best models of Hidasi et al. [17] but inferior to that of Tan et al. [40].

Sheil et al. [36] have presented a multi-layer RNN for purchase detection with the RecSys Challenge 2015 dataset. Their motivation was to investigate an alternative approach to state-of-the-art methods, such as gradient boosted machines, which require sophisticated feature engineering. The researchers used trainable vector spaces (embeddings) to model input data containing categoricals, quantities and unique instances. Embeddings combined with the use of neural networks provide a learnable capacity to encode more information beyond the original numeric value of the data input. This allows the model to capture session-local and dataset-global event dependencies as well as relationships for sessions of any length. Their approach achieved 98% of the model result of the purchase detector of Romov and Sokolov [32]. Nevertheless, they have not performed any domain- or dataset-specific feature engineering like that of Romov and Sokolov [32].

The results of the RNN of Sheil et al. [36] indicate that it is possible to detect buyers and non-buyers with the sequential session data. However, they did not investigate the ability to detect item buys and non-buys with a sequential model on session data. Thus, I investigate this matter.

The previously described works have addressed item recommendation with the RecSys Challenge 2015 dataset. One study concerned purchase session detection. However, no publications on the buy prediction of items in a session with RNNs could be found during this research.

2.4 Preliminarity Experiments

I conducted my first experiments on deep learning for buy prediction as part of my master's studies. The results and findings have been published in [16].

For this purpose, I re-implemented the feature sets of Romov and Sokolov [32] and Cohen et al. [6] and then trained multilayer perceptron (MLP) models on the data.

Model	**Features**	**Accuracy**	**Precision**	**Recall**
Winning Solution RecSys 2015	Romov and Sokolov	0.77	0.16	0.77
NN 2x64	Cohen et al.	0.66	0.18	0.79
NN 2x64 FS 40	Romov and Sokolov	0.71	0.19	0.61
NN 2x128 FS 99	Romov and Sokolov	0.69	0.19	0.64

Table 2.3: Model training results

Table 2.3 displays a selection of the best results of these experiments. The name of the model is described under *Model*, and NN denotes a neural network. To distinguish between experiments, the identifier provides more details about the model. The number of hidden layers with the number of neurons per hidden layer then follows. For example, 2x64 signifies two hidden layers with 64 neurons per neural network. If a feature selection was carried out in the experiment, the abbreviation FS is included in the name followed by the number of features that were used (e.g. FS 40), as not all features were used every time to train the models. Thus, all experiments can be identified by their model name. The column *Features* indicates whether the model was trained with features from Romov and Sokolov or Cohen et al. For metrics, I utilized the model's accuracy, precision and recall. In binary classification, the accuracy is the number of correct predictions divided by the total number of predictions. The results were not adequate to surpass those of the ensemble learning model of Romov and Sokolov [32]. Their results are presented in the first line of the table. In addition, no neural network could be successfully trained on the entire feature set of Romov and Sokolov.

Since these experiments did not extend the ensemble models with the feature sets with neural networks, the idea came to try other deep learning methods that do not build on the feature sets. Since the raw data are sequences, I wanted to use sequence-based deep learning methods.

During the experiments for the MLP models, a development environment was created that was used partly for this work as well. The development environment has been described in [15].

3 Machine Learning

Machine learning algorithms differ in their approaches. Such algorithms distinguish between the type of data that they input and output as well as the type of task or problem that they are intended to solve.

In this thesis, I concentrate on supervised learning [5, p. 3]. However, there are other sub-areas of machine learning, such as unsupervised learning [5, p. 3] and reinforcement learning [39]. Supervised learning is used to construct a mathematical model for a set of data that contains the inputs and the desired outputs. The YOOCHOOSE dataset offers both through the click stream data and the corresponding buy information. Such data can be used to predict whether a user will buy an item in a session.

This prediction is a classification problem. Classification entails identifying to which of a set of categories a new observation belongs [5, p. 3]. Therefore, a training set of data that contains observations whose category membership is known is used. In the case of buy prediction, the categories/classes are buy and non-buy. Later, I encoded the classes in the data with 1 for buy and 0 for non-buy. A two-category classification is called a binary classification. If there are more classes, it is multiclass classification. If there are more classes the classification is called multiclass classification. Regression algorithms are used when the outputs may have any numerical value within a range.

Performing classification involves creating models. There are various types of models for classification. The following section presents those that were employed for this work.

3.1 Ensemble Learning

Ensemble learning builds a prediction model by combining the strengths of a collection of simpler base models. Two tasks are part of ensemble learning: developing a population of base learners from the training data and combining them to form the composite predictor. Ensemble learning is primarily used to improve the performance of a model or reduce the likelihood of unfortunate selection of a poor one [14].

The original ensemble method is Bayesian averaging. However, more recent algorithms include error-correcting output coding, bagging and boosting [7]. Ensemble

learning uses concepts such as Bayesian voting, manipulating the training examples, manipulating the input features, manipulating the output targets, and injecting randomness [7].

3.1.1 Boosting

Boosting is a committee-based learning approach that combines the outputs of many weak classifiers to produce a powerful committee [14, p. 337]. Gradient boosting is one boosting algorithm and was used by Romov and Sokolov [32] to train their purchase and item detection models.

One implementation of gradient boosting is CatBoost [47]. CatBoost uses oblivious decision trees, wherein the same splitting criterion is applied across an entire level of the tree. Such trees are balanced, less prone to overfitting and significantly increase the speed of prediction at testing time.

I use this implementation to train my purchase and item detection models as part of the redevelopment of Romov and Sokolov's models. I need these detection models to combine them with the RNNs.

3.2 Deep Learning

Ahmad et al. [1, p. 201] have noted that deep learning definitions have two key aspects in common:

1. Deep learning models consist of multiple layers or stages of nonlinear information processing.
2. Deep learning uses methods for supervised or unsupervised learning of feature representation at successively higher, more abstract layers.

Deep learning is founded on research on neural networks, artificial intelligence, graphical modeling, optimization, pattern recognition and signal processing. The main reasons for the current popularity of deep learning are the increased capabilities in processing power, the expanding size of the data that are used in model training, and recent advances in machine learning and signal/information processing research. "These improvements have enabled the deep learning methods to effectively exploit complex, compositional nonlinear functions, and also to learn distributed and hierarchical feature representations" [1, p. 201]. Deep learning is effective with both labeled and unlabeled data [1, p. 201].

Deep learning also includes methods that make it possible to train models with sequential data. This approach is relevant to models that will be trained directly with

click stream data in order to discover product purchases. Therefore, the methods that I employ in this thesis for sequence models are associated with deep learning.

3.2.1 Recurrent Neural Networks

Recurrent neural networks are a family of neural networks for processing sequential data. A recurrent neural network (RNN) can scale to much longer sequences than is practical for networks without sequence-based specialization. The key purpose of RNNs is to share parameters across various parts of a model, which allows for extending and applying the model to examples of different forms (lengths, in this case) and generalizing across them. By comparison to fully connected feedforward networks, RNNs share the same weights across several time steps [10].

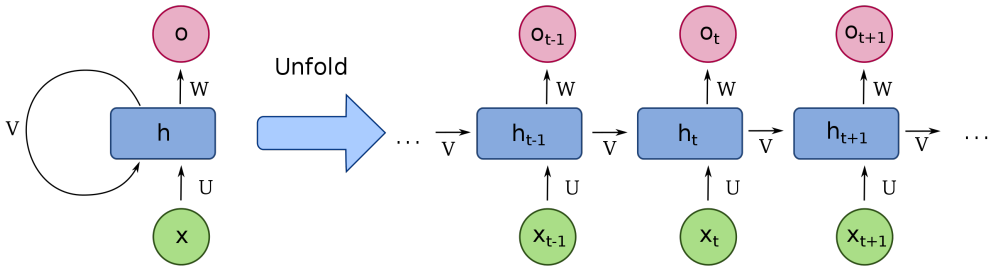

Figure 3.1: Recurrent neural network [46]

Figure 3.1 provides a computational graph of an RNN that maps an input sequence of x values to a corresponding sequence of output o values. The RNN has input-to-hidden connections parametrized by a weight matrix U, hidden-to-hidden recurrent connections parametrized by a weight matrix V, and hidden-to-output connections parametrized by a weight matrix W. The RNN with recurrent connections is on the left sight of the figure, while the right side displays the same RNN as a time-unfolded computational graph, wherein each node becomes associated with one particular time instance.

Important design patterns for RNNs that are relevant to this thesis include the following examples:

- Recurrent neural networks that produce an output at each time step and have recurrent connections between hidden units (illustrated in Figure 3.1).
- Recurrent neural networks that read an entire sequence and then produce a single output and have recurrent connections between hidden units.

The first design pattern facilitates, for example, sequence labeling (see Section 3.3), and the second one is used for sequence classification (see Section 3.4).

I use sequence labeling to label every clicked item in a session as buy or non-buy. In addition, I apply sequence classification that utilizes the entire click stream to predict whether an item will be purchased or not.

Gated Recurrent Neural Networks

The most effective sequence models for practical applications are gated RNNs. These models include the long short-term memory and networks that are based on the gated recurrent unit. Gated RNNs are founded on the notion of creating paths through time that have derivatives that neither vanish nor explode. Gated RNNs generalize this notion to connection weights that may change at each time step, and they learn to decide when to forget the old state by setting it to 0.

The initial long short-term memory (LSTM) model introduces self-loops that produce paths by which the gradient can flow for long durations [18].

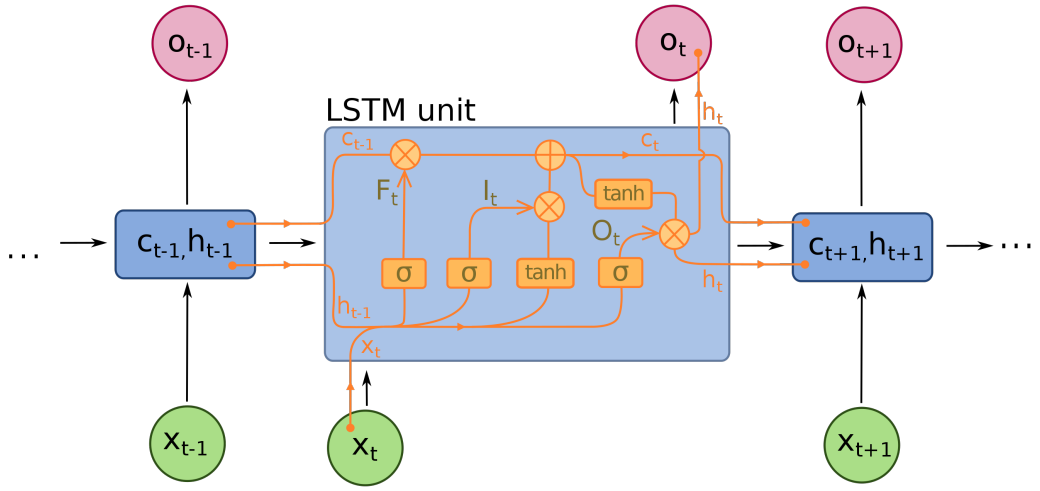

Figure 3.2: Long short-term memory [45]

Research has indicated that LSTM networks can learn long-term dependencies more easily compared to simple recurrent architectures [10, p. 411].

Another gated RNN architecture is the gated recurrent unit (GRU). Its main point of difference from the LSTM is that a single gating unit simultaneously controls the forgetting factor and the decision to update the state unit.

However, several investigations of architectural variations of the LSTM and GRU have found no variant that clearly outperforms both across a wide range of tasks [10, p. 412].

I apply the LSTM model since the architectures I use employ LSTM in their models.

Bidirectional Recurrent Neural Networks

Bidirectional recurrent neural networks (BRNNs) [35] were invented to address the need to output a prediction of $y^{(t)}$ that may depend on the whole input sequence. A BRNN can be used to label each item in a click stream by considering all of the information in the sequence instead of taking into account only the previous clicks, which is the case with a RNN. It is an interesting tool since the detection models of Romov and Sokolov also utilize all available information about a session.

A BRNN combines two RNNs: one moves forward through time from the start of the sequence, while the other moves backward through time from the end of the sequence.

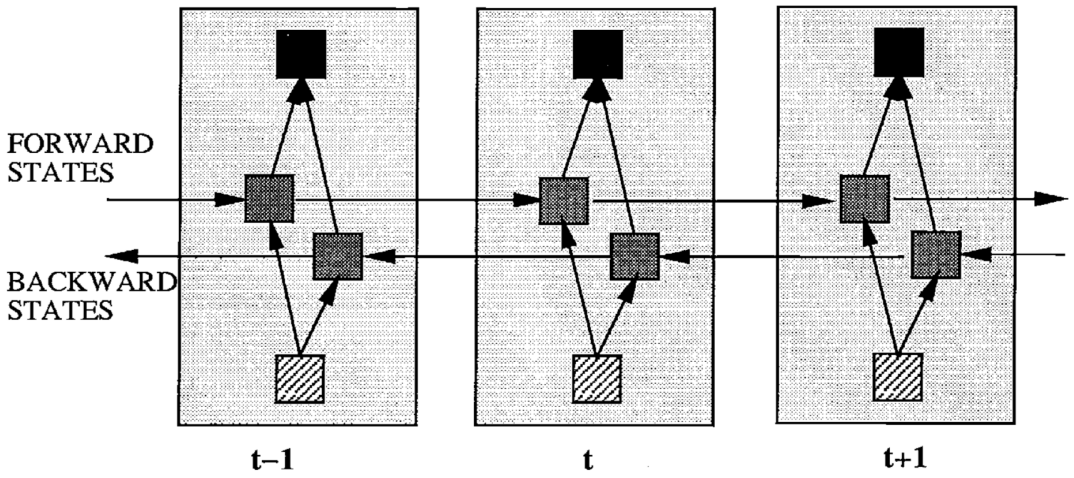

Figure 3.3: Bidirectional recurrent neural network [35]

Figure 3.3 illustrates a typical BRNN. The forward states of the sub-RNN move forward through time, and the backward states of the sub-RNN move backward through time, which allows the output units $o^{(t)}$ to compute a representation that depends on both the past and the future. The network is most sensitive to the input values around time t without having to specify a fixed-size window around t [10].

3.2.2 Dimensionality Reduction with Embeddings

An embedding is typically a low-dimensional vector with fewer dimensions than the "ambient" space, of which the manifold is a low-dimensional subset. Non-parametric manifold learning algorithms directly learn an embedding for each training example, while other algorithms learn a more general mapping, which is sometimes called an encoder or representation function, that maps any point in the ambient space to its embedding [10, p. 518].

Guo and Berkhahn [12] have introduced one method to calculate embeddings for entities. They have mapped categorical variables in a function approximation problem into Euclidean spaces, which they refer to as the entity embeddings of the categorical variables. The embedding is learned by a neural network during the standard supervised training process and not with a separate model. The embedding space represents the intrinsic properties of the categorical variables by mapping similar values proximal to each other.

Nguyen et al. [28] have proposed a method to extract co-occurrence-based item embeddings. They aimed to extract the relationships between items in the same manner as that of word embedding techniques. Therefore, they used a shared positive pointwise mutual information (SPPMI) matrix [22] of items based on co-occurrences of items in the interaction list of a user. They have argued that items that co-occur frequently in the interaction lists of certain users are similar, and their latent vectors should be close to each other in the latent space.

In addition, Barkan [3] has introduced item2vec. This neural embedding algorithm enables item-based collaborative filtering. Item2vec is based on skip-gram with negative sampling [24] with minor modifications.

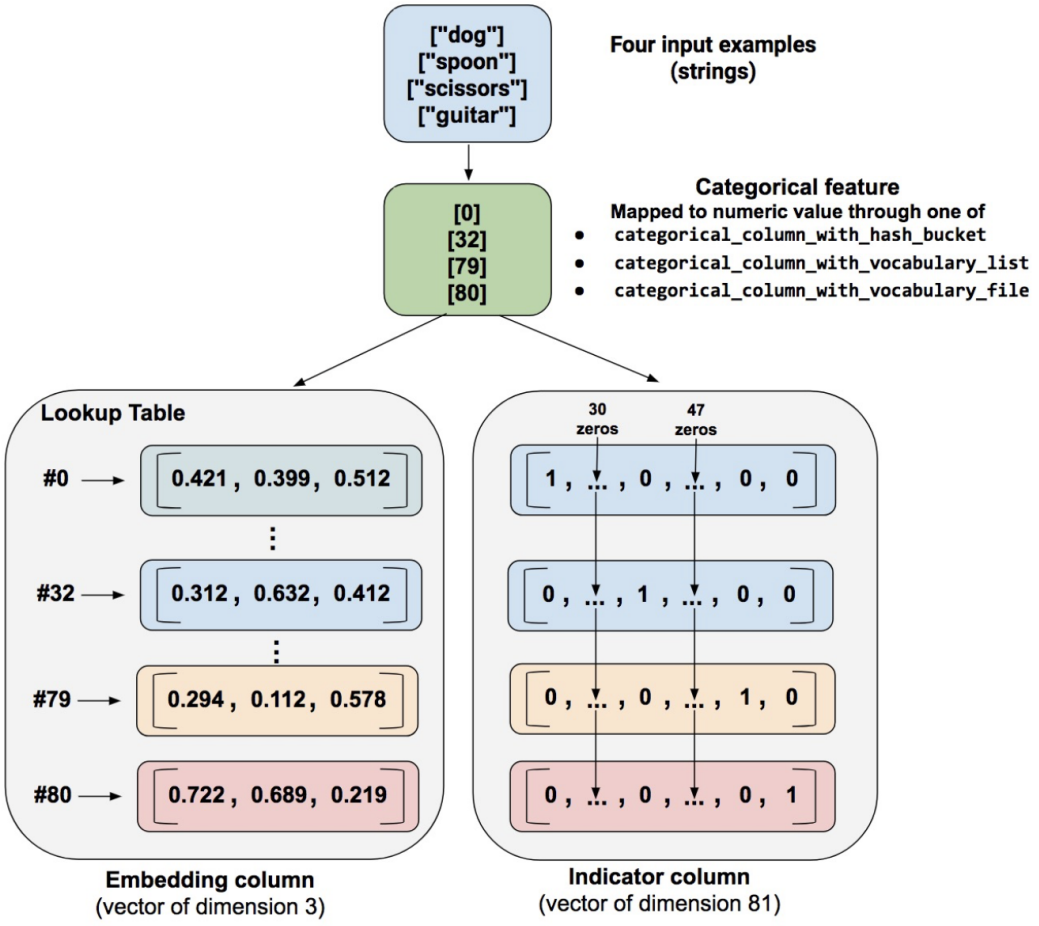

Figure 3.4: Convert categorical features to embedding vectors [42]

Figure 3.4 visualizes the use of embeddings as representations for categorical data. The input data are parsed to categorical data with a numerical value as representation. Such representation can be transformed into an embedding vector or a one-hot vector. An indicator column from TensorFlow treats each category as an element in a one-hot vector, whereby the matching category has a value of 1, and the rest have a value of 0. An embedding column represents that datum as a lower-dimensional, ordinary vector in which each cell can contain any number and is not limited to 0 or 1. The advantage over one-hot encoding is the reduced memory usage and the speed up on neural networks [12].

The concept of embeddings enable item IDs to be represented as more than a number. Although the item IDs are numerical values, the model should not treat them as such. It is not logical to add or multiply item IDs. Instead, the co-occurrence of items with

an embedding is represented. Thus, I use embeddings to obtain a meaningful input representation for my sequence models.

3.3 Sequence Labeling

Sequence labeling or sequence tagging [11] is a type of pattern recognition task that involves the algorithmic assignment of a categorical label to each element of a sequence. In natural language processing (NLP), it is employed for part of speech tagging, chunking and named-entity recognition.

Huang et al. [19] have proposed a variety of LSTM-based models for sequence tagging. Their work was the first to apply a bidirectional LSTM conditional random field (CRF) (BI-LSTM-CRF) model to NLP benchmark sequence tagging datasets. It illustrates that the BI-LSTM-CRF model can efficiently use both past and future input features because of a bidirectional LSTM component. Moreover, the model also utilizes sentence-level tag information through a CRF layer. The BI-LSTM-CRF model can support state-of-the-art and highly accurate speech tagging, chunking and named-entity recognition datasets. In addition, it is robust and less dependent on word embedding compared to other models. Lample et al. [21] have also engaged this concept in their work.

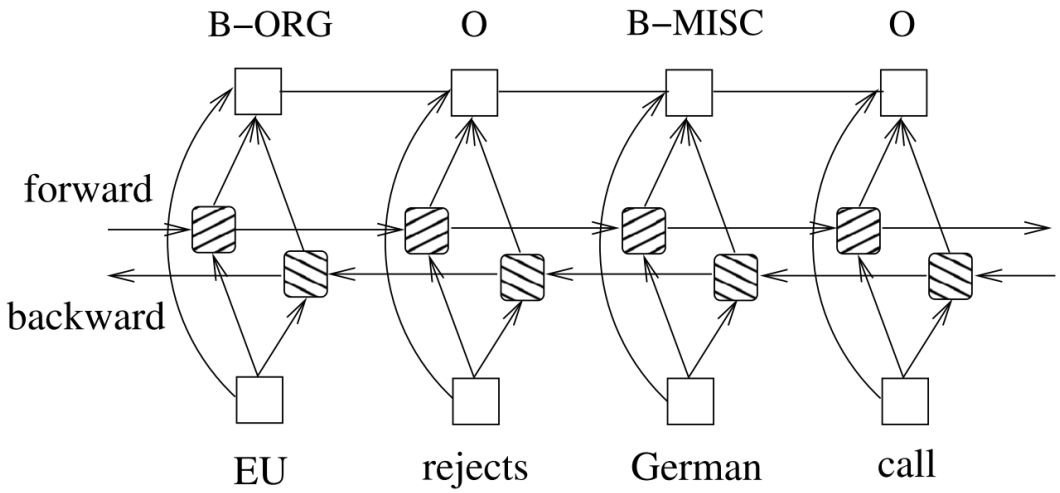

Figure 3.5: The BI-LSTM-CRF model of Huang et al. [19]

Figure 3.5 iillustrates a named-entity recognition system in which each word is tagged. Huang et al. [19] have combined a bidirectional LSTM network and a CRF

network to form a BI-LSTM-CRF network. With the bidirectional LSTM , the network can use past and future sequence information. A CRF layer is represented by lines in the figure that connect consecutive output layers, and it has a state transition matrix as parameters. With such a layer, information about past and future tags is used to predict the current tag, which is similar to the use of past and future input features via a bidirectional LSTM network.

Newer implementations of sequence labeling models employ bidirectional language models [30] or newer concepts of NLP, such as embeddings from language models (ELMo) [31] in their models.

I transfer the BI-LSTM-CRF model of Huang et al. [19] from text to click streams in my experiments. I chose the model because it does not exploit text-specific features that cannot be transferred to other sequence data. In addition, it uses the whole sequence as well as the labels through the CRF for a prognosis.

3.4 Sequence Classification

Sequence classification is a predictive modeling problem to predict a category for a sequence. In NLP, sequence classification is applied for text categorization and sentiment classification.

Time series classification is one type of sequence classification. Examples of real-world applications of time series classification include human activity recognition and acoustic scene classification. However, buy prediction is an time series classification problem as well. Smirnov and Nguifo [37] have found no work or experimental studies on the efficiency of LSTMs or other RNNs as standalone classifiers for time series classification. These researchers trained simple RNNs and LSTMs with a varying number of layers and differing layer widths for time series classification. None of their trained RNNs was able to achieve the same average results as the fully convolutional network of Wang et al. [44]. Additionally, Ismail Fawaz et al. [20] have re-implemented only one non-convolutional recurrent architecture of Tanisaro and Heidemann [41] in their deep learning for time series classification review study. Thereby, in time series classification, convolutional networks dominanted the deep learning methods.

In this thesis, I aim to develop a sequence classification model with RNNs to advance knowledge of the topic in this area. Therefore, I am oriented toward the architecture of Smirnov and Nguifo's models.

3.5 Model Evaluation

To answer the research question, I compare the models. Certain metrics can evaluate the quality of models. Here, to compare the models, I use the same metrics as Romov and Sokolov: the area under the curve (AUC), precision and recall.

These metrics use test data for which the true class is known. The prediction values of the model are compared with the true values, which results in the designations in Table 3.1.

		True class		
		p	**n**	**total**
Hypothesized class	**p′**	True Positives	False Positives	P′
	n′	False Negatives	True Negatives	N′
	total	P	N	

Table 3.1: Confusion matrix [8]

In this case, the binary classification is buy or non-buy. There are both true positives (TP) and true negatives (TN) cases. These cases refer to items in sessions that have been correctly classified as buy or non-buy and thus match their true class. Buy is the positive class. The false positivess (FPs) and false negativess (FNs) are items in sessions for which the false class was predicted. For example, if a buy item is predicted as a non-buy item, it is an FN. In addition, P is the sum of all positive samples, and N is the sum of all negative samples in the dataset. These four values form a confusion matrix, which describes the performance of a model, and are necessary to calculate the metrics. For purchase detection and item detection, I applied the metrics of precision, recall and area under the curve (AUC).

Precision [26, p. 285] is the fraction of relevant instances among the retrieved instances. With relevant instances, the binary classification refers to the positive class (in

this case, the user buys the item).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.1}$$

The recall also called true positive rate (TPR) of a classifier is estimated as

$$\text{Recall} = \text{TPR} = \frac{\text{TP}}{\text{P}} \tag{3.2}$$

Recall [26, p. 285] is the proportion of relevant instances that have been retrieved over the total amount of relevant instances.

The AUC is equal to the probability that a classifier will rank a randomly chosen positive instance more highly than a randomly chosen negative one [8]. The implementation of AUC for scikit-learn applies the trapezoidal rule to calculate the AUC. Such implementation was conducted by Romov and Sokolov as well, and I use it to compare the results. The AUC is given by

$$\text{AUC} = \int_{\text{x}=0}^{1} \text{TPR}(\text{FPR}^{-1}(\text{x}))\text{dx} \tag{3.3}$$

The false positive rate (FPR) is defined as

$$\text{FPR} = \frac{\text{FP}}{\text{N}} \tag{3.4}$$

To evaluate the item embedding, I employed the accuracy as a metric. This metric is often incorporated for balanced datasets, wherein both classes have approximately the same number of samples.

$$\text{Auccuracy} = \frac{\text{TP} + \text{TN}}{\text{P} + \text{N}} \tag{3.5}$$

4 Results and Discussion

4.1 Reproduction of Purchase Detection and Item Detection Classifiers

This experiment reproduces Romov and Sokolov's models. Such reproduction is necessary for my further experiments, which combine the models with new models to investigate the behavior of item detection with sequence models. Therefore, I consider the existing sources of their work as a basis. The sources are the paper to the challenge and a Github[1] repository with some Jupyter[2] notebooks. I first present their approach before describing my reproduction of it.

Romov and Sokolov decided to use a two-stage classifier for the buy prediction on the YOOCHOOSE dataset. They classified the buy sessions first and then, for the buyers only, what they bought. One model targets purchase detection, and the other supports item detection. Romov and Sokolov assumed this approach because the dataset is not balanced, as only 5.5% of users bought at least one item. Furthermore, they opted to work with thresholds instead of binary classes to optimize them directly.

Romov and Sokolov developed two feature sets: one with session features that describe a session and another with session-item features that describe an item in a session. A list of all features with descriptions is available in Appendix A.1. The purchase detection classifier uses only session features, and the item detection classifier employs both session features and session-item features.

As the model algorithm, they applied gradient boosting. Their feature sets have some categorical features, and such features have to be treated differently from numerical ones. Thus, they integrated the gradient boosting implementation of [49], which uses oblivious decision trees. This implementation uses *hash tables* with the *MatrixNet* [48] tool to handle categorical inputs. Since the loss function *mean squared error* [13, p. 24] tended to overfit the models, they used finally *log-likelihood* [13, p. 31]. For model training, they

[1] https://github.com
[2] https://jupyter.org

utilized 90% of the dataset, and 10% served as a validation set. The threshold values of the models were optimized with the validation set.

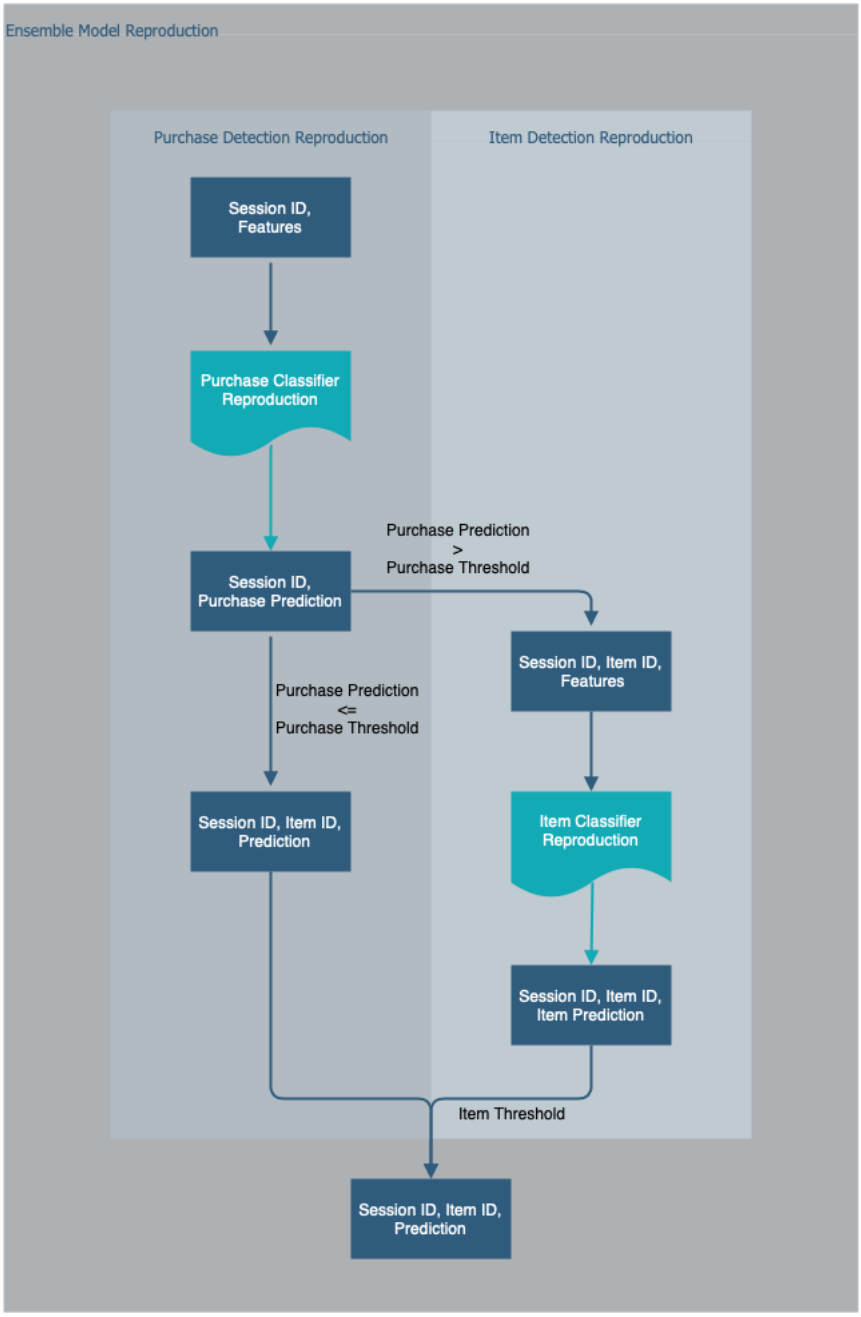

Figure 4.1: Components of the ensemble model

Figure 4.1 visualizes the reconstruction of the ensemble model, which contains the two detection models. The session features are incorporated into the purchase detection model. Only those that surpass the threshold are considered by the item detection model, which then predicts the item purchases with the session features and item-session features. In this context, the values must exceed a certain threshold to constitute a buy. Ultimately, all predictions are combined.

For this reproduction, I used the available code.[3] The repository contains code for downloading the data, calculating the features and threshold optimization, and conducting data analysis. The only missing element was the modeling. The features were calculated from the data with the Apache Spark's RDD API.[4] This API provides distributed calculation on the dataset and is therefore especially fast.

However, in this format, the features could not be given directly as features in a data mining algorithm. I transformed the data into NumPy[5] arrays and hashed the categorical features at this step. Then, I trained one model for purchase detection with the session features and the other model for item detection with the session and session-item features. At this point, I used CatBoost from Yandex for gradient boosting, which is freely accessible. The number of trees is 1,000 by default.

Model	**AUC**	**Precision**	**Recall**
Purchase Classifier	0.85	0.16	0.77
Purchase Classifier Reproduction	0.81	0.15	0.70
Item Classifier	0.89	0.75	0.85
Item Classifier Reproduction	0.86	0.72	0.81

Table 4.1: Results of the reproduced ensemble model

Five training runs were conducted for each of the replicated models. The mean values of the metrics are included in the table. Since the variance of the experiments was significantly low, it is not listed in the table.

The results are similar to those of Romov and Sokolov but slightly worse. The disparity is probably due to deviation in the implementation of the modeling.

[3] https://github.com/romovpa/ydf-recsys2015-challenge
[4] https://spark.apache.org/docs/latest/rdd-programming-guide.html
[5] https://numpy.org

4.2 Item Detection with Sequence Models

This section addresses my experiments on sequence models to investigate the influence of sequence models on buy predictions. I developed sequence models for item detection to compare their behavior toward the ensemble of Romov and Sokolov. Then, I built a new ensemble with the item detection model from the previous chapter and the sequence models, and I assessed its relation to the item detection model. In this regard, I explored whether the sequence information had an influence on the item detection. I then embedded the new ensemble model into the overall buy prediction and observed its effects.

However, before creating the sequence models, the data must be prepared as suitable input for the models, and an embedding of the items is required. This embedding is the first point of discussion in this section. Then, the individual experiments with the sequence models follow.

4.2.1 Item Embeddings

The input for the sequence models in this section are the sequences of the items. The item IDs must not be given as numbers in the network, as they are not to be treated as such but rather interpreted as categorical values. One-hot encoding [27, p. 215] is often carried out for this purpose. However, because of the number of distinct items, the one-hot vector has a dimension of 52,739 and needs a substantial amount of memory. With this dimensionality, I used item embeddings to enter the item IDs into the network. These can be learned with the network architecture; still, not all items are included in the training set of Romov and Sokolov.

One option is to adjust the training set. However, as a consequence, the models will no longer be trained on the same data, which limits the comparability and causes obstacles to the construction of the ensembles in the next step. Therefore, I trained the embeddings independently from the actual item detection models. I could then use the pre-trained embeddings in the item detection models. I chose to train a co-occurrenced item embedding based on skip-gram with negative sampling [24] but domain-specific adaptions. The model is described as follows.

I trained a net, which should decide if two item IDs occur in a sequence. For the training data, I extracted all possible tuples of item IDs from the sequences of the entire dataset. However, I did not use duplicates of tuples, as there are 28,418,581 distinct

pairs, and I encountered problems with memory in preprocessing this amount of data with pandas.[6]

When calculating the training data, the click streams and pairs that formed must be in memory, as pandas works only in memory and uses one thread. Otherwise, parallel frameworks, such as dask[7], have to be employed. These are the positive examples, while the negative examples were generated randomly from the item IDs. If the sample was not a positive example, then it was a valid negative example. Test data were created to test the model after training. To this end, the same positive examples were selected as for the training dataset. Negative examples were also randomly calculated, as for the training dataset. The two datasets can differ only in the negative examples.

The network then receives input in the form of the categorical codes of the item IDs and, as output, returns 0 or 1. The two inputs use distinct embedding layers that are trained during the training of the network. Like Guo and Berkhahn [12], I chose 50 dimensions for the embedding. The embeddings then contain information from the classification of items that occur together in sequences.

Instead of testing the item embedding, it is easier to evaluate the entire item co-occurrence model, as the embeddings of the model offer the only way to store the knowledge about co-occurrence. It cannot be stored in further layers, as the model has none. Thus, the degree to which the embedding reflects the knowledge is indirectly tested.

Model	**Accuracy**	
	Train	Test
Item Co-Occurrence	0.95	0.95

Table 4.2: Results of the item co-occurence prediction model

Table 4.2 spresents the results of the item co-occurrence model. The model accuracy is regarded as the only metric. It is a balanced dataset because the data have an equal amount of positive and negative samples. The two values are highly similar, and the training and test datasets contain the same positive samples. In the table, they are called *Train* and *Test*. However, the negative samples differ because they are random, which suggests that the model is not overfitted. Otherwise, the disparity in accuracy would be more significant. Here, only the generalization for unseen negative samples can be tested.

[6]https://pandas.pydata.org
[7]https://dask.org

This item embeddings I use for the next experiments, these are the item detection models with sequence data.

This is the necessary preprocessing of the data to model the purchase prediction with item click streams. No domain knowledge is needed, which presents an advantage over the massive feature engineering approaches, such as that of Romov and Sokolov. A substantial amount of time is saved at this point since the relevant features do not need to be determined for the prediction model.

4.2.2 Item Detection with Sequence Models

The purchased item detection recognizes purchased items from a session. Since this model follows the purchase detection model of Romov and Sokolov, the assumption is that purchased items must be detected from a session in which at least one item was purchased. In this section, alternative approaches to the item detection classifier of Romov and Sokolov are developed. These alternatives are based on sequence models from deep learning since the models should classify on the basis of sequence information.

Given a sequence, I aimed to predict a buy or non-buy of an item from that session. In the recommender systems, all 52,739 items were considered as possibilities. In the classification with 52,739 possible items, the output of the model was highly dimensional. I took advantage of the special feature of the domain and considered only those items that also occurred in the session as potential purchased products, as a user could only buy an item if he or she viewed it and it appeared in his or her click sequence. This condition diverges from the approaches of recommender systems in that the dimension of the output of the model was not 52,739. I limited the relevant items only to those that occurred in the respective session. However, if only the item sequences of the session were considered as input, then the sequences did not differ. Furthermore, different classes could not be predicted for the same input, as some items in this sequence could be bought, while others are not. Consequently, the model cannot predict a buy for one item and a non-buy for another item if it received only the click stream as input for a binary classification.

There are two options of employing the sequences and solving this prediction with sequence models. Thereby, either the input or the output is adapted. To adjust the input, the sequence was defined as input to the model in addition to the item for which the prediction will be made. A sequence classification could then be performed with this input since the inputs differ in the items of the session to be considered. The second possibility is to adjust the output. To this end, I used a sequence labeling model

to calculate a prediction for the considered item from a sequence at each time step. Therefore, the output is a sequence of classes that is predicted for each click. The output has the dimension of the input sequence. However, there may then be several predictions for an item in a sequence in the output of the model. These values must subsequently be combined into one prediction.

In this regard, two important design decisions were made. The first was to reduce the output dimension by classifying only items that were viewed by the user in the session. The second necessity was to adjust input or output to apply a sequence model to the click streams. The approaches are presented in the following sections.

Sequence Classification

In the sequence classification model, I entered the click stream and the target item for classification into the model. The model then returned output indicating the buy or non-buy status of an item in that session. Figure 4.2 illustrates this process.

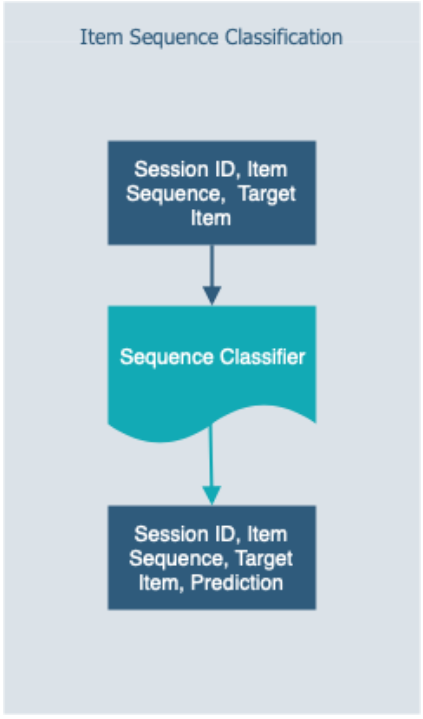

Figure 4.2: Components of the sequence classifier

For the sequence classification model, I used the *lstm 128 dense* model of Smirnov and Nguifo [37]. This model has an LSTM with 128 neurons and a dense layer as output.

The researchers used an Adam optimizer with default parameters, 500 epochs and a batch size of 64.

I adapted this model to the purchased item detection as follows. I trained on only 10 epochs and increased the batch size to 1,024 for the first classifiers to increase the iteration speed. I tried a variety of network sizes, including 128 neurons, 256 neurons and 512 neurons with various amounts of layers. For the train set and test set, I split the sessions in the same manner as Romov and Sokolov. Thus, the sets contain the same session IDs as their sets. As data input, I used the target item ID and the sequence of the item IDs of a session. The output of the model was the label of the target item ID. Therefore, there was one label per sequence. I carried out the pretrained item embedding and loaded the weights in the embedding layer of the network.

Model	**AUC**	**Precision**	**Recall**
LSTM-128	0.79	0.68	0.69
LSTM-256	0.80	0.70	0.68
LSTM-512	0.76	0.65	0.69
LSTM-256-256	0.84	0.74	0.71
LSTM-256-256-20	0.85	0.72	0.77
LSTM-256-256-256	0.50	0.00	0.00

Table 4.3: Results of sequence classifiers for item detection with batch size 1024

Table 4.3 presents the parameters and metrics of the test set. I first trained the model with one LSTM layer with 128 neurons, then with 256 neurons and, finally, with 512 neurons. The best results derived from the classifier of 256 neurons. Furthermore, I trained classifiers with two layers and three layers of 256 neurons each. The classifier with the best metric performance was that with two LSTM layers of 256 neurons each (*LSTM-256-256*). The average values of precision, recall and AUC for this model were 0.69, 0.66 and 0.79, respectively. The model from the table demonstrated the best run; those that followed had slightly worse metrics. In total, I performed five model trainings of *LSTM-256-256*.

Model	**Batch size**	**AUC**	**Precision**	**Recall**
LSTM-256-256	64	0.50	0.00	0.00
LSTM-256-256	1	0.80	0.70	0.68

Table 4.4: Results of sequence classifiers for item detection with batch size 64

Then, I tried with lower batch sizes to check if I could improve the model performance metrics. However, the models performed worse compared to the *LSTM-256-256* classifier. Table 4.4 provides the metrics of the models with batch sizes of 1 and 64.

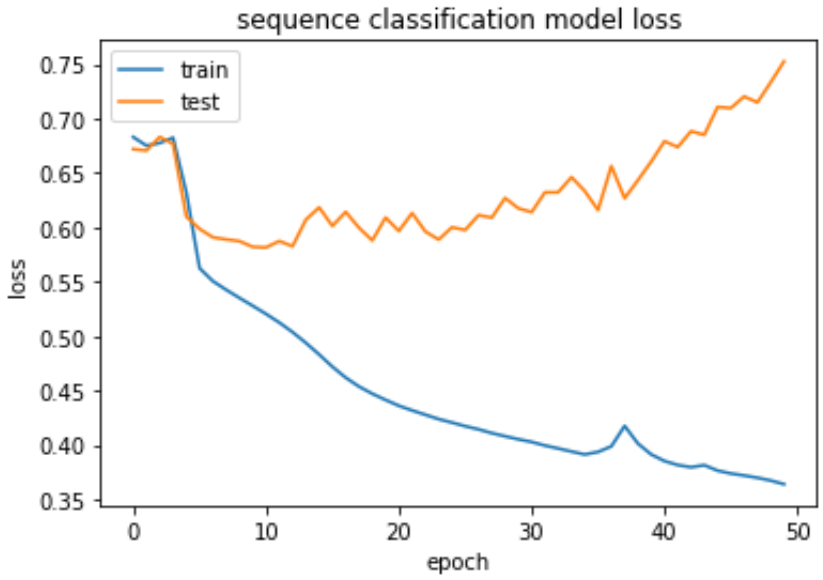

Figure 4.3: Visualization of the loss in each epoch of the sequence classifier training

Next, I checked the appropriate number of epochs for model training. For this purpose, I trained the model 50 epochs and plotted the training and validation loss in Figure 4.3. The figure conveys a minimum validation loss between 10 and 20 epochs. Based on the Keras callbacks, the minimum validation loss was 11 epochs. Therefore, I opted for my model with 10 epochs of training.

Model	**AUC**	**Precision**	**Recall**
LSTM-256-256	0.84	0.67	0.83

Table 4.5: Results of the sequence classifier for item detection with threshold optimization

Finally, I performed the threshold optimization of the classifier *LSTM-256-256* in accordance with Romov and Sokolov's models. This step further improved the recall, but the precision decreased (see Table 4.5). The results could then be compared with those of the item detection model of Romov and Sokolov. I treated this comparison later after first training a sequence labeling model for item detection.

Sequence Labeling

The sequence labeler receives the item click sequences as input and labels each item as buy or non-buy. Thus, the output sequence corresponds to the length of the input sequence. However, the items can occur more than once in a sequence and thus be assigned several labels. Therefore, the results for each session and item pair must be aggregated into a single prediction. Figure 4.4 illustrates this process.

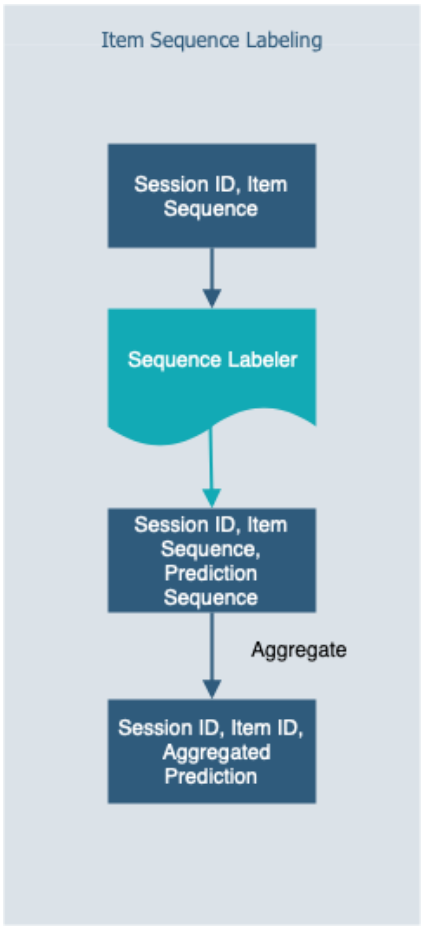

Figure 4.4: Components of the sequence labeling model

For the sequence labeling model, I referenced the model of Huang et al. [19], which has been described in Section 3.3. It is a bidirectional LSTM model with a CRF.

I adapted this model to the purchased item detection as follows. The embedding layer of the model was adapted to the pre-trained item embeddings. In addition, the weights of the embedding were loaded in the layer, and the embedding was not further

adjusted during the training. As data input, I used the sequence of the item IDs of a session, while the output was the sequence of labels. A label of 0 represents no purchase of the item at the position in the click stream, while a label of 1 signifies a purchase of the item.

The bidirectional LSTM layer was adopted, and the CRF was switched to a marginal CRF. This is needed to adjust the thresholds and utilize them as input to the ensemble model for purchased item detection.

Model	**AUC**	**Precision**	**Recall**
BI-LSTM-CRF-Marginal-100	0.87	0.69	0.74
BI-LSTM-CRF-Marginal-200	0.89	0.68	0.79
BI-LSTM-CRF-Marginal-400	0.86	0.71	0.72
BI-LSTM-CRF-Marginal-200-200	0.91	0.68	0.83
BI-LSTM-CRF-Marginal-200-200-200	0.90	0.71	0.80

Table 4.6: Results of sequence labelers for item detection with batch size 1024

First, I trained sequence labelers with 100, 200 and 400 neurons in the LSTM layer with a batch size of 1,024. Table 4.6 displays the results. The most highly performing model was *BI-LSTM-CRF-Marginal-200*, which registered an AUC of 0.89. Thus, I trained two more models with two and three LSTM layers of 200 neurons each. The best model was the *BI-LSTM-CRF-Marginal-200-200* labeler with an AUC of 0.91.

Model	**AUC**	**Precision**	**Recall**
BI-LSTM-CRF-Marginal-200-200-1	0.90	0.77	0.81

Table 4.7: Results of the sequence labeler for item detection with batch size 1

Then, I trained the *BI-LSTM-CRF-Marginal-200-200* labeler with a batch size of 1, which is the batch size that Huang et al. [19] selected for their sequence labeling model. Table 4.7 specifies the metrics of the *BI-LSTM-CRF-Marginal-200-200-1* model. The AUC value is reduced by 0.01, but the precision increases from 0.68 for the *BI-LSTM-CRF-Marginal-200-200* to 0.77. In view of this, I decided to employ the *BI-LSTM-CRF-Marginal-200-200-1* model; since the threshold optimization later further reduces the precision, I wanted to use the model with the higher precision value.

The average values of precision, recall and AUC for this model are 0.77, 0.80 and 0.90, respectively. The values of the metrics are very close to each other. I trained the *BI-LSTM-CRF-Marginal-200-200-1* model three times.

The *BI-LSTM-CRF-Marginal-200-200-1* model can label the items of a click sequence as buy or non-buy. However, the model can yield different values for the prediction of an item in a session since an item can occur more than once in the click sequence, and the sequence labeler subsequently predicts a value for each item occurrence. For the classification, I needed one prediction per session item pair. There are three ways to aggregate the predictions of an item in a session to obtain one prediction.

Model	**AUC**	**Precision**	**Recall**
MAX-200-200	0.84	0.70	0.79
AVG-RNN-200-200	0.83	0.70	0.77
MIN-RNN-200-200	0.82	0.70	0.74

Table 4.8: Results of RNNs for purchased item detection

For the aggregation model *MAX-200-200*, I took the maximum value of all predictions for an item in a session as the prediction for the session item pair. In contrast, the *AVG-200-200* assumes the average of all predictions of an item in a session as the prediction for the session item pair, while the *MIN-200-200* takes the minimum of all predictions of an item in a session as the prediction for the session item pair. Table 4.8 presents the model metrics of the three models. The metrics values are highly similar; only the recall and AUC values can be distinguished. Therefore, I decided to continue with the *MAX-200-200* model, which exhibits the highest recall and AUC.

Model	**AUC**	**Precision**	**Recall**
MAX-200-200	0.84	0.67	0.83

Table 4.9: Results of the sequence labeler for item detection with threshold optimization

The model, with the help of a sequence labeler, could calculate a prediction for each session item pair as buy or non-buy. Finally, I performed the threshold optimization for this model as well as for that of Romov and Sokolov. Table 4.9 provides the metrics. The recall improved, and the precision decreased, which was consistent with the other models.

Comparison of Item Detection Models

After implementing two approaches to item detection with sequence information, I could compare them with the item detection model of Romov and Sokolov.

Model	AUC	Precision	Recall
Item Classifier	0.89	0.75	0.85
Item Classifier Reproduction	0.86	0.72	0.81
LSTM-256-256	0.84	0.67	0.83
MAX-200-200	0.84	0.67	0.83

Table 4.10: Summary of model results

Table 4.10 contains the metrics of the various item detection models with optimized thresholds. The model *Item Classifier* refers to the model of Romov and Sokolov, and the table presents the results that they published in [32]. The second row of the table, *Item Classifier Reproduction*, displays the model metrics of my reproduction of the item detection model of Romov and Sokolov. The next row contains the metrics of the first sequence-based item detection approach with the sequence classification model *LSTM-256-256*. The last row provides the metrics of the sequence-labeling-based approach with the maximal aggregation prediction. It is called *MAX-200-200*.

The best model results still derive from the *Item Classifier* of Romov and Sokolov. However, if one compares both sequence models with my reproduction, then the two sequence models exhibit superior recall. Thus, the sequence data contributed to the recognition of a higher number of bought items compared to my reproduction. However, the precision was lower for the sequence models than for the other two classifications. In this context, the model causes not only heightened recall but also more extensive recognition of the non-buys as buys. Consequently, there are more false positives, which reduces the precision. The approaches of the sequence models seem to achieve the same acceptable results from the validation dataset, according to the metrics. In view of the values of the metrics, both approaches can facilitate item detection. These results confirm the first hypothesis that item buys depend on the clickstream and can be used for a buy prediction.

The two sequence models do not achieve better results in model metrics compared to those of the ensemble models. However, it is remarkable how closely the sequence models resemble the other models despite receiving only the item click sequence as input, which is displayed with an item embedding. Meanwhile, the ensemble models receive complex states with over 300 features as input, which also consider the times and context of the item. This model requires no domain knowledge and less time, as it involves no complex feature engineering. This finding also confirms the fourth hypothesis, which assumes that the approach requires less time for data preparation.

However, examining only the item click sequences was not sufficient to surpass such a complex model as that of Romov and Sokolov. Therefore, in the next step, I investigated whether the ensemble models could be improved by adding the item click sequence information, which is not included in the ensemble models of Romov and Sokolov.

4.2.3 Ensemble Item Detection Classifier

After developing two sequence models with the item click stream as input, I investigated the influence of this information on the item detection model of Romov and Sokolov. According to the second hypothesis, the addition of the item click streams will improve the prognosis. Therefore, I created a new model that uses the outputs of the sequence models and the reproduced item detection model as input. I implemented two models: one with the outputs of the item sequence classifier *LSTM-256-256* and another with the item sequence labeler based model *MAX-200-200*. I decided to incorporate two models as algorithms, namely a neural network and a logistic regression.

I performed the logistic regression through *scikit-learn* with the standard parameter options. As input, I used two features: the prediction of the *Item Classifier Reproduction* and either the prediction of the *LSTM-256-256* model in one model or the prediction of *MAX-200-200* in the other model. The output was the prediction for the item detection.

The neural network is designed as follows. The input is the same as in the logistic regression model. The neural network has two dense layers of 64 neurons each and relu as activation function. After each hidden layer, a dropout of 0.5 was used. The output layer had one neuron to predict a buy or non-buy of an item in a session, and the activation was the sigmoid function. I utilized binary cross-entropy as a loss function with the rmsprop optimizer. I trained the networks for 20 epochs with a batch size of 128.

Model	**AUC**	**Precision**	**Recall**
LOG-R-LSTM-256-256	0.87	0.78	0.74
NN-LSTM-256-256	0.87	0.77	0.75
LOG-R-MAX-200-200-1	0.87	0.79	0.75
NN-MAX-200-200-1	0.87	0.78	0.75

Table 4.11: Results of ensemble models for purchased item detection

Table 4.11 displays the results. The first two rows contain the metrics of the models with the predictions of *LSTM-256-256* as the input feature. The following two rows present the metrics of the models with the predictions of *MAX-200-200* as the input

feature. LOG-R represents the models with logistic regression, and NN signifies the models with neural networks as the algorithm. The AUC has the same value for all models, and the precision and recall differs by only 0.01.

Model	**AUC**	**Precision**	**Recall**
LOG-R-LSTM-256-256	0.87	0.72	0.82
NN-LSTM-256-256	0.87	0.73	0.82
LOG-R-MAX-200-200-1	0.87	0.74	0.81
NN-MAX-200-200-1	0.87	0.73	0.83

Table 4.12: Optimizied thresholds of ensemble models for purchased item detection

Table 4.12 provides the item ensemble model metrics for the models with optimized thresholds. As in the unoptimized models, the metrics are very close together. The AUC remained 0.87 for all models. The precision decreased, while the recall increased again.

Model	**AUC**	**Precision**	**Recall**
Item Classifier Reproduction	0.86	0.72	0.81
LOG-R-LSTM-256-256	0.87	0.72	0.82
NN-LSTM-256-256	0.87	0.73	0.82
LOG-R-MAX-200-200-1	0.87	0.74	0.81
NN-MAX-200-200-1	0.87	0.73	0.83

Table 4.13: Comparison of ensemble models for purchased item detection

At this point, I compare the results of the ensemble item detection models with those of Romov and Sokolov. It is interesting to compare my replication of the item detection model with the new ensemble models to gain insight into whether the addition of sequence information to Romov and Sokolov's model can lead to a more accurate prognosis.

Table 4.13 presents the results of the item classifiers and the new ensemble models for item detection. The ensemble methods utilize the prediction of the *Item Classifier Reproduction* model as the input. A comparison of the results of the item ensemble models with these results reveals improvement through the sequence information. All values were rounded up to two digits from 0.5. The AUC improved by 0.01 to 0.87 in all item ensemble models. In the models with the sequence classification predictions, the recall improved by 0.01, and it improved by 0.02 in the *NN-MAX-200-200-1* model.

Moreover, the precision improved in some of the ensemble models, and the maximum improvement was 0.02.

By comparing the results, I observed that the sequence information of the models led to a positive change in the model metrics. The precision did not improve and remained the same only in the *LOG-R-LSTM-256-256* model. In the *LOG-R-MAX-200-200-1* model, the recall did not improve and remained the same. However, the AUC improved in all models. In these experiments, the addition of the sequence information resulted in improvement in the model metrics. Thus, the second research hypothesis is confirmed.

4.2.4 Ensemble Model with the Ensemble Item Detection Classifier

With this experiment, I aimed to investigate how the new ensemble item detection models perform compared to the reproduction of Romov and Sokolov in the broader context of buy predictions. For item detection, I considered only sessions in which at least one item was purchased. However, in the overall model of Romov and Sokolov, the purchase detection first filters the sessions into buyer and non-buyer, which raises the question of how effective the new item detection models are for the ensemble of purchase and item detection. The third hypothesis expects that the prediction will improve.

Therefore, I used the implemented models of the purchase and item detection and combined the results to calculate the metrics for comparison to the reproduction model of Romov and Sokolov. I chose to continue with the ensemble item detection models based on the neural networks in this last experiment, as most metrics displayed superior values to those of the models with logistic regression. The ensemble model for buy prediction consists of the *Purchase Classifier Reproduction* and *NN-MAX-200-200-1* or *NN-LSTM-256-256*.

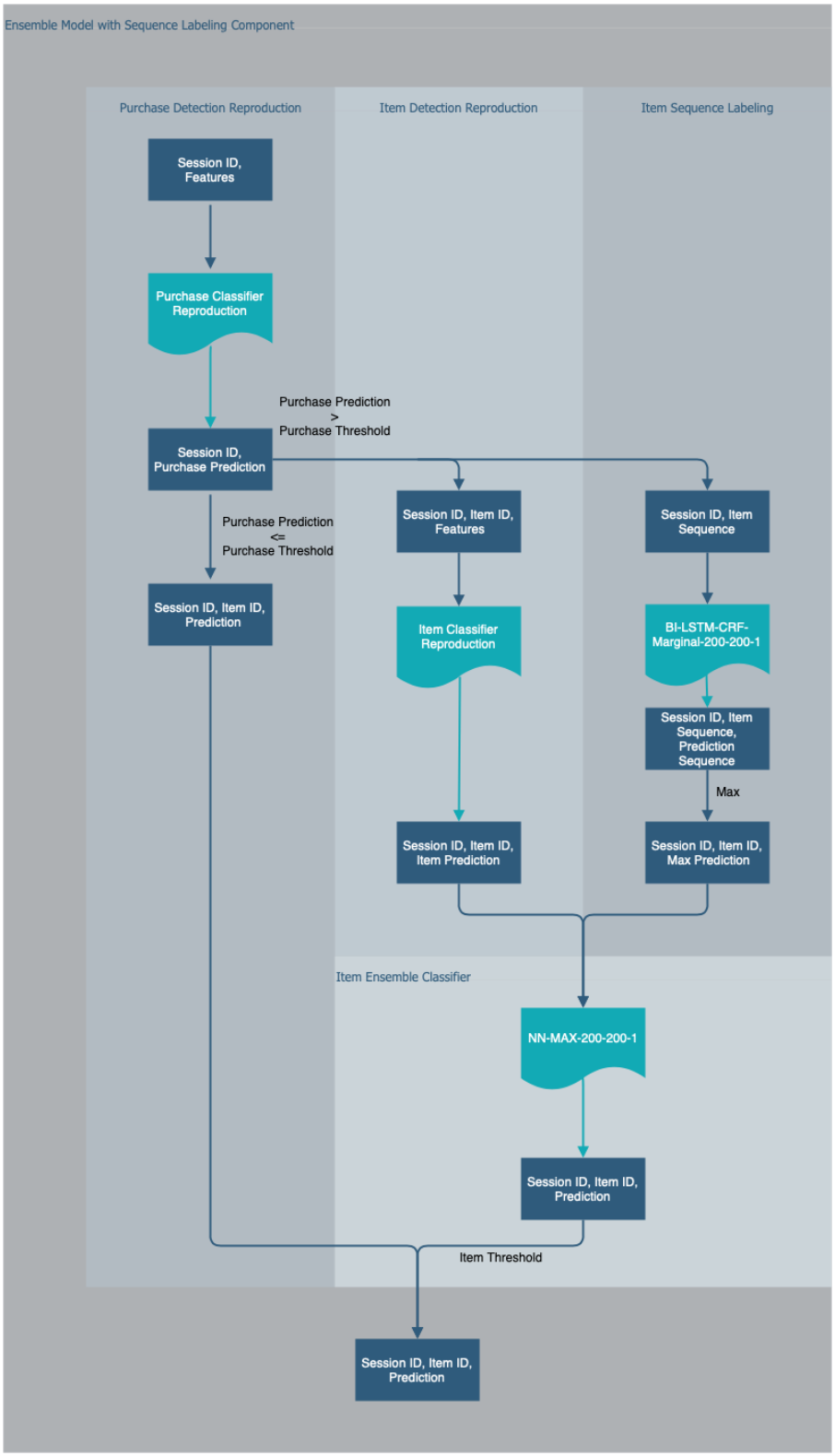

Figure 4.5: Components of the ensemble model

Figure 4.5 illustrates the combination of models into a novel ensemble model for buy prediction. First, the data were scored with the reproduced purchase detection model, which is called *Purchase Classifier Reproduction*. Then, the threshold from the threshold optimization is considered to classify the sessions as buyer or non-buyer. Only buyer-predicted sessions are included as input for the item detection models to score.

For item detection models, I used the item detection model with sequence classification or the item detection model with sequence labeling. The figure presents the ensemble model for the sequence-labeling-based model. The item detection model is, in this case, the *NN-MAX-200-200-1*. Therefore, I had to score them as positive-classified sessions with the reproduced item detection model of Romov and Sokolov (*Purchase Classifier Reproduction*). I labeled the positive-classified sessions with the item sequence labeling model *BI-LSTM-CRF-Marginal-200-200-1*. At that point, I obtained the labeled item sequences and aggregated the session item pairs to yield the maximum prediction for each pair.

The outputs of both models are the inputs for the ensemble item detection model to classify the items of the sessions as buy or non-buy. Therefore, I used the *NN-MAX-200-200-1* model and the threshold from the threshold optimization to classify as buy or non-buy. I then combined the classifications of the ensemble item detection model *NN-MAX-200-200-1* and of the reproduced purchase detection model *Purchase Classifier Reproduction* to evaluate these. For the evaluation, non-buyer classified sessions of the *Purchase Classifier Reproduction* model were also a non-buy for each item in this session. The classifications of the ensemble item detection model *NN-MAX-200-200-1* are combined with the classifications of the *Purchase Classifier Reproduction* model. Subsequently, I could calculate the metrics for the whole ensemble model.

The same procedure is performed again for the sequence-classification-based model *NN-LSTM-256-256* as the ensemble item detection model.

Model	**AUC**	**Precision**	**Recall**
Ensemble Reproduction	0.72	0.13	0.61
Ensemble SC	0.73	0.14	0.62
Ensemble SL	0.73	0.12	0.67

Table 4.14: Results of the ensemble model

Table 4.14 presents the metrics of the whole ensemble models for buy prediction. The first row indicates the metrics for the reproduced ensemble model, for which I aggregated the predictions of the *Purchase Classifier Reproduction* and *Item Classifier*

Reproduction. The second row specifies the metrics of the ensemble model with the sequence-classification-based item detector *NN-LSTM-256-256*, and the last row specifies those of the sequence-labeling-based approach *NN-MAX-200-200-1*.

The AUC improved by 0.01 in both new models. In addition, the *Ensemble SC* improved by 0.01 in precision and recall. The *Ensemble SL* improved the recall by 0.06 points but reduced the precision by 0.01. The model predicted a buy more often, which is probably because it chooses the maximum prediction as its representative. So, non-buys are also predicted as buys more frequently, and the false positives are more numerous, which reduces the precision. Nevertheless, the recall increases significantly in contrast to the decline in precision.

The small positive change due to adding the sequence information of the items to the item detection also affected the entire ensemble model of the purchase prediction. This result confirms the third hypothesis that the addition of item click streams will lead to an improvement of the prognosis.

Model	**Data**	**AUC**	**Precision**	**Recall**
Item Classifier Reproduction	Buyers	0.86	0.72	0.81
Item Classifier Reproduction	Predicted as Buyers	0.72	0.14	0.70
NN-256-256	Buyers	0.87	0.77	0.75
NN-256-256	Predicted as Buyers	0.73	0.15	0.73
NN-200-200-1	Buyers	0.87	0.78	0.75
NN-200-200-1	Predicted as Buyers	0.70	0.13	0.77

Table 4.15: Results of item detection ensembles on predicted buyer sessions compared to buyer sessions

The problem with the purchase prediction is to distinguish between and recognize buy and non-buy sessions. This problem reflects the low precision of the purchase detection models, which also impacts the ensemble models. Moreover, many false positives in the ensemble model are passed on to the item detection.

However, the models in this context were trained only on data from purchase sessions (i.e. sessions in which at least one item was bought). In this case, there was a sharp decrease in the precision of the item detection models on the dataset predicted by the purchase detection model as purchase sessions. Table 4.15 offers the metrics for the individual item detection models for the training set of the models with only buyer sessions (*Buyers* in the *Data* column) and for the set with the predicted buyer sessions

(*Predicted* as buyers in the *Data* column). The diminishing precision is a flaw of not only my models but also those of Romov and Sokolov.

An improvement suggestion is to also train the models with a few sessions in which no item was bought. This measure can make them more robust against the false positives of the purchase detection.

Model	**Data**	**AUC**	**Precision**	**Recall**
LSTM-256-256	Buyers	0.84	0.67	0.83
LSTM-256-256	Predicted as Buyers	0.62	0.05	0.71
MAX-200-200-1	Buyers	0.84	0.67	0.83
MAX-200-200-1	Predicted as Buyers	0.45	0.04	0.98

Table 4.16: Results of sequence based item detection on predicted buyer sessions compared to buyer sessions

Especially with sequence-based models, this leads to poor metrics, as Table 4.16 demonstrates. The model with the sequence labeling method did not cope with the false positives. Thus, the precision decreased, and the AUC was strongly reduced.

4.2.5 Summary of Results

The experiments have illustrated that a sequence-based model can produce good purchase predictions, which confirms the first hypothesis

This information has not been included in the features that Romov and Sokolov used for their predictions, as its addition has improved the results of their approach. The improvement in item detection confirms the second hypothesis, while that in the ensemble model confirms the third hypothesis.

An advantage compared to their approach is a reduction in the preprocessing of the data since only one embedding has to be created for the items, and no complex feature engineering is performed. Because of these differences, it requires less time, which confirms the fourth hypothesis.

5 Conclusion

The thesis has addressed the buy prediction of items with the help of sequence models on click streams of an online shop. The dataset was published in the frame of the RecSys Challenge Challenge 2015 by YOOCHOOSE and ACM. It informed an ensemble model by Romov and Sokolov, who developed a model for purchase detection as well as a model for item detection. However, the exact click sequences of the sessions were not taken into account in the models. Therefore, I have formulated hypotheses to examine in this thesis. The first hypothesis investigated whether buying behavior is reflected in the item click streams and if such information can be used to predict item purchases. The second hypothesis was that a prediction of item purchases would improve upon adding the item click stream information to the existing model, if it did not previously contain this information. To do so, I wanted to rebuild the winning Romov and Sokolov model and subsequently combine it with the sequence models that I developed. In this context, I have explored whether the item detection of Romov and Sokolov can be enhanced by adding the item click streams and whether such measure improves the overall buy prediction of their approach. This was the third hypothesis. The fourth hypothesis was that the preparation of the input data would be less time-consuming for sequence-based models than for state-based models.

The first insight could be gained by developing the sequence-based models for item detection. The models demonstrated that the inclusion of click streams could inform predictions about the purchase of a clicked item. Therefore, I used sequence classification and sequence labeling approaches. The model results closely resemble those of Romov and Sokolov's item detection model [32] even though they are based on significantly less information. This outcome confirms the assumption that the choice to buy an item depends on the click stream of the items in a session, which reflects the first hypothesis. Next, my results indicate that item detection according to Romov and Sokolov's approach can be improved by including sequence information in the prediction with an ensemble approach. Finally, the research has illustrated that the inclusion of item click streams in item detection improved the overall buy prediction in accordance with Romov and

Sokolov's approach. Although the improvements in the results in both experiments are not significant, the results confirm the second and third hypotheses.

The results evidence the potential to make predictions for an item purchase on the basis of only item click streams. Such ability would avoid the expense of feature engineering for the time being if it does not depend on top performance. However, the restriction is that the model trainings used data from only those sessions in which at least one buy took place. This limitation had a negative effect on the results once the models received sessions without a purchase for the scoring. The next step is to train a sequence model that also considers sessions without buys during its training to distinguish more effectively between the buy and non-buy of an item.

However, preprocessing has been substantially reduced. Consequently, feature engineering is no longer necessary, and only one embedding must be created for the items. These outcomes present a major advantage for prediction, as no extensive domain knowledge is required, and less time is consumed. These findings confirm the fourth hypothesis.

5.1 Transferability of Results

The item detection procedures can be applied to any online shop that records and stores the item IDs of the click stream. The portability includes the models as well as the preprocessing of the item IDs as represented by embeddings. Since the item IDs were considered as categorical values in this case, strings can also be used as identifiers and not strictly numbers. Co-occurrence-based embeddings can also be calculated from the data. Therefore, they are realizable within the domain.

The models for prediction can be transferred to all sequence data from other domains since no domain-specific knowledge is required for the models. However, depending on the type of data, the preprocessing may need to be adapted; thus, it is possible that embeddings must be calculated differently, or embeddings as representations are not needed. The preprocessing is highly specific and always individual according to the type of data. However, no statement can be made regarding the effectiveness of the prediction models in other domains. It would be interesting to investigate how sequence-based deep learning methods are used for predictions in other domains.

5.2 Outlook

It could also be insightful to examine how accurately buys can be predicted for items with sequence models without having to perform a purchase detection and train a model

for the entire task. Such a sequence model could then be directly used live in an online shop and yield a prediction for buys based on the items that have been clicked so far. However, this would have to be evaluated to allow the model to predict the purchase of items on the partial sequences. If this sequence model works, it presents the advantage of avoiding any large preprocessing of the data for features, which is a necessary step of Romov and Sokolov's approach.

Another question that emerged is how to incorporate more feature sequences. Thus far, my models have only examined item click streams, though they still record categories and timestamps that can provide sequence information. These sequences would enrich the models with further click information and possibly enhance prediction results. Other datasets of click streams with further features would be an interesting resource for investigating whether this information improves buy predictions.

Bibliography

[1] Jamil Ahmad, Haleem Farman, and Zahoor Jan. Deep Learning Methods and Applications. *SpringerBriefs in Computer Science*, pages 31–42, 2019. ISSN 21915776. doi: 10.1007/978-981-13-3459-7_3.

[2] Machine Learning Group at the University of Waikato. Weka 3: Data mining software in java, 2019. URL https://www.cs.waikato.ac.nz/ml/weka/. Accessed 2019-03-15.

[3] Oren; Noam Koenigstein Barkan. Item2Vec: Neural Item Embedding for Collaborative Filtering. pages 1–6, 2016. ISSN 16130073. doi: 1603.04259.

[4] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J. Christopher Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 2011. ISSN 01679236. doi: 10.1016/j.dss.2010.08.008.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[6] Nadav Cohen, Adi Gerzi, David Ben-Shimon, Bracha Shapira, Lior Rokach, and Michael Friedmann. In-House Solution for the RecSys Challenge 2015. *Proceedings of the 2015 International ACM Recommender Systems Challenge on - RecSys '15 Challenge*, pages 1–4, 2015. doi: 10.1145/2813448.2813519. URL http://dl.acm.org/citation.cfm?doid=2813448.2813519.

[7] Thomas G. Dietterich. Ensemble Methods in Machine Learning. pages 1–15, 2000. doi: 10.1007/3-540-45014-9_1. URL http://link.springer.com/10.1007/3-540-45014-9{_}1.

[8] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8): 861–874, 2006. ISSN 01678655. doi: 10.1016/j.patrec.2005.10.010.

[9] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–37, mar 1996. ISSN

2371-9621. doi: 10.1609/AIMAG.V17I3.1230. URL https://www.aaai.org/ojs/index.php/aimagazine/article/view/1230.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016. URL http://www.deeplearningbook.org.

[11] Alex Graves. Sequence Labeling using Recurrent Neural Networks. *Lancet*, 346 (8988):1501, 1995. ISSN 01406736.

[12] Cheng Guo and Felix Berkhahn. Entity Embeddings of Categorical Variables. (1): 1–9, 2016. URL http://arxiv.org/abs/1604.06737.

[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, second edition, 2009. ISBN 978-0-387-84857-0.

[14] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2017. URL https://web.stanford.edu/{~}hastie/Papers/ESLII.pdf.

[15] Tasmin Herrmann. Development Environment for Buy Predictions with Machine Learning. 2018. URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2018-proj/herrmann.pdf.

[16] Tasmin Herrmann. Neural Networks for Buy Prediction. 2019. URL https://users.informatik.haw-hamburg.de/~ubicomp/projekte/master2019-proj/herrmann.pdf.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based Recommendations with Recurrent Neural Networks. nov 2015. URL http://arxiv.org/abs/1511.06939.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, nov 1997. ISSN 08997667. doi: 10.1162/neco.1997.9.8.1735. URL http://www.mitpressjournals.org/doi/10.1162/neco.1997.9.8.1735.

[19] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF Models for Sequence Tagging. 2015. URL http://arxiv.org/abs/1508.01991.

[20] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre Alain Muller. Deep learning for time series classification: a review.

Data Mining and Knowledge Discovery, pages 1–44, 2019. ISSN 1573756X. doi: 10.1007/s10618-019-00619-1.

[21] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural Architectures for Named Entity Recognition. 2016. URL http://arxiv.org/abs/1603.01360.

[22] Omer Levy and Yoav Goldberg. Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 1–9, 2014. URL http://u.cs.biu.ac.il/{~}nlp/wp-content/uploads/Neural-Word-Embeddings-as-Implicit-Matrix-Factorization-NIPS-2014.pdf.

[23] Robin Van Meteren and Maarten Van Someren. Using Content-Based Filtering for Recommendation. *ECML/MLNET Workshop on Machine Learning and the New Information Age*, 2000. ISSN 15506606. doi: 1011255743.

[24] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *Proceedings of NIPS 2013*, pages 3111–3119, 2013. URL http://arxiv.org/abs/1310.4546.

[25] R Keith Mobley. Predictive Maintenance. In *Plant Engineer's Handbook*. 2001. ISBN 978-0-8493-3598-3; 978-1-4200-5618-1. doi: 10.1016/B978-075067328-0/50052-5.

[26] A.C. Müller and S. Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016. ISBN 9781449369897.

[27] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python*. O'Reilly Media, Inc., third edition, 2017. ISBN 978-1-449-36941-5.

[28] Thai Binh Nguyen, Kenro Aihara, and Atsuhiro Takasu. Collaborative item embedding model for implicit feedback data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10360 LNCS:336–348, 2017. ISSN 16113349. doi: 10.1007/978-3-319-60131-1_19.

[29] Charles Nyce. Predictive Analytics White Paper. *American Institute for Chartered Property Casuality Underwriters*, page 16, 2007. URL http://www.theinstitutes.org/doc/predictivemodelingwhitepaper.

pdf{%}5Cnhttp://ieg-sites.s3.amazonaws.com/sites/4e70a00a3723a839c1000042/contents/content{_}instance/4ec268ce3723a856ba00015c/files/PredictiveModelingWhitepaper.pdf.

[30] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. 2017. URL http://arxiv.org/abs/1705.00108.

[31] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. 2018. URL http://arxiv.org/abs/1802.05365.

[32] Peter Romov and Evgeny Sokolov. RecSys Challenge 2015. In *Proceedings of the 2015 International ACM Recommender Systems Challenge on - RecSys '15 Challenge*, pages 1–4, New York, New York, USA, 2015. ACM Press. ISBN 9781450336659. doi: 10.1145/2813448.2813510. URL http://dl.acm.org/citation.cfm?doid=2813448.2813510.

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th . . .*, 2001. ISSN 09501991. doi: 10.1145/371920.372071.

[34] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. Elsevier, 1999.

[35] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. ISSN 1053587X. doi: 10.1109/78.650093.

[36] Humphrey Sheil, Omer Rana, and Ronan Reilly. Predicting purchasing intent: Automatic feature learning using recurrent neural networks. *CEUR Workshop Proceedings*, 2319, 2018. ISSN 16130073.

[37] Denis Smirnov and Engelbert Mephu Nguifo. Time Series Classification with Recurrent Neural Networks. *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, pages 1–8, 2018.

[38] statista. E-commerce worldwide, 2019. URL https://www.statista.com/outlook/243/100/ecommerce/worldwide?currency=eur. Accessed 2019-08-06.

[39] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

[40] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved Recurrent Neural Networks for Session-based Recommendations. 2016. doi: 10.1145/2988450.2988452. URL http://dx.doi.org/10.1145/2988450.2988452http://arxiv.org/abs/1606.08117.

[41] Pattreeya Tanisaro and Gunther Heidemann. Time series classification using time warping invariant Echo State Networks. *Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016*, pages 831–836, 2017. doi: 10.1109/ICMLA.2016.166.

[42] TensorFlow Team. Google developers: Introducing tensorflow feature columns, 2017. URL https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html. Accessed 2019-06-17.

[43] Kiewan Villatel, Elena Smirnova, Jérémie Mary, and Philippe Preux. Recurrent Neural Networks for Long and Short-Term Sequential Recommendation. (October), 2018. URL http://arxiv.org/abs/1807.09142.

[44] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. *Proceedings of the International Joint Conference on Neural Networks*, 2017-May:1578–1585, 2017. doi: 10.1109/IJCNN.2017.7966039.

[45] Wikipedia. Long short-term memory unit, 2019. URL https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Long_Short-Term_Memory.svg. Accessed 2019-08-22.

[46] Wikipedia. Unfolded basic recurrent neural network, 2019. URL https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Recurrent_neural_network_unfold.svg. Accessed 2019-08-22.

[47] Yandex. Catboost, 2019. URL https://catboost.ai. Accessed 2019-05-29.

[48] Yandex. Matrixnet: New level of search quality, 2019. URL https://yandex.com/company/technologies/matrixnet/. Accessed 2019-03-15.

[49] Yandex. Yandex, 2019. URL https://yandex.com/company/. Accessed 2019-03-15.

A Appendix

A.1 Feature Engineering

Feature Description	**Number/Type**
Numerical time features of the start/end of the session (month, day, hour, minute, second, etc.)	2×7 Num
Categorical time features of the start/end of the session (month, day, month-day, month-day-hour, hour, minute, weekday)	2×7 Categ
Length of the session in seconds	1 Num
Number of clicks, unique items, categories and item-category pairs in the session	4 Num
Top 10 items and top 5 categories by the number of clicks in the session	15 Categ
IDs of the first/last item clicked at least k = 1, 2 . . . , 6 times in the session	12 Categ
Vector of click numbers and total durations for 100 items and 50 categories that were the most popular in the whole training set	150×2 Num

Table A.1: List of session features used in models of Romov and Sokolov [32]

Feature Description	**Number/Type**
Item ID	1 Categ
Total and relative number of clicks in the session for the given item	2 Num
Numerical time features of the first/last click on the item (month, day, hour, minute, second, etc.)	2×7 Num
Categorical time features of the first/last click on the item (month, day, month-day, month-day-hour, hour, minute, weekday)	2×7 Categ
Number of seconds between the first and the last click on the item	1 Num
Total duration of the clicks on the item in the session and of all item's categories seen in the session	2 Num
Number of unique categories seen in the session for a given item	1 Num

Table A.2: List of paired session-item features used in models of Romov and Sokolov [32]

Glossary

ACM The Assocoation for Computing Machinery is an organization which brings together computing educators, researchers, and professionals to inspire dialogue, share resources, and address the field's challenges..

API An application program interface is a set of routines, protocols, and tools for building software applications..

RDD The resilient distributed dataset is the main abstraction of a Spark application and contains collection of elements partitioned across the nodes of the cluster that can be operated on in parallel..

RecSys Challenge The RecSys Challenge is held within the RecSys Conference every year and deals with current topics of recommender systems..

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Masterarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Deep Learning basierte Kaufprognose mit Sequenz Modellen

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original