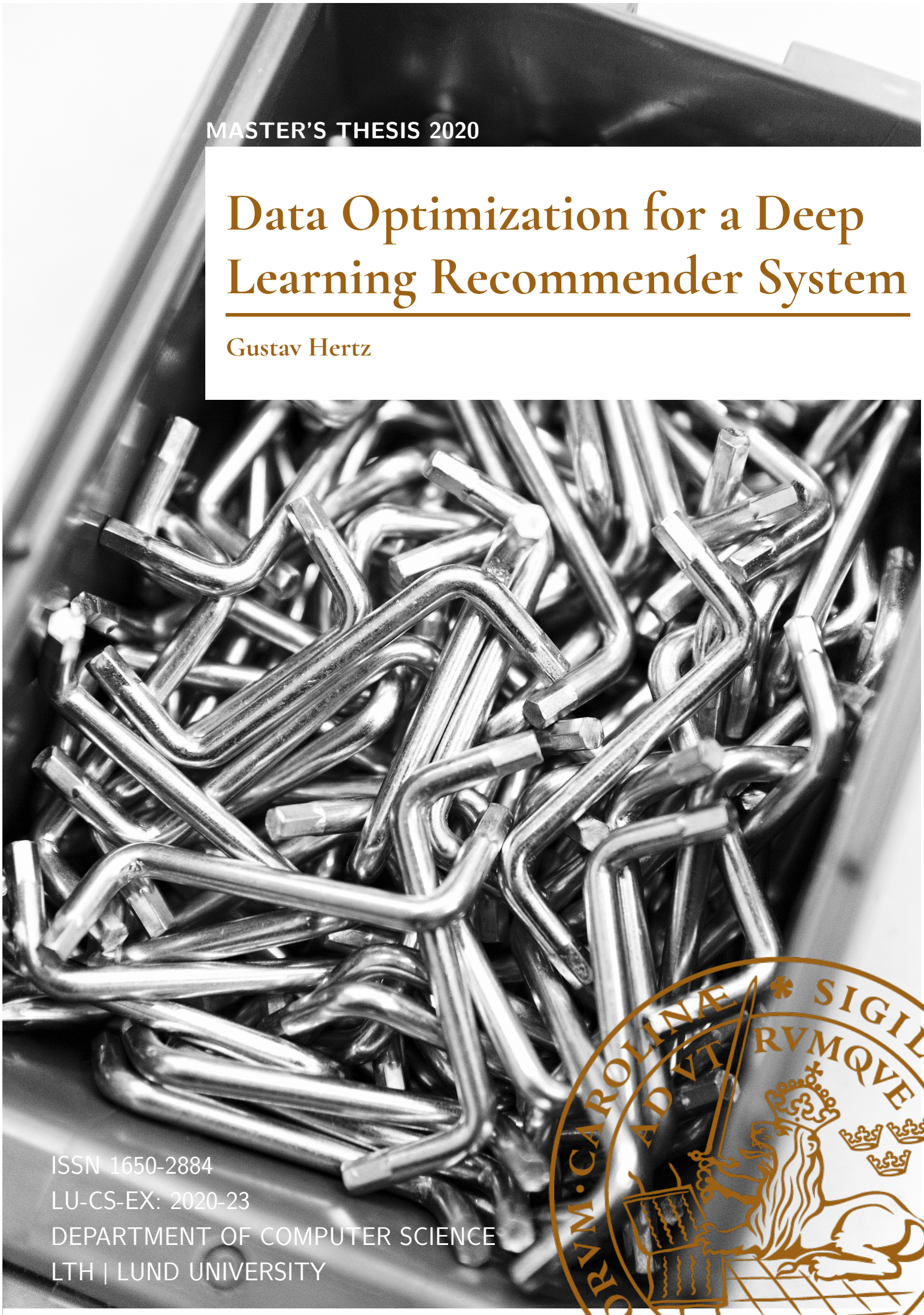


MASTER'S THESIS 2020

Data Optimization for a Deep Learning Recommender System

Gustav Hertz

ISSN 1650-2884
LU-CS-EX: 2020-23
DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-23

Data Optimization for a Deep Learning Recommender System

Gustav Hertz

Data Optimization for a Deep Learning Recommender System

Gustav Hertz
gust.hertz@gmail.com

June 11, 2020

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Patrik Persson, patrik.persson@cs.lth.se
Emil Joergensen, emil.joergensen@ingka.ikea.com

Examiner: Flavius Gruian, flavius.gruian@cs.lth.se

Abstract

This thesis investigates the performance of a deep learning recommender system based on the given training data. The recommender system used in this thesis a Long-Short Term memory network for sequence prediction of item sequences, i.e purchases at an e-commerce website. First, the performance of the recommender system, in this case, is defined as a combination of Precision, Catalog Coverage, and Novelty. Then the performance as a function of the available amount of training data is investigated. It is concluded that depending on how one values the given performance metrics, there is an optimal amount of training data, and increasing the size of the training dataset might reduce performance. Secondly, how to handle an excess of training data is investigated, concluding that more recent data leads to better recommender system performance. Finally, how to use data from secondary markets when there is a lack of data for training the recommender system is investigated. A similarity metric for purchase data between markets is proposed and results are promising, although more research is needed on this topic.

Keywords: MSc, LSTM, recommender systems, IKEA, data ethics, deep learning

Acknowledgements

I would like to thank Emil Joergensen, Balázs Toth, Sandhya Sachidanandan, and Martin Tegner and the rest of the Data Science team at IKEA for the support and the great patience during the work with this thesis.

I would also like to thank Patrik Persson for the great discussions during the process as well as valuable feedback on the report.

Gustav Hertz

Lund, Sweden, 20th May 2020

Terminology

In this thesis we use some terms and abbreviations that require a short explanation.

- **RS**: Recommender System
- **LSTM**: Long Short Term Memory network, briefly described in section 2.3
- **GCP**: Google Cloud Platform, an service provided by Google for cloud computing and storage.

Contents

Chapter 1
Introduction

Recommender systems (RSs) for personalized content are a huge part of our online experience today. They are algorithms and techniques that suggest what items might be of interest to a particular user. The items suggested can be anything; products to purchase, links to click on or music to listen to. Examples of when these algorithms has been used are personalized playlists of songs on Spotify or *"Customers who bought items in your cart also bought"*-list on Amazon [36].

The idea of RSs originates from the observation that individuals tend to rely on recommendations from others when making decisions [36]. This is particularly important when the number of available options is too large for the average user to be able to consider them all, which is the case for services such as Youtube, Netflix, etc. From an e-commerce perspective, this means showing products to users which they probably would not have found themselves but are likely to appreciate. A successful RS in e-commerce helps customers make better decisions online and thereby increase user satisfaction while also increasing revenue.

In this thesis, we study how to optimize the training data for an RS used in production at the IKEA online store. This particular RS is based on a type of recurrent neural network known as a Long Short Term Memory (LSTM) network. This type of deep learning architecture is specialized in sequence prediction, and in this implementation, it predicts the continuation of product sequences. Every purchase is modeled as a sequence of items. The model is trained to, from a sequence of items that have been added to the cart, predict what item that is most likely to be the customer's next purchase. This is a type of RS that does not require extensive user profiling, as it can generate recommendations based on only data from the current user session.

RSs for personalized recommendations are in general based on data generated by the users, such as browsing patterns, purchase history, and item ratings [36]. This poses some ethical difficulties as the data collection, what the data is used for, and its potential side effects have surpassed the understanding of the average consumer. Privacy has been one of the areas that have been discussed in connection to the ethics of data collection [44].

IKEA and Ingka Group have taken a unique standpoint on this matter where they promise

to provide customers with control and accessible information about how data is handled, giving customers the ability to make informed decisions about their data [15]. Customers should be aware of what data is being collected and for what purpose while having the ability to deny data collection and thereby personalization of their shopping experience. When giving customers the option to opt-out of personalized content and the collection of their data, it is reasonable to assume that some customers will, and as consequence, the amount of relevant collected user data will decrease. The user data is necessary to build well-functioning RSs, and a decrease of the available amount of data poses a potential hindrance to the development of these.

It is widely assumed that the performance of deep learning networks, such as the LSTM used in this thesis, is an increasing function of the amount of available data to train the network on [17]. However, the performance of RS is not only a question of prediction accuracy, as the most accurate recommendation might not be the most useful one. An example of this can be taken from movie recommendations; If a user has watched *Harry Potter and the Philosopher's Stone* the most accurate recommendation might be the sequel, *Harry Potter and the Chamber of Secrets*, but it is likely that the user was already aware of this movie and planned on watching it anyway. This makes the recommendation accurate without being useful and providing any value for the user or increased revenue for the provider. Therefore other metrics for evaluating RSs, focusing on enhancing both customer experience and revenue, also have to be considered such as catalog coverage and novelty [10, 3].

On the other hand, if metrics such as novelty and catalog coverage were to be maximized, randomly sampled recommendations from the catalog would be the best possible RS. These recommendations would be very novel and cover the entire catalog, but most likely useless and without any accuracy at all. It is apparent that there is a trade-off between accuracy-metrics and diversity metrics such as novelty and catalog coverage [43].

In this thesis, with IKEAs ambitious standpoint regarding data ethics as the main motivator, we investigate how the performance of this particular RS is affected by the data used for training, hoping to answer how to handle both excess and shortage of relevant user data.

1.1 Research Questions

Giving customers control of their data and giving them the option to opt-out means less feedback and impacts the training of models. The main research questions to be answered in this thesis can thereby be formulated as below.

- **RQ1**: How is performance of this particular RS affected by the amount of available training data?

Given that there seems to be a trade-off between accuracy focused metrics and diversity-focused metrics which are both included in what we consider good RS performance we try to answer:

- **RQ2**: Is there an optimal amount of training data that gives the best overall RS performance?

And in relation to these question we also try to answer the two following research questions:

- **RQ3**: If there is more data available on a certain market than necessary to reach sufficient performance, is it possible to select a more relevant subset of this data to achieve even better performance?
- **RQ4**: If there is less data available on a certain market than necessary to reach sufficient performance, is it possible to complete the data using data from a secondary market?

1.2 Contributions

This thesis aims to make the following contributions:

- Evaluate the performance of an LSTM recommender system as a function of the available training data.
- Investigate how to select the most useful subset of a dataset when there is an excess of training data for a recommender system.
- Propose a metric for measuring similarities between markets, based on purchase behavior.
- Evaluate the possibility of using data from secondary markets to build recommender systems in markets where there is a lack of data.

1.3 Related Work

1.3.1 Recommender Systems

In this work the performance of RS as a function of the available training data is investigated. The field of RSs is huge and a vast number of approaches and algorithms exists.

With the rise of the internet for personal use in the early 1990s the need for RSs also increased. The idea in the earliest stages was that humans undoubtedly rely on the opinions of our peers in decision-making and that the internet needed help in navigating increasing information overflow. In light of this, the early RSs were developed, starting with various variations of collaborative filtering algorithms. Early examples are Tapestry [11] in 1992 for filtering among incoming emails and Group Lens [35] in 1994 designed to recommend online news articles. Collaborative filtering aims to predict how a user will appreciate an item based on how other users with similar tastes have appreciated the item before. Which users who can be considered similar is determined by similarities in earlier usage patterns, such as similar ratings of other items.

Collaborative filtering algorithms exist in many variations and have significantly developed since the early days, with one notable implementation being the item-item collaborative filtering algorithm used to produce product recommendations at Amazon [27] from 2003.

In 2006 Netflix released a dataset containing 100M movie ratings and putting a 1M USD reward for the team that could produce the most accurate RS based on this dataset. The competition sparked the development of RSs and various collaborative filtering algorithms in focus with more than 20 thousand teams registering for the competition [2].

In 2004 matrix factorization (MF) for collaborative filtering was introduced [20, 29], further improving the accuracy of collaborative filtering types of RSs. Matrix factorization techniques compared to classic collaborative filtering have the benefits of higher accuracy, constant time prediction, and compact model representation. There are several types of matrix factorization models, such as Singular Values decomposition (SVD), Probabilistic Matrix Factorization (PMF), and Principal Component Analysis (PCA). The core idea behind matrix factorization is to factorize the user-item matrix containing information about interactions between all users and items into a user-matrix with a set of k features for each user and an item-matrix with k features for each item. Using the user-matrix and item-matrix it is then possible to compute predictions for how each user will appreciate each item.

In 2007 the use of Restricted Boltzmann Machines, a generative artificial neural network, for collaborative filtering was introduced, with the motivation that most existing collaborative filtering algorithms struggle to handle very large datasets, something RBM's excels at. The first implementation was done using the dataset from the Netflix Prize [37].

Another popular type of RS is the graph-based approach where each item is represented as a node in a graph. The connections between nodes are determined and given weights (probabilities). There are several methods of how to determine the weights on these connections, e.g. based on how often items are purchased together or based on deep learning embeddings. From this graph, there are several methods to compute recommendations, e.g. using Random Walk over the graph, and computing similarities between the walk of two different items. These RSs have the benefit of being computationally light in real-time and are able to produce recommendations with very low latency, without the need for data about the specific user [21]. A notable implementation of a graph-based RS is *Pixie* by Pinterest from 2018, which can produce image recommendations in real-time from a database containing billions of images [7].

RSs using recurrent neural networks such as Long Short Term Memory or Gated Recurrent Units have a slightly different approach based on sequences. Given a sequence of items, e.g. items added to cart on an e-commerce website, they predict the most likely following item in the sequence. This approach does not require user profiling and can produce recommendations based on data from the current session. These RSs have shown to be very effective in previous research such as [18], where the authors use a Gated Recurrent Unit (GRU) on session data from two different sets of data, one of them being click-stream data from an e-commerce site, similar to the datasets used in this thesis. GRU is a similar technique to LSTM, both are recurrent neural networks. The LSTM has a slightly more complex architecture, explained in section 2.3.

The recurrent neural network architecture behind the LSTM was introduced in 1997 in [19] and has since been able to achieve astonishing results for time series prediction within a range of fields apart from RSs such as text generation [23], music generation [22], language translation [5], etc.

There are a plethora of approaches to RSs within the field of deep learning and recently it has been discussed whether these novel approaches are better than simpler algorithms. In 2019 the progress made by these novel approaches was questioned in [6], as many of them were not possible to recreate and even fewer managed to outperform simple baseline algorithms. The authors stressed the importance of the work being reproducible and good baseline models for comparison.

Lastly, we can conclude that there are many options when choosing an algorithm for an

RS. In this thesis, we are limited by the lack of extensive user profiling. Without being able to connect different sessions by the same user, finding relevant similar users based on a single session is not feasible. This makes all classical collaborative filtering methods unsuited for this specific task. Without extensive user profiling, the approaches using recurrent neural networks or graphs are most suited. In this thesis we work with a recurrent neural network, however, a graph-based approach would also be a possibility for this type of RS.

1.3.2 Evaluating a Recommender System

This work relies heavily on offline metrics for evaluating RSs. The choice of what metric to use is therefore of great importance for the results. How to evaluate an RS using prerecorded offline data has been discussed heavily within the field. It has been suggested that simple accuracy metrics cannot fully describe the success of an RS. In [30] the authors suggest that the accuracy focus within the field has hurt development. In [3] the authors suggest Novelty and Diversity as other important metrics for offline evaluation. In [10] the authors suggest two other metrics, Catalog Coverage and Serendipity while concluding that there is no perfect metric and that every RS designer has to choose what metrics to optimize on depending on how they would like the RS to perform. In [43] the authors conclude that there is a trade-off between diversity metrics and accuracy metrics. If an RS were to be optimized on diversity without considering accuracy at all, selecting items by random would be sufficient. While an RS optimized on accuracy would trend towards only recommending the most sold items as they are more likely to be what the user actually bought. The authors conclude that the trade-off between these two important aspects of an RS and that the designer of the RS has to decide on what trade-offs to make, depending on the specific use case and implementation.

These studies all lead us to the conclusion that RSs cannot be judged by one metric alone. In this work we will, therefore, use a combination of a few metrics, capturing both accuracy and diversity, to judge the performance of an RS.

1.3.3 Learning curves in deep learning

In this work, the performance of a deep learning RS as a function of the amount of available data is investigated. The performance of a generic learning system, such as a neural network, as a function of the available amount of training data, has been referred to as its learning curve. In 1997 it was proposed in [1] that the entropic loss of the system was proportional as $\varepsilon(m) \sim d/m$ where d is the number of parameters and m the amount of available data. In 2017, the learning curves of deep learning networks were further studied in [17] where the authors implemented four different deep learning networks, where one of them was an LSTM for language modeling, similar to what we have used for product recommendations in this thesis, and trained them on datasets of different sizes. They concluded that the performance of deep learning networks scaled predictably and proved a power-law ($\varepsilon(m) \sim am^{\beta}$) generalization error scaling as a function of the dataset size.

These studies suggest two different, although very similar, relationships between performance and the available training data. The more recent of them have performed empirical research using more recent techniques within deep learning and having access to powerful computing resources which were inaccessible in 1997. We, therefore, consider it more likely that the findings in our thesis will conform to those in [17]. However, since theses studies

only consider generalization error and loss it is not fully applicable on RSs as the definition of performance must be wider, considering other metrics as well.

1.3.4 Embeddings

In this thesis embeddings (vector representations) are used for finding similarities for abstract things such as buying behavior on a specific market. There are several approaches to generate such embeddings, and depending on the selected approach results may vary vastly. Using a neural network to create embeddings of an item is a popular technique within the field of Natural Language Processing, with models like Word2Vec [32] and GLOVE [34] providing vector representations of words capturing their semantic meaning. This technique can be applied to many more categories such as images, songs, videos, and products. Meta-Prod2Vec is a method to generate product embeddings for RSs introduced in [40], using a similar strategy as in Word2Vec with the addition of a product attribute such as category. In this thesis, we use embeddings to compare markets, which are represented as sets of purchase sequences, much like a document that contains sets of word sequences (sentences). Doc2Vec is a method for representing documents as embeddings introduced in [26], used to classify text in document categories such as *positive* or *negative*.

The results from Word2Vec, Meta-Prod2Vec, and Doc2vec acknowledge that embeddings from deep learning networks work well to capture information about an item based on the context. For the applications in this thesis, both Meta-Prod2Vec and Doc2Vec could potentially be sufficient solutions. However, these require implementations of new deep learning models unrelated to the RS. Instead, we use the LSTM network used for the RS to generate embeddings.

1.3.5 Data ethics

Data ethics is the motivation behind this work, in the sense that we investigate how giving customers more control of their data might affect RS performance, as less data inevitably will be collected. Ethics is a topic widely discussed within the field of data science today as privacy and integrity might be compromised without the average person being aware of how his or her data is being used. In [44] the author suggests that Big Data might compromise traditional assumptions of individuality, free will, and privacy. It is stressed that the population needs to be educated in the consequences that the digital footprint left behind by each and everyone can have. This discussion has been actualized in media recently with projects such as *The Privacy Project* by *The New York Times* [39]. The authors describe how positioning data is collected without the user's knowledge and then sold to third parties without restrictions on how its used. This highlights how data even though it has been anonymized often can be used for identifying an individual. IKEA has the ambition to educate and make the average consumer aware of what data is being collected and for what purpose, giving back control to the users. Personalization based on user data should be a choice made by each and every user, as stated in [15].

These works highlight how topical this subject is. The rise of commercialized Big Data and Artificial Intelligence has been rapid, leaving the understanding of the average consumer behind. In this work, we touch on this by questioning the necessity of hoarding data without the consent of a well informed, aware consumer.

Chapter 2

Background

In this chapter, we present a brief description of some key concepts in this thesis. It is intended to be an introduction to the field as well as a background necessary to understand the thesis.

2.1 Recommender Systems

The main purpose of an RS is to produce meaningful content recommendations to users. The RSs help users discover new and relevant items and thereby enhance the user experience while increasing revenue for the company. Examples of notable RSs are the algorithm suggesting movies to users on Netflix, the algorithms that build personalized playlists on Spotify, and algorithm generating the *"Customers who bought this item also bought"*-recommendations on Amazon. These kinds of systems are of huge importance on the web today. 35% of purchases on Amazon and 75% movies watched on Netflix are results of suggestions by RSs [28]. Users are often presented with an overwhelming amount of choices and it can be very difficult to navigate this overflow of options for most. Therefore RSs are of great importance in e-commerce.

There are several different types of algorithms for RSs and which one to use depends on what to be recommended, to whom, and what data that is available. Most RSs aspire to develop notions of affinity between users and items and recommend items that match the user particularly well. There are three main approaches, the content-based recommendations, the collaborative filtering methods, and the hybrid systems which are a combination of the two [31]. In this thesis, we use a deep recurrent neural network approach to the collaborative filtering technique. Below, we present a brief overview of collaborative filtering techniques.

2.1.1 Collaborative-filtering methods

Collaborative filtering methods work by collecting data about user interactions with items, such as ratings, views, purchases, etc. and then explore similarities in usage patterns. In user-user collaborative filtering similarities between users are analyzed and users with similar preferences are identified. These similar users are often referred to as neighbors. Based on the preferences of the neighbors to a particular user, recommendations can be produced by analyzing which items that are highly regarded by the neighbors are then likely to be of interest to the user in question.

There's also item-item collaborative filtering that analyses similarities between items. Similarities between items are determined by usage data and items with similar usage patterns are considered similar. This technique does not require as much information about a user as the user-user collaborative filtering as its not necessary to determine similar users, instead its only necessary to identify a small set of items that the user has enjoyed and then recommend similar items. In online applications, the item-item approach often works better as there are fewer items available than users in the system, in practice, this leads to a faster system and often more accurate recommendations. There are several techniques and algorithms that implement collaborative filtering type of RS, utilizing techniques such as, matrix factorization, restricted Boltzmann machines, recurrent neural networks, etc [31].

2.1.2 Sparsity and the Long tail

One of the greatest challenges facing the development of RS is data sparsity, and the RS in this thesis is no exception. This is due to most users not interacting with most items, and the vast majority of the interactions are only with a few items, the most popular ones. This causes problems when trying to find similar users and items based on interaction data, as with different collaborative filtering techniques. The probability of finding users or items with similar interactions decreases when less data is available [31].

Figure 2.1 describes the sales distribution on *IKEA.de*. It is clear that very few items sell in great quantities while most sell less regularly. Understanding the dynamics of the sales on an e-commerce platform is necessary to understand the workings of the diversity metrics used in this thesis, catalog coverage, and novelty (described in section 2.7) and the choice of RS algorithm made by the developer.

In this particular case, **50%** of the total number of items sold is just composed of **5.4%** of the products. This condition with a small number of very popular items and the remaining items (the vast majority) being less popular is called *the Long Tail*. A consequence of the sales being unevenly distributed over the product space is that there is an excess of data about the most sold products, but a significant shortage of data about the less popular products. Good RSs helps users discover products that may be of interest to them from the entire product space, not only among the most popular ones. It has been suggested that RSs that premier more novel, or less known products to a higher extent may increase total sales [4]. This poses a great challenge to most RS as they are more likely to recommend products that more frequently occur in the data, as the probability of finding similar items based on the data increases when there is more data available.

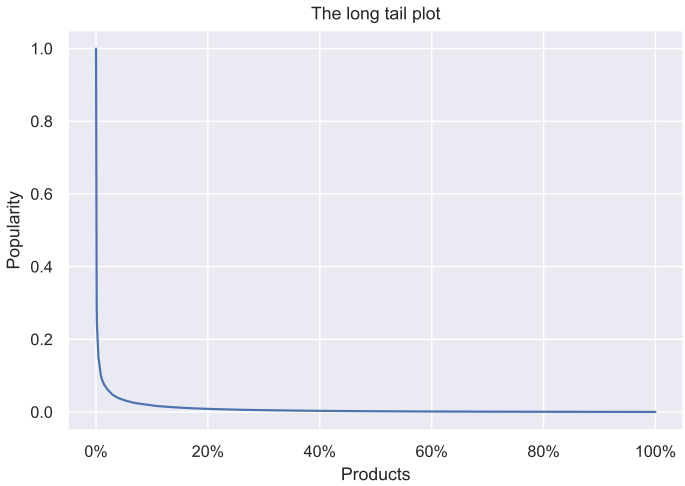


Figure 2.1: The long tail plot showing the popularity distribution among products. The y-axis displays the product popularity scaled between 1 and 0, and the x-axis displays the percentage of the products that reach given popularity or higher. Sales data from IKEA.de has been used to produce this figure.

2.1.3 The Cold-start problem

Introducing new items and users to the system causes a problem to most RSs. This is due to the lack of data concerning these new items or users. If there is no data about the users, it is not possible to find suitable recommendations and if there is no data about a specific item it will never occur in the recommendations. This is often referred to as the cold-start problem. If we consider a user-user collaborative filtering system where the idea is to find similar users (neighbors) based on usage patterns, but if there are no earlier interactions by a user it is impossible to determine which other users that are similar.

Similar problems arise in item-item based collaborative filtering systems when there is no usage data about a new item, which then will lead to that this item is not considered similar to any other item and therefore never recommended. If an item is never recommended, the time required to collect enough data for it to be included in the recommendation will be significantly longer.

Content-based RSs do not suffer from the cold start problem in the same way as collaborative-filtering approaches, as all the data needed are attributes of that specific item or user which most often are added when the item or user is registered.

2.2 Neural networks

In this thesis, we use a deep recurrent neural network, LSTMs which are described in section 2.3, as an RS. To be able to understand these networks, some general knowledge about neural networks is necessary.

Neural networks are a type of machine learning, in its earliest models designed to mimic a biological neural network, such as the human brain. The network is built up from a series

of non-linear elements (neurons). Each neuron takes an input, applies a weight, W, and adds a bias, b, and then applies a non-linear activation function, f. See equation 2.1. Each neuron takes several inputs and has a specific weight and bias for each input. There are a couple of different non-linear activation functions commonly used such as *tanh* (Hyperbolic tangent) and *sigmoid* among others, see equation 2.2 and figure 2.2. Each activation function takes a single number and performs a fixed operation on it. Which one to use depends on the specific use case. The *sigmoid* function is often used for models that predict probabilities since it ranges between 1 and 0, while the *tanh* which ranges from -1 to 1 often is used for classification problems.

$$Y = f(X \cdot W + b) \tag{2.1}$$

$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}} \qquad f(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.2}$$

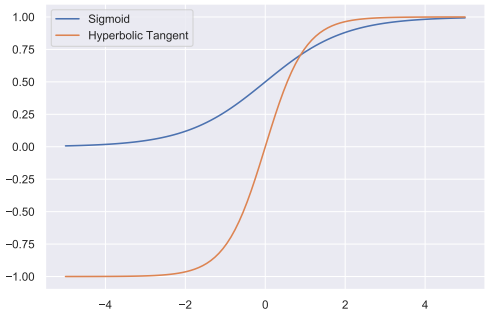


Figure 2.2: The Sigmoid and Hyperbolic tangent activation functions commonly used in neural networks.

Neural networks are often structured as distinct layers of neurons stacked together, most commonly fully connected layers which means that neurons are fully pairwise connected between two adjacent layers. Therefore, outputs of neurons in one layer become inputs for the neurons in the following layer. A simple neural network is described in figure 2.3.

The weights and biases are the learned parameters of the network and the learning happens when the network is trained on labeled training data using backpropagation. Backpropagation is an algorithm using stochastic gradient descent to calculate the gradient of a given loss function (error) with respect to the weights. This is known as supervised learning, where an input with a correct output is provided to the network which updates its parameters. Deep neural networks can capture exceptionally complex patterns [24].

2.2.1 Loss function - Categorical Cross Entropy

The loss function is what the neural network is minimizing when training the weights using stochastic gradient descent. In previous work, the loss of a neural network as a function of the size of the available training dataset (the Learning curve) is described [17]. In this thesis, the

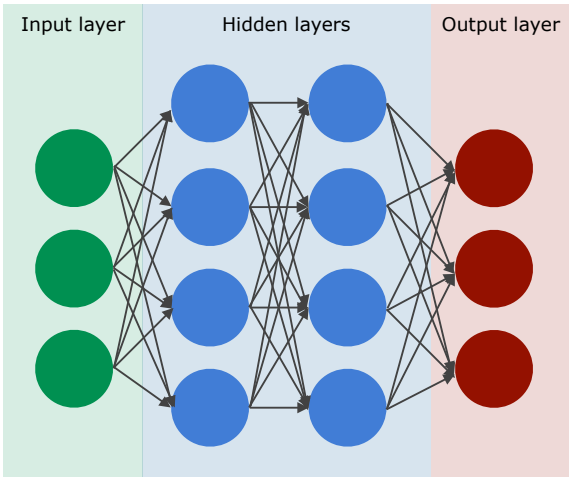


Figure 2.3: A three layer (input layer does not count in accordance with the convention) neural network with two hidden layers with 4 neurons each and one output layer of 4 neurons.

loss as a function of training dataset size when validated on a separate validation set is used to compare the results with previous work. In the implementation used in this thesis a loss function known as Categorical Cross Entropy is used [13]. It is also known as Softmax Loss or simply Cross-Entropy loss. It is most commonly used for classification tasks where more than one option among the possible classes could be correct. Categorical Cross Entropy is defined in equation 2.3 where $\boldsymbol{C}$ is the set of all possible classes in the classification problem. $\mathbf{t}$ is the target vector where all elements, t_i are zero except for the ground truth $t_p = 1$, s_i is the score for that given class from the neural network. $f(s_i)$ is the softmax activation function which is a common activation function for the final layer in a classifying neural network as it outputs a probability for each class between 1 and 0, and the sum of the probabilities for all classes is 1.

$$CE = -\sum_{i}^{C} t_i \log\left(f(s)_i\right) \qquad f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{2.3}$$

2.3 Recurrent neural networks and LSTMs

Recurrent neural networks are a type of neural network which utilizes previous outputs from the neural network as an input. This is particularly useful when predicting time series and other sequences. In this thesis, a recurrent neural network is used as an RS. Understanding the difference between an ordinary neural network and a recurrent neural network is vital to understand why they are a good fit for an RS based on purchase sequences.

Traditional neural networks do not consider earlier results which makes them less suited for modeling time series, where the current state might depend on the previous state. The recurrent neural network utilizes the previous states for predicting the following. Figure 2.4 gives a graphical representation of a generic recurrent neural network. Each cell, A, is a part of the neural network which considers some input x_t and outputs a value h_t and passes the

output to the next cell in the network, which takes the following input and so on.

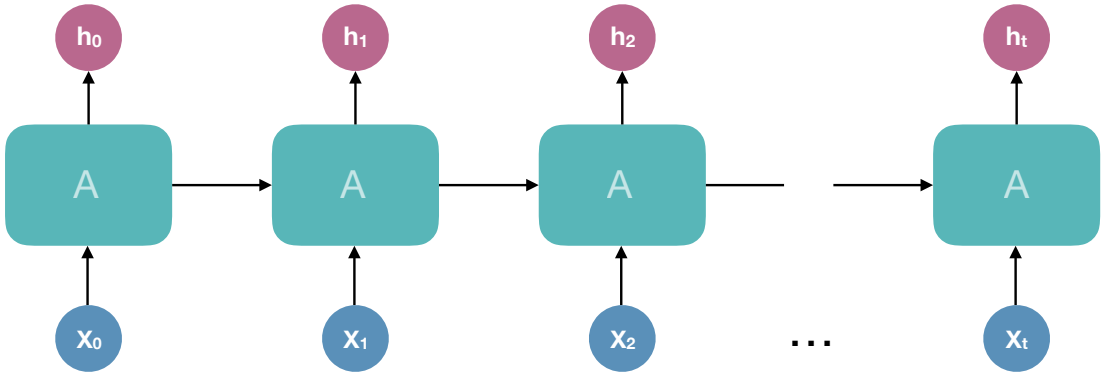


Figure 2.4: The architecture of a generic neural network, where the output of one cell is passed on to the next. This enables learning from sequences of data, where the one data point in some way depends on the previous ones. Image adapted from [33]

A powerful version of recurrent neural networks are Long Short Term Networks, referred to as LSTMs, which are capable of capturing long term dependencies. Long term dependencies are connections between data points that are adjacent to each other. An example from language modeling is the sentence *"When I traveled to Paris last year, I learned to speak French"*. From the context, it is obvious to a human that the language the person learned was French, because he or she went to Paris. For a neural network, this is really hard if it is not able to keep the geographical information from word 5 in the sentence until it needs to predict the language for word 12. LSTMs can capture these long term dependencies better than other types of recurrent neural networks which makes them remarkably powerful for a variety of applications, from financial predictions and language models to RSs. Figure 2.5 describes the dynamics of an LSTM-cell, denoted as $\boldsymbol{A}$ in figure 2.4. This is the architecture of the recurrent neural network in the RS used in this thesis, and to be able to understand why it is an efficient algorithm for this purpose, a general knowledge about the algorithm in each LSTM-cell is necessary. Each LSTM cell, see figure 2.5 and equation 2.4, consists of 4 neural network layers, connected in a slightly different way than the ordinary feed forward neural network presented in figure 2.3.

The reason LSTMs are able to model long term dependencies successfully is the cell state, denoted as $\boldsymbol{C}$ in figure 2.5 and equation 2.4. It runs through the entire LSTM chain with a slight modification in every cell and therefore it can transport and preserve information over long distances in sequences. Information can be added and removed from the cell state through gates, and the LSTM cell has three different gates. Gates are operations that select what information to let through to the cell state.

The first gate in the LSTM is the *forget gate*, labeled as f_t in equation 2.4. This is where some information from the incoming cell state can be removed, decided by the sigmoid layer in the forget gate. The layer considers h_{t-1} and x_t and the sigmoid function outputs a number between $\mathbf{1}$ and $\mathbf{0}$ for all values in the cell state, where a $\mathbf{1}$ completely preserves the information

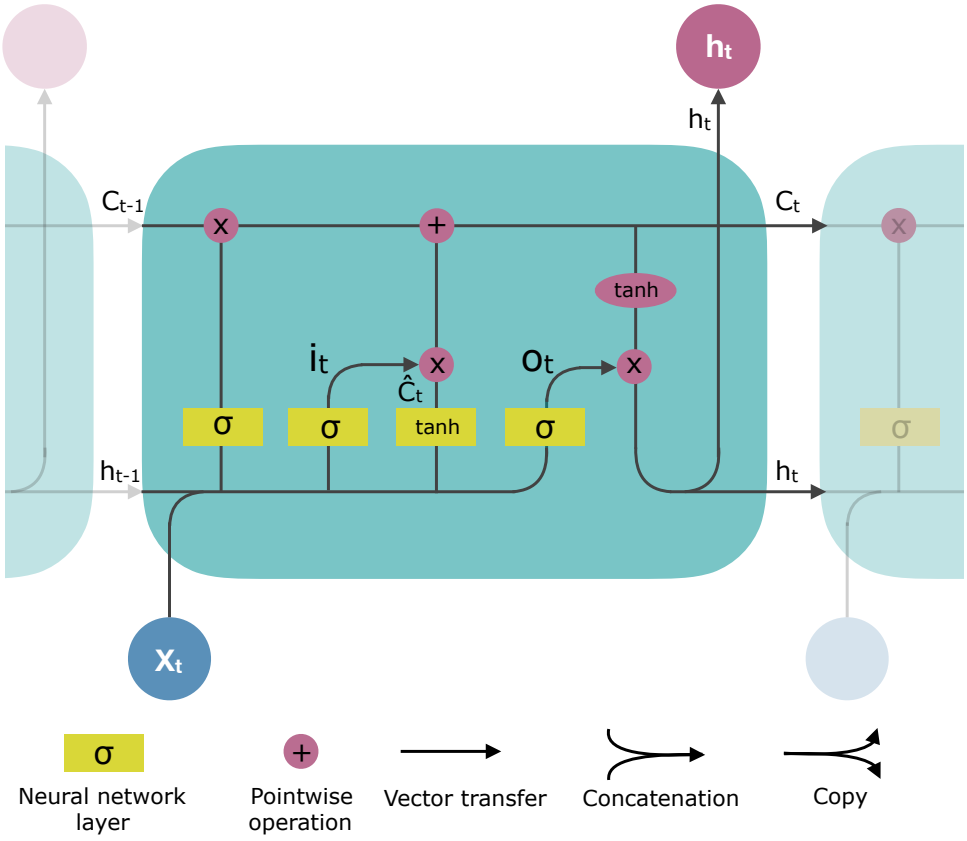


Figure 2.5: The architecture of an LSTM cell in a neural network. Image adapted from [33]

and a 0 deletes the information.

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \end{aligned} \tag{2.4}$$

The second gate is known as the *input gate* and it is the combination of i_t and $\tilde{C}_t$ in equation 2.4. i_t is often referred to as the input gates activation vector as it decides which values to update the cell state with and to what degree, from 1 to 0. The values to update the cell state with comes from the tanh layer, denoted as $\tilde{C}_t$. The final operation in the input gate is adding $i_t * \tilde{C}_t$ to the cell state.

$$\begin{aligned} o(t) &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{2.5}$$

The output of the LSTM-cell, h_t, is computed in the output gate, described in equation 2.5. The output is a filtered version of the cell state, o_t is the activation vector which decides what to keep from the cell state to the output. The cell state is put through a tanh operation to push the values between -1 and 1 and then multiplied with the activation vector, o_t.

The equations 2.4 and 2.5 describe the forward pass in one LSTM-cell. The chain of LSTM cells are often between 20 to 100 cells long depending on the application [33].

2.4 The LSTM as a Recommender System

The LSTM networks described above can be applied to RSs advantageously when there is a lack of user profiling, as it is possible to make recommendations based on only data from the current session without any specific knowledge about the user. It is an implementation of an item-item based RS which utilizes information about the items, instead of information about the specific user to whom the item is being recommended. In this case, the information about an item is which context it occurs in, i.e which items that frequently occur together. This means that this particular RS suffers from the cold start problem with new items, as there is no information about what context a new product is usually bought in. The network is

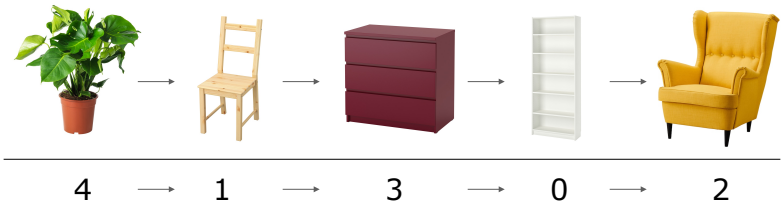


Figure 2.6: One possible sequence of products bought together. Each product has a unique product ID and the sequence of products is represented as a vector of these IDs.

trained on purchase data from an online store. Every purchase can be converted to a sequence with the order of the purchased items being based on in which order they were added to the cart, as in figure 2.6. Each product is given an ID number from in the range from 0 to $|\text{items in catalog}| - 1$. Indexing starts from 0 instead of 1 as it is intended to be used as indexes in a vector.

The sequences contain plenty of information. In the case of the sequence in 2.6 there is value in knowing that the chair follows the plant and after a plant and a chair the drawer is a likely next purchase. Therefore it is favorable to split the sequence into as many subsequences as possible, with the shortest sequence containing only 2 items, this is illustrated in figure 2.7. This way, one purchase with 5 items contains 4 subsequences worth using as data points for training the LSTM-network. To prepare the data for training the network, each purchase is divided into as many subsequences as possible and in each sequence the last item is selected as the label while the rest is given as the input to the LSTM. This input/target partition is highlighted in figure 2.7.

To produce product recommendations the trained LSTM is given a sequence, with minimum length 1 of products that a user has added to the cart. The LSTM then outputs a vector with length $|\text{items in catalog}| - 1$ with the estimated probability for each item being the next purchase at that index in the vector, illustrated in figure 2.8 As it is often desired to recommend more than one item at the time, to increase the likelihood of giving the user a useful recommendation. This is done by selecting the k most probable items, where k is the number of items to be presented to the user.

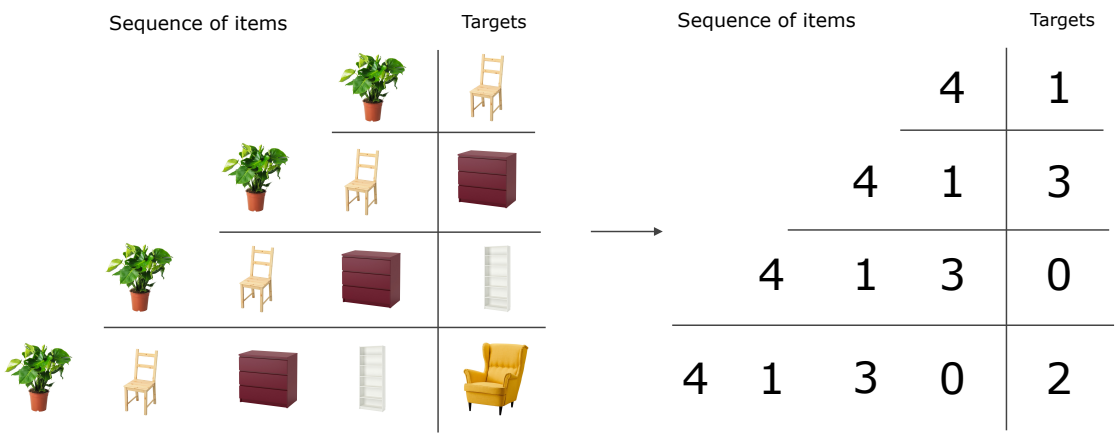


Figure 2.7: The different subsequences that can be gathered from purchase in figure 2.6 and then used to train an LSTM RS.

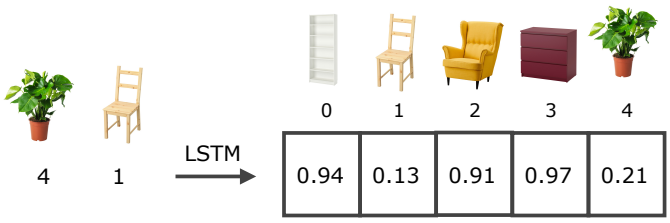


Figure 2.8: Given the input of sequence the sequence *plant* then *chair* the LSTM would output a probability vector with the probabilities for each product in the catalog being the next purchase.

2.4.1 The existing IKEA implementation

The RS described in the previous section has been implemented by the Recommender System team at IKEA using Keras and Tensorflow. The implementation as presented by Keras is described in figure 2.9.

The input is a sequence of 64 products. This poses a problem in the sense that not all purchases contain the same number of products (64 in this case). To solve this all sequences shorter than 64 items are padded with zeroes. This makes a purchase containing 20 products represented by 44 zeroes and then the 20 products.

The first layer is an embeddings layer. The details of embeddings are described in section 2.5, generating a sequence product embeddings with 20 dimensions per product which are then the input to the LSTM layer. Using embeddings as inputs to the LSTM instead of only the product ID has the benefit that information about relations between products is included when using embeddings the model is made aware that two different chairs are similar. If only product IDs are considered, there is no information that makes a chair more similar to another than something completely unrelated like a fork. Using embeddings like this was shown to substantially improve recommender performance in [38].

The LSTM layer has 50 LSTM-cells and their structure is described in detail in section 2.3.

The output layer has the length of the number of products included in the dataset. Each

neuron in the layer outputs the probability of this particular product being the following one given the input sequence.

```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 64, 20)            197840
_________________________________________________________________
lstm_1 (LSTM)                (None, 50)                10650
_________________________________________________________________
dense_1 (Dense)              (None, 9892)              504492
=================================================================
Total params: 712,982
Trainable params: 712,982
Non-trainable params: 0
```

Figure 2.9: The model summary printed by the `summary()` method in Keras.

2.5 Embeddings

Embeddings are a technique to map an item to a vector with numerical values. The item could be a word, a product, a sequence, or anything else that can be an input to a neural network. In this thesis, we use embeddings to describe the purchase behavior of customers, to compare the purchase behavior of customers in different markets.

The numerical values in the vector are meant to capture information about the item based on its context, and if successful enables comparing similarities between items. Examples of this are the words *Hike* and *Trek* which are words very semantically similar, but very different in spelling. If an embeddings model has been able to capture semantically similarity it should consider these words very similar. Embeddings are produced by neural networks, often trained to perform another, related task. The network is stripped of one or more layers, and the embedding vector is the output of one of the hidden layers in the network. The dimensions of the embedding are therefore equal to the number of neurons in that particular layer.

2.5.1 Vector representations of anything

An example of how things can be represented as numerical vectors could be persons and their attributes. An attribute could be humor, generosity, temperament, etc. In figure 2.10 four persons are described as vectors of five attributes. Each attribute has a score between 1 and 0. This range can also be between -1 and 1 or any other range depending on the activation function in the neural network. Five dimensions are not enough to capture a full personality or the full semantical representation of a word, but if increased to between 100 and 1000 dimensions, a lot of information can be stored.

Each position in a vector represents how this particular person ranks on this specific attribute. When these vectors are generated by a neural network, the values in cells do not

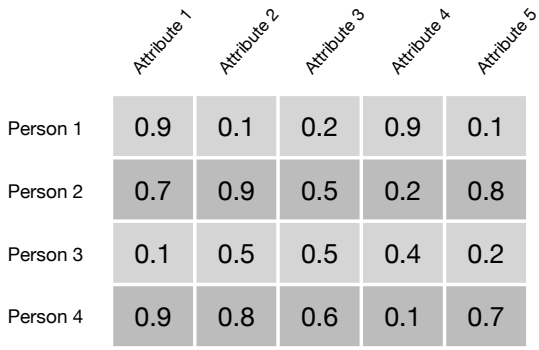


Figure 2.10: Five persons described as vectors containing their attributes as a score in the range from 0 to 1

correspond to a specific attribute that can be interpreted by a human. Instead, it captures an attribute important the network found important to consider when minimizing the loss function.

2.5.2 Calculating similarities between vectors

Similarities between these vector representation of persons can be calculated using cosine similarity. [14]

$$\text{cosine similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \cdot ||\mathbf{B}||} \tag{2.6}$$

If this is applied to the persons in figure 2.10, and their made up personality vectors we obtain the similarity metric in figure 2.11. Among these four persons, 2 and 4 are the most alike when only considering these 5 attributes of a person.

This method of computing similarities between the vector representation of persons can be applied to any type of embedding describing anything.

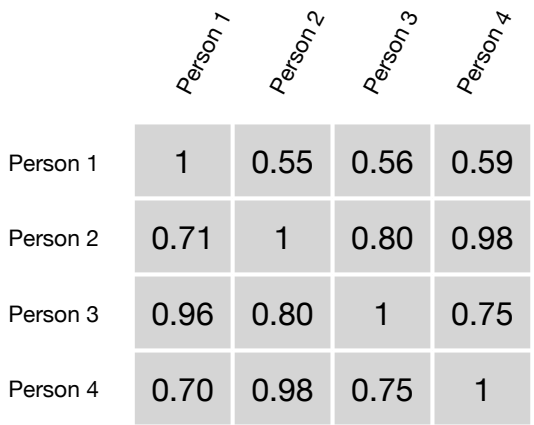


Figure 2.11: A similarity matrix comparing 4 different persons using cosine similarity on the numerical vectors (embeddings) describing them in figure 2.10.

2.5.3 Word embeddings

The most common application of this technique is word embeddings, where words are given a vector representation to capture their semantic meaning, based on what context they occur in. A commonly used model to do this is the Word2Vec algorithm. In the original Word2Vec paper [32], two different algorithms were proposed. For simplicity, only one of them, known as *Continuous bag of words*, is described here. Word2Vec is used as an example to give a general understanding of the concept of embeddings. Although Word2Vec is not used in this thesis, the method of generating the embeddings is identical to what has been used here and therefore suitable as an example of how to generate embeddings.

A neural network was trained to predict the missing word given the previous two and the following two words, as described in table 2.1. The neural network learns what words often occurs in what contexts and is thereby able to capture its meaning. An example of this is the second sentence in table 2.1 where *ten* just as well could have been *five* and these words should occur in similar contexts, and therefore *ten* and *five* are semantically similar.

Table 2.1: The data structure for training the continuous bag of words model in word2vec.

Sentence	Input	Label
What does the fox say	what, does, fox, say	the
I found ten dollars today	i, found, dollars, today	ten
Some people like spicy food	some, people, spicy, food	like

The network is trained to perform a *fake* task, and when the training is done, the output layer of the network is removed. This enables access to the output of one of the hidden layers, and this is the numeric vector containing the semantic information about the word.

To demonstrate the function of this alogirthm one can take the vectors representing *France, Italy* and *Paris* and compute *Paris - France + Italy*. The most similar word vector in the dictionary to the resulting vector is *Rome*, proving that the embeddings have captured the relationship between the words and their meaning [32].

2.6 Learning Curves in Deep Learning

Learning curves describe at what rate a learning machine, such as a deep learning network as the LSTM, improves as the amount of training data increases. In this thesis, the performance of a deep learning RS depending on what and how much training data it is provided with is evaluated.

It is widely believed that increasing the amount of training data will improve the accuracy of the deep learning networks. However, information gained by each added training example will decrease as a function of the total amount of available training data.

In 1992 it was proposed that learning curves of learning machines universally would follow the asymptotic function in equation 2.7 where $e(t)$ is the average entropic error or loss, t is the number of available data points for training and d is the number of parameters in the

model [1].

$$e(t) \sim \frac{d}{t} \tag{2.7}$$

More recently it has been suggested that the error scaling would take more of a power-law relationship between the error and the number of data points available for training as in equation 2.8 where γ is the error when the model has exhausted its capacity, α is a constant and the exponent β is an exponent in the range $0 > \beta > -0.5$ [17].

$$e(t) \sim \alpha t^{-\beta} + \gamma \tag{2.8}$$

However, also according to [17] there are two regions of dataset sizes where the error does not decrease when adding more samples. At first, when the dataset is very small, the model will struggle to learn and only perform as well as random guessing. There is also a theoretical limit to the lower bound of the error which the model is unable to improve upon. Due to noise and other factors that cause imperfect generalizations, there will be an irreducible error, and as the model approaches this limit the information gain per added sample will asymptotically approach 0 [17]. These dynamics are sketched in figure. 2.6

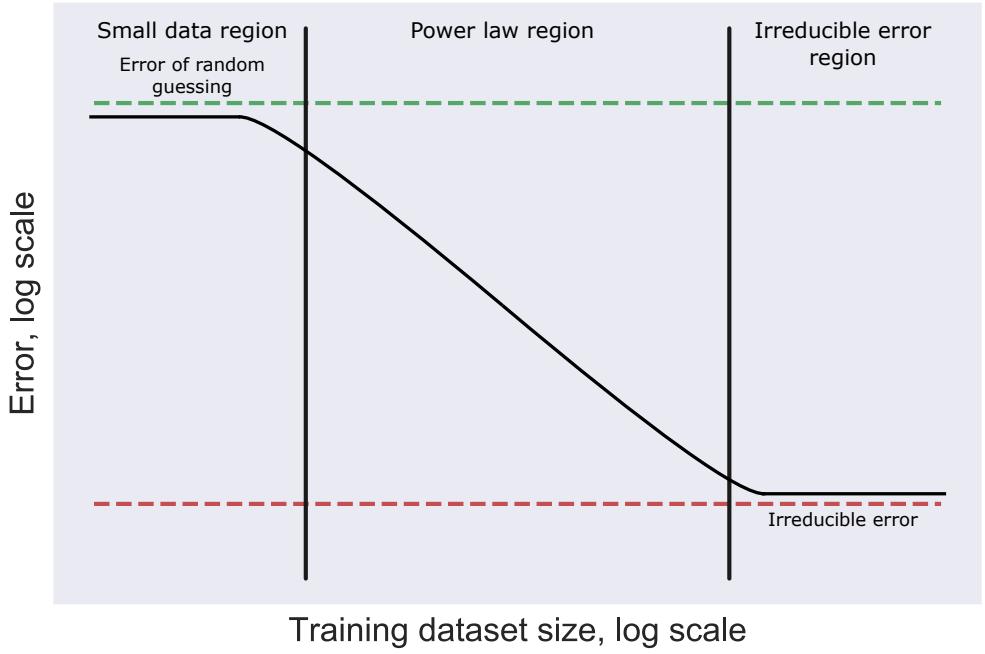


Figure 2.12: A sketch of law learning curves for deep learning networks. Image adapted from [17]

2.7 Metrics for evaluating the performance of a Recommender System

The best way to evaluate an RS is through online A/B testing on real users [16]. Due to time constraints, it was not a possibility for this thesis. Instead offline evaluation using previously

recorded user-data was used for evaluation in this thesis.

Evaluating RSs on recorded data offline is complex and non-obvious, and how its done is dependent on what is considered good performance for a specific application [10].

Traditionally RS has been evaluated on accuracy and the main target has been improving the accuracy. However, accurate recommendations are important but insufficient to describe the performance of an RS. One example of when accuracy is an insufficient metric could be if a user has watched *The Lord of the Rings: The Fellowship of the Ring*, recommending the sequel *The Lord of the Rings: The Two Towers* will most likely be very accurate but since the user with high likelihood already knew that there was a sequel and planned on watching it anyway, the recommendation did not provide much value for the user. Several metrics are related to accuracy but have slightly different definitions, such as Mean Average Precision and Precision@k.[30, 16]

It is therefore important to consider other aspects of performance as well to judge an RS. Metrics that do not premier the most accurate recommendation but instead try to describe value provided to the user in the sense that the recommendations are more diverse are among others catalog coverage and novelty. These are described in detail in section 2.7.3 and 2.7.2

To be able to measure the performance of an RS, performance has to be defined. Concerning which metrics to use there has been a lot of discussion within the field [10, 43, 3, 30] and the answer is problem specific and therefore chosen with that specific problem in mind. In this thesis, we consider three metrics, Precision@k, Novelty, and Catalog Coverage as defined in the following sections. These have been chosen to be able to judge both based on how close the RS predicts the truth (Preciscion@k) while also delivering useful, diverse, and valuable recommendations (Catalog Coverage and Novelty) to the users, that helps them navigate among more than 20000 different products.

We also consider the loss function, Categorical Cross Entropy described in section 2.2.1, to be able to compare with previous work on learning curves as in [17, 1].

2.7.1 Precision@k

Most RS delivers more recommended items than just the one considered most likely. It is common for an RS to recommend $2-6$ items. A recommendation is considered successful if any of these likely items are correct compared to which item the user actually watched or bought. We use a slightly modified version of the definition found in *Practical Recommender Systems* by Kim Falk [8], as described by equation 2.9 where $\text{rec}_k(i)$ is defined by equation 2.10 and N is the number of recommendations made when validated on a validation set. The definition in *Practical Recommender Systems* varies in the definition of $\text{rec}_k(i)$ where the definition in equation 2.10 only considers if one of the top k recommendations are relevant. The definition in the book considers if more than one of the top k recommendations are relevant recommendations. However, the available dataset in this thesis only allows comparing the predictions with one true item, as the dataset only contains which item the customer actually bought next. It is therefore not possible, from the available data, to determine if more than one of the top k recommendations were relevant.

$$\text{precision@k} = \frac{\sum_{i \in N} \text{rec}_k(i)}{N} \tag{2.9}$$

$$\text{rec}_k(i) = \begin{cases} 1 & \text{if any of the top k recommendations are correct} \\ 0 & \text{if none of the top k recommendations are correct} \end{cases} \tag{2.10}$$

2.7.2 Catalog coverage

The Catalog Coverage is the percentage of the available items in the catalog which are actively being recommended by the RS. A recommender with a high coverage gives the user a more detailed and nuanced picture of the product space and is, therefore, an indicator of quality [10]. However, if one only considers Catalog Coverage when evaluating an RS it would be sufficient to sample random products from the catalog as recommendations to reach 100%, therefore Catalog Coverage alone is not very useful as a metric. It must be considered together with other metrics.

In this thesis we use the definition of Catalog Coverage found in equation 2.11 [10]. $\boldsymbol{I}$ is the total set of items in the product space and $\boldsymbol{I}_L^j$ is the list of the top k most likely items given by the RS for any given recommendation. $\boldsymbol{N}$ is the number of recommendations made when the RS is validated on a validation set.

$$\text{Catalog Coverage } = \frac{\left|\cup_{j=1,N} \boldsymbol{I}_L^j\right|}{|\boldsymbol{I}|} \tag{2.11}$$

2.7.3 Novelty

Novelty is a metric that tries to capture how different a recommendation is compared to what the user has previously seen. Since the data analyzed in this thesis does not contain user identifiers it is impossible to judge if an item has been viewed by the user before. Instead, the popularity of the item is used as a measurement for novelty, less popular items are therefore premiered in this metric.

We use the definition of novelty proposed in [3] which uses item popularity to determine if a recommendation is novel or not. We then measure the average novelty of all recommendations made in a validation set to be able to compare different models, see equation 2.12. $\boldsymbol{p(i)}$ is the relative popularity of i in the full dataset, which is measured as the sales frequency of all items normalized between 1 and 0, i.e the most popular item has $\boldsymbol{p(i) = 1}$. $\boldsymbol{N}$ is the list of all items that has been recommended by the RS on the validation set.

$$\text{novelty(N)} = -\frac{\sum_{i \in N} \log p(i)}{|N|} \tag{2.12}$$

2.8 The importance of recent data when developing Recommender Systems

Seasonal preferences and an ever-evolving product catalog poses great challenges to RSs, as older data will become less relevant with time. Products that sell very well during the months before Christmas in countries where Christmas is celebrated might not be all that popular

in the following months and colors that sell well in spring and summer might not be equally popular in autumn.

When a new collection of products is introduced these all experience the Cold start problem described in section 2.1.3. If these products correspond to a high percentage of the sales, the model trained on data from a period before they were introduced will produce less accurate recommendations [25].

In [25] and in [42] the authors compare the performance of different matrix factorization RS on a dataset with movie ratings. When incorporating time as a factor and putting more weight on recent data than older they saw a significant increase in prediction accuracy, confirming the importance of having recent data available to make a recommendation from. Similar performance gains are achieved in [9] where the authors suggest a version of classical collaborative filtering where more weight was put on never data points.

Chapter 3
Method

There are plenty of hyper-parameters available to optimize in the LSTM, such as how many LSTM units used in the network, the number of epochs used for training, and so on. These hyper-parameters have not been optimized in this thesis. It is likely that the peak potential of the RS is higher than what the results in this thesis show if hyper-parameters would be optimized correctly, however, due to time constraints this has not been done.

To be able to judge RS performance, we defined the three relevant metrics to consider in section 2.7. In the following experiments, these metrics have been used.

3.1 Tools and workflow

The LSTM RS has been implemented in Python using Keras running on top of Tensorflow. Keras is a high-level package for the development of neural networks and deep learning developed by Google to enable fast experimentation. Tensorflow is an open-source platform for machine learning, developed and published by Google.

Google Cloud Platform (GCP) has been utilized for all data storage, data processing, and computation. GCP enables cloud computing and cloud data storage, eliminating the need for local high power clusters and data centers.

The workflow of all conducted experiments has been the following:

- Extract relevant data from Big Query-database on GCP using SQL.
- Preprocess data to sequences suitable for LSTM training using the data preparation tool Dataflow on GCP.
- Save training and validation datasets to files in Google Cloud Storage on GCP.
- Train the LSTM model on the relevant data using AI Platform on GCP and save the trained model to Google Cloud Storage.

- Validate model performance using the validation dataset and AI Platform for computing.
- Visualize results using python packages for plotting such as matplotlib and seaborn.

As the three different parts of the thesis all aim to understand how the data selected for the training of the model affects its performance, the variation has been in the selection of the data. Choosing different amounts of data, from varying markets, from varying periods and using varying validation datasets.

3.2 Performance as a function of dataset size

To test **RQ1** and **RQ2**, the LSTM RS was trained on different sized datasets from a few different markets, generating different RS models with identical settings.

The purchase data available from online sales at IKEA is structured market-wise and therefore one LSTM RS was trained per market, with training data and validation data from that specific market.

To obtain a better picture of the dynamics of this RS, the experiment was conducted on data from six markets: *Sweden*, *Canada*, *Poland*, *Spain*, *France* and *Germany*. The markets have different amounts of data available and there is also a slight difference in the product catalog for each market.

To split the available data from one market into a validation set and 10 training sets, first, all available data for that specific market was extracted and divided randomly into 11 sets. One of the sets was selected to be the validation set for that market. Then the 10 remaining sets were combined to form the 10 training sets of different sizes. This means that the 10% set is a subset of the 20% set and both the 10% and the 20% sets are subsets of the 30% set and so on. See figure 3.1

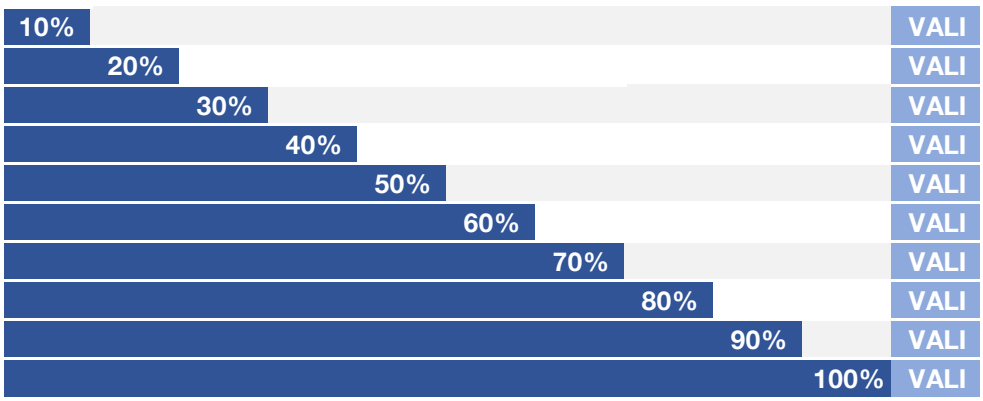


Figure 3.1: The structure of the datasets generated in each of the different countries. The full training set is the one named 100%, while all the others are different sized subsets. The same validation set was used for all models trained on data from one market.

The models that had been trained on the different datasets were then evaluated on the validation set, using the same validation set for all models within a market and the metrics

described in section 2.7. To be able to compare the results with the previous work on the subject, discussed in section 2.6, an exponential function describing the loss as a function of the number of data points in the dataset was fitted to the results.

3.3 Down-sampling on excess data

To test **RQ3**, if it is possible to select an optimal subset of the available data to achieve optimal performance, two different approaches were tested. First, the data were filtered based on when it was recorded and secondly on the number of purchases in a data point.

As described in section 2.8, it has been known in the industry that more recent data provides more value when training an RS than older data. By removing older data points from the datasets, the training sets decrease in size but might be more relevant as the data is more recent.

Another hypothesis on what data that contains the most valuable information for training the LSTM RS is that longer sequences of items capture purchase patterns better than the shorter sequences. Therefore it is reasonable to believe that removing the shortest sequences might increase performance.

Data was extracted from the French market from a period of 561 days. From these eleven datasets where created. A validation set containing data gathered during the most recent 51 days. Then datasets containing data from the 51 days previous to the validation set, the 102 days following the validation set, etc. This generated 10 datasets for training, with the largest containing data gathered during 510 days.

From each of these datasets, shorter sequences were then removed, with a varying minimum allowance being set to $2-7$ products in a sequence. This generated in total 70 datasets, with minimum sequence length from $2-7$ items in a purchase and period during which data was gathered varying from $51-510$ days.

The LSTM RS was trained on each of these 70 datasets and validated on the validation set containing the data from the 51 most recent days.

3.4 Up-sampling using data from other markets

To test **RQ4**, we established the hypothesis that it could be possible to use data from a secondary market to expand a small dataset from a market where there is a lack of data.

It is reasonable to believe that some markets are more similar than others regarding purchase patterns. Customers in different parts of the world have different preferences, seasons, and financial situations. To be able to judge what market to use data from, a similarity metric between markets must be defined.

There are several possible methods of creating similarity metrics between markets. In this thesis, one possible solution using sequence embeddings is tested.

In previous work, embeddings have been very successful in capturing complex behavior and patterns from large datasets. In this thesis, a method trying to describe purchase patterns from the different markets using embeddings is proposed and tested. Similarities between markets are then described using these embeddings.

To obtain these embeddings the LSTM RS was trained on a dataset containing data from all markets available, to obtain a model with no regional dependencies. This model will not be a sufficient recommender on any market since purchase patterns vary from market to market. This particular model is not purposed to produce recommendations, as it would be very inaccurate on all markets, but to produce embeddings comparing the markets.

We use the embeddings layer in the Keras sequential API [12]. This layer was placed before the LSTM layers in the network architecture as described in section 2.4.1.

When the model had been trained, all layers except for the initial embeddings layer was removed. The output of the model, when given a purchase sequence, was an embeddings vector (with length 1280) describing this particular purchase.

To describe a market, embeddings from 850 purchases from that market was created. Each market was then represented by the centroid [41] of these 850 purchase embeddings. The similarities between markets were computed by cosine similarity between the market representations in the embedding space.

When this proposed similarity metric between markets had been obtained it was evaluated and tested. The test has been constructed to imitate a situation where one small market (**A**) lacks enough data, where it could be useful to add some data from another, similar market to reach the amount of data necessary to train a model which reaches sufficient performance levels.

The test uses datasets for training the model from three different markets, denoted as **A**, **B** and **C**, where **A** and **B** are similar, while **A** and **C** are dissimilar.

We selected a validation dataset from **A** and training datasets from **A**, **B** and **C**. The training sets from **B** and **C** are selected to be equally sized and all purchases were randomly selected from the available data, to eliminate factors that affect performance such as the dataset size and how recently the data was collected. The set of products that exists in the three different markets are not completely equal, some products only exist in one market and some products have different ID:s in different markets. Unfortunately no mapping between product ID:s on different markets exists. To overcome this the training datasets from **B** and **C** were filtered before training to only contain purchase sequences with product ID:s that exists in **A**.

Three LSTM models were then trained on these training data sets and validated on the validation set from **A**. If the similarity metric functions as intended, the model trained on similar data from market **B** should perform better than the model trained on dissimilar data from market **C** when validated on the validation set from **A**. This test was executed for three different markets, **A**, with corresponding similar and dissimilar markets.

Chapter 4
Results

In this chapter the results obtained from the experiments presented in chapter 3 are presented.

4.1 Performance as a function of dataset size

Splitting the datasets into different subsets of the full dataset for the markets *Germany, France, Poland, Spain, Sweden, Canada* resulted in datasets of the sizes presented in the table 4.1. The number of purchases is normalized to avoid disclosing any sensitive business information. As described in figure 2.7, one purchase generates more than one data point. Due to this, the number of purchases is not the exact percentage of the full dataset as would be expected. The number of purchases in the 10% dataset is not exactly 10% of the 100% dataset. This is a consequence of the datasets being divided based on the number of data points.

Table 4.1: The number of purchases in different datasets from different markets.

	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Germany	0.098	0.198	0.298	0.398	0.499	0.598	0.700	0.800	0.900	1.000
France	0.040	0.081	0.123	0.165	0.207	0.248	0.289	0.331	0.372	0.413
Poland	0.021	0.044	0.067	0.089	0.112	0.135	0.157	0.180	0.202	0.225
Spain	0.020	0.041	0.063	0.084	0.106	0.127	0.148	0.169	0.190	0.211
Sweden	0.022	0.046	0.070	0.094	0.117	0.141	0.165	0.188	0.212	0.236
Canada	0.028	0.058	0.089	0.119	0.149	0.179	0.209	0.239	0.270	0.300

The loss (categorical cross entropy) for the models trained on the datasets in table 4.1 are presented in figure 4.1. As expected from [17],a power-law function with loss as a function of available training data ($e(t) \sim \alpha t^{-\beta} + \gamma$) could be fitted. These functions are presented in equation 4.1, where C_i are constants.

The learning curves show a power-law pattern on the form presented in equation 2.8 as was expected. This further confirms that there is an irreducible error which the model cannot overcome. It is, however, possible to further optimize the model for this particular use case using different hyperparameters, but due to the power-law behavior of the learning curve, a certain amount of error cannot be avoided.

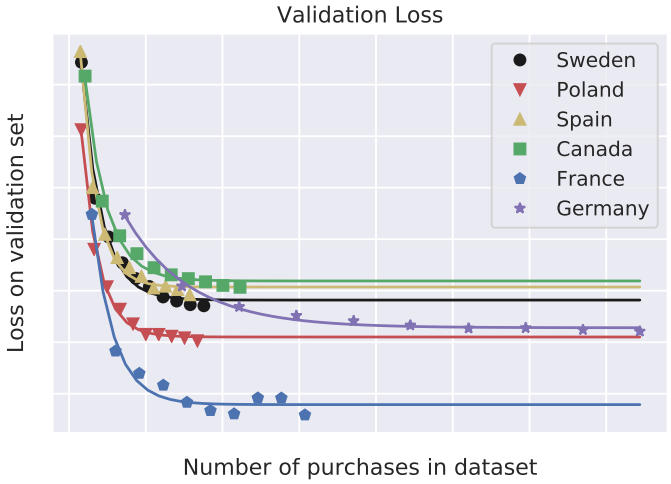


Figure 4.1: The loss (categorical cross entropy) of the final epoch in training the LSTM plotted as a function of the amount of available training data. Exponential functions were fitted and plotted to the results from each market.

$$\begin{aligned}
\text{Germany} &\quad 1.03 \cdot t^{-5.93\times10^{-6}} + C_1 \\
\text{Sweden} &\quad 1.96 \cdot t^{-1.94\times10^{-5}} + C_2 \\
\text{France} &\quad 2.09 \cdot t^{-1.76\times10^{-5}} + C_3 \\
\text{Spain} &\quad 1.83 \cdot t^{-2.39\times10^{-5}} + C_4 \\
\text{Poland} &\quad 1.62 \cdot t^{-2.23\times10^{-5}} + C_5 \\
\text{Canada} &\quad 1.64 \cdot t^{-1.77\times10^{-5}} + C_6
\end{aligned} \tag{4.1}$$

In figure 4.2 it is clear that as the amount of training data increases, Precision@k also increases. It is increasing with a logarithmic behavior suggesting that there is a law of diminishing returns due to the irreducible error that cannot be surpassed with larger amounts of training data, this pattern is observable for all studied markets. The difference in performance between the markets is most likely due to how much the buying patterns vary within a market. If buying patterns vary more within a market, predicting them will be less trivial.

Figure 4.3 shows the opposite relationship between Catalog Coverage and the amount of training data. The Catalog Coverage is very low for the smallest datasets. When reaching a sufficient amount of training data Catalog Coverage peaks. After this peak, Catalog Coverage decreases when more training data is added. Similar patterns are observed from all markets.

Figure 4.4 describes the Novelty of the recommendation as a function of the amount of training data. It is clear that the recommendations are significantly more novel in general in the French and German markets. These are also the two markets with the largest sets of available training data. For all markets except Germany, Novelty decreases when the amount

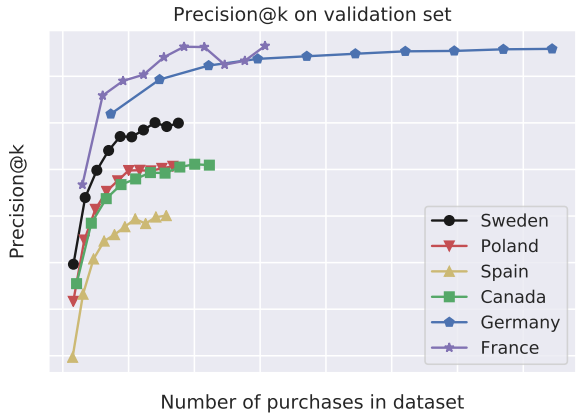


Figure 4.2: Precision@k plotted as a function of available training data. $k = 4$ was used at this time.

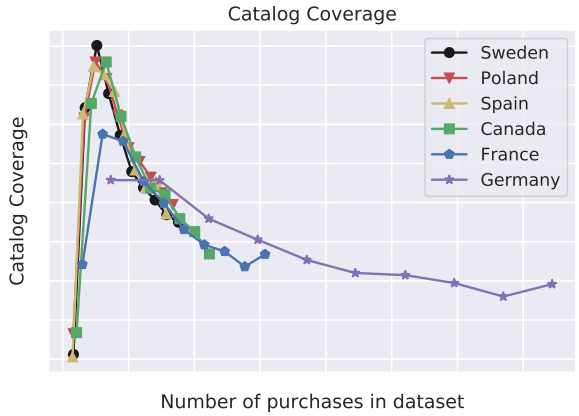


Figure 4.3: Catalog Coverage plotted as a function of available training data.

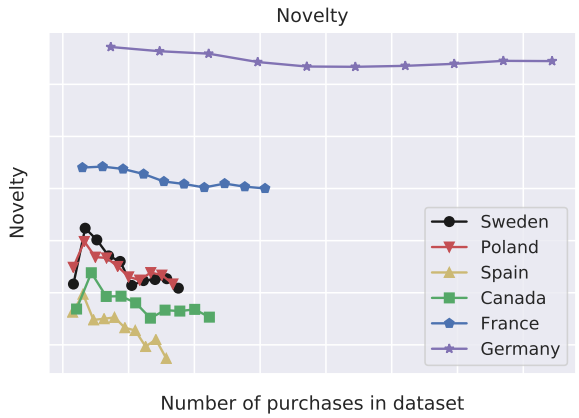


Figure 4.4: Novelty plotted as a function of available training data.

of training data increases. In the German market, where the Novelty overall is notably higher than in other markets. The difference in Novelty is too small to draw any conclusions about how it is affected by the amount of training data.

4.2 Down-sampling on excess data

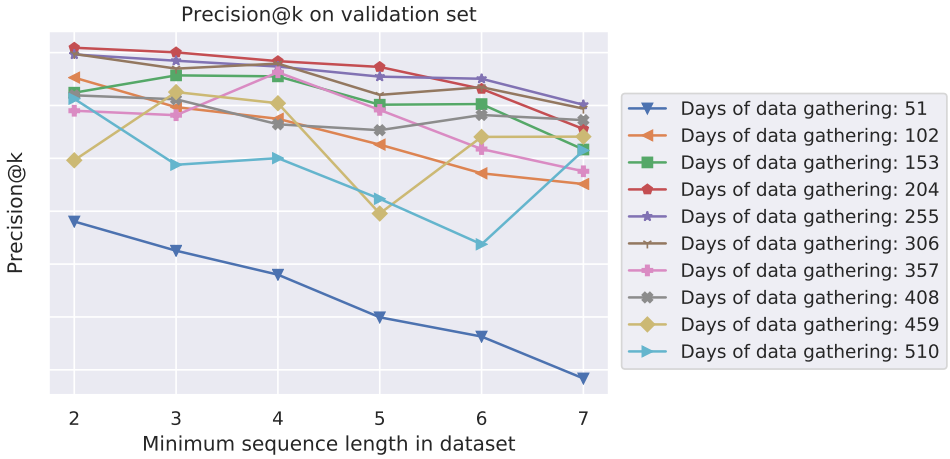
The size of the generated datasets when down-sampling using both minimum sequence length and period of data gathering are presented in table 4.2. At the time of data extraction, data had been gathered for 561 days, resulting in 10 training sets with the largest containing data from 510 days and the shortest only 51 days, together with a validation set containing data from the 51 most recent days. The numbers have been normalized between 1 and 0 to avoid disclosing any sensitive business information. The dataset with the longest period of data gathering and allowing the shortest minimum sequence length is the largest of the generated datasets.

Table 4.2: The number of purchases in each of the generated datasets, down-sampled both on the minimum sequence length allowed (columns) and the duration during which the data has been collected (rows). The largest dataset allows sequences as short as 2 products and has data collected during 510 days.

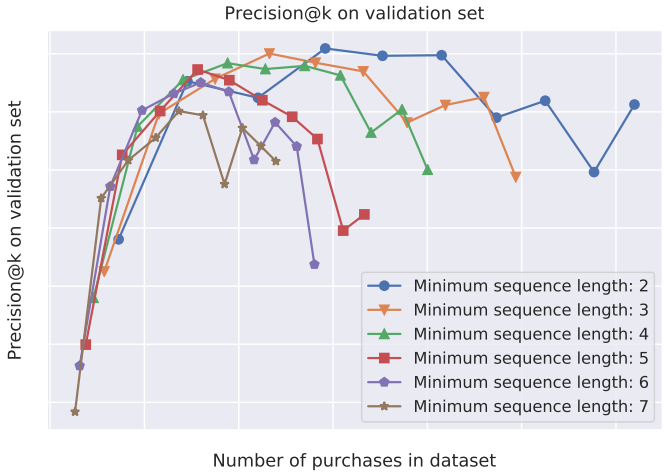
	2	**3**	**4**	**5**	**6**	**7**
51	0.117	0.093	0.074	0.061	0.051	0.043
102	0.235	0.186	0.150	0.123	0.103	0.087
153	0.356	0.282	0.227	0.188	0.157	0.134
204	0.471	0.375	0.304	0.252	0.212	0.181
255	0.569	0.454	0.368	0.307	0.258	0.220
306	0.670	0.536	0.436	0.363	0.306	0.261
357	0.764	0.611	0.497	0.414	0.349	0.299
408	0.847	0.676	0.549	0.457	0.385	0.329
459	0.931	0.742	0.602	0.502	0.422	0.361
510	1.000	0.742	0.646	0.538	0.452	0.386

The results from down-sampling on minimum sequence length and duration for data collection are presented in figures 4.5, 4.6 and 4.7. Each metric is plotted both as a function of the size of the dataset the model was trained on and the minimum sequence length allowed in that dataset. The datasets with longer periods of data collection are larger, and datasets containing the shorter sequences as well are also larger. The size of the different datasets can be viewed in table 4.2. When the duration for data collection has been increased, it has been done by adding older data to the training dataset. The smallest datasets contain data from the most recent period before the validation set was collected as described in table 4.2.

In figure 4.5a the Precision@k for the different minimum sequence lengths are presented. The Precision@k is decreasing when shorter sequences are removed from the training dataset. The models trained on data from only 51 days of data collection does not achieve the same



(a) Precision@k plotted as a function of the minimum sequence length used in the dataset, for different duration of data collection



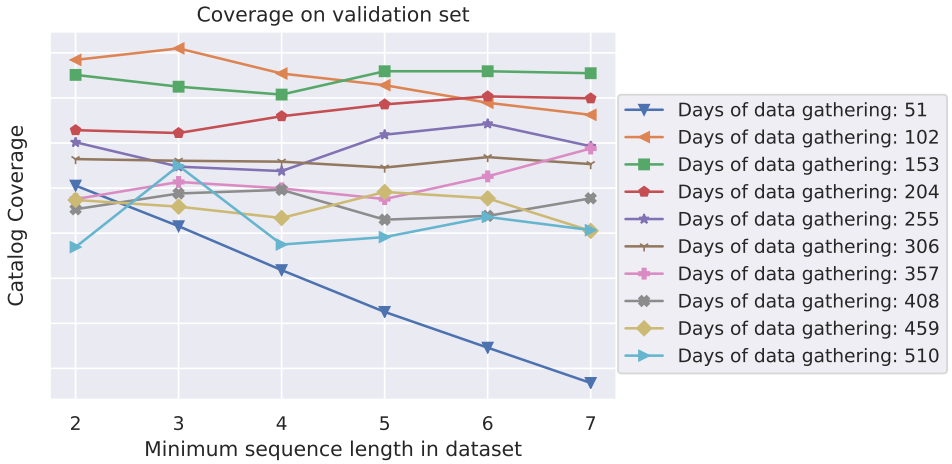
(b) Precision@k plotted as a function of the size of training dataset.

Figure 4.5: Precision@k when down-sampling on minimum sequence length and duration of data collection

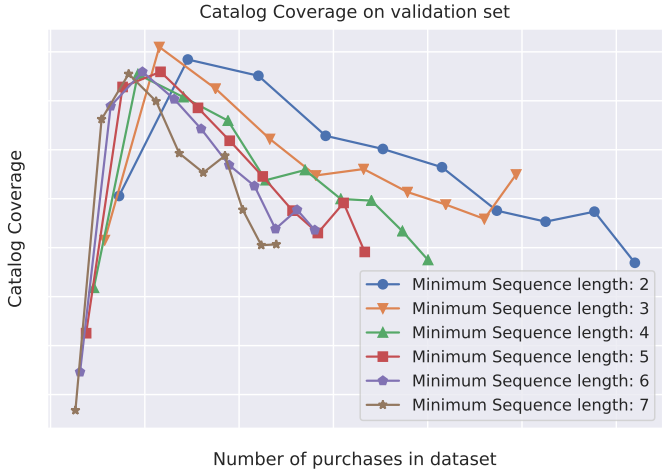
performance as the others. This is likely due to this dataset being the smallest one, and below the amount of data required to reach sufficient Precision@k performance. Apart from this, the other datasets all perform similarly.

In figure 4.5b the Precision@k for the different datasets are presented as a function of the size of the training dataset. It is possible to conclude that there is an optimal amount of data since the Precision@k is at first increasing when older data is added and then decreasing after a peak point. The location of the peak seems to depend on what minimum sequence length is allowed. When shorter sequences are removed, the Precision@k decreases more rapidly than in the datasets containing the shorter sequences.

In figure 4.6a the Catalog Coverage for the different minimum sequence lengths is presented. it is not possible to draw any significant conclusions from this graph since there is no general pattern for how the Catalog Coverage depends on the minimum sequence length.



(a) Catalog Coverage plotted as a function of the minimum sequence length used in the dataset, for different duration of data collection

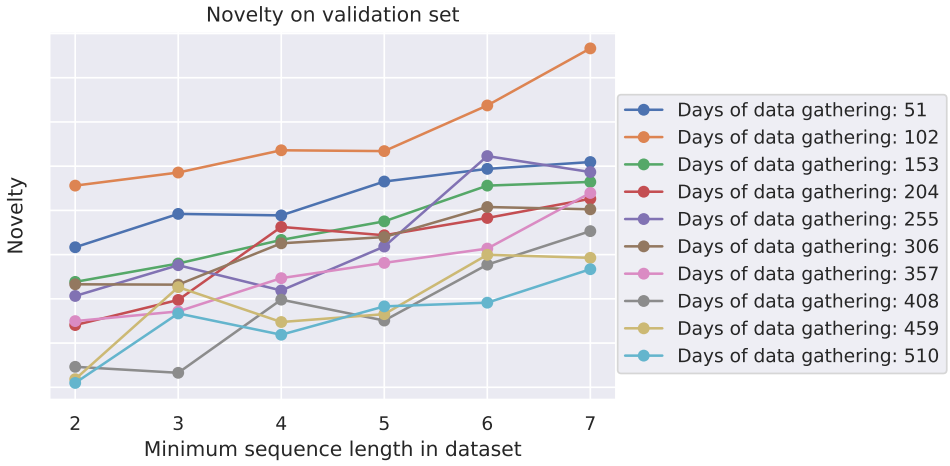


(b) Catalog Coverage plotted as a function of the size of training dataset.

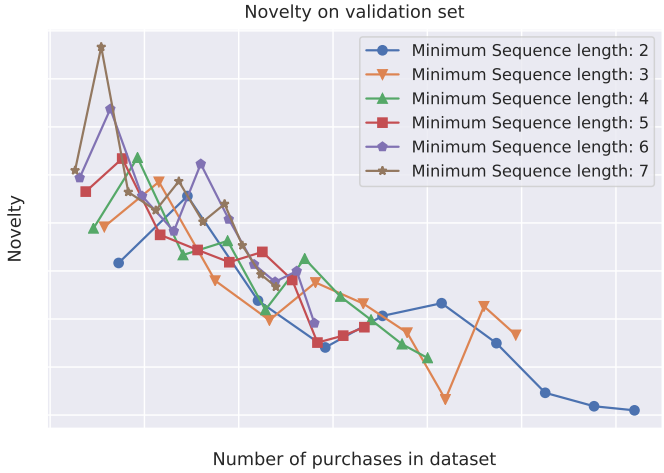
Figure 4.6: Catalog Coverage when down-sampling on minimum sequence length and duration of data collection

The models trained on datasets with 51 days of data collection again performs significantly worse than the other datasets. Here a clear pattern with decreasing Catalog Coverage is visible. This is likely due to the dataset becoming too small when filtering out shorter sequences. This is a similar pattern as to what can be seen in figure 4.3 where the smallest datasets have a lower Catalog Coverage before there is an initial peak.

In figure 4.6b the Catalog Coverage for the different datasets is presented as a function of the size of the training dataset. Similar to the results in section 4.1, Catalog Coverage decreases when the model is trained on more data. It is also clear that datasets where shorter sequences have been removed, such as the one with minimum sequence length 7, Catalog Coverage decreases more rapidly than in the datasets containing shorter sequences. This is a result which is not visible in figure 4.6a, but obvious in figure 4.6b.



(a) Novelty plotted as a function of the minimum sequence length used in the dataset, for different duration of data collection



(b) Novelty plotted as a function of the size of the training dataset.

Figure 4.7: Novelty when down-sampling on minimum sequence length and duration of data collection

In figure 4.7a the Novelty for the different minimum sequence lengths are presented. From the graph, it can be interpreted, the novelty is increasing when shorter sequences are removed from the training dataset. Removing the shorter sequences makes the recommendations more novel, meaning that less popular products are recommended to a higher extent. The line describing the dataset with 102 days of data gathering has higher novelty than the other lines, most likely due to the same reason as to why the second smallest datasets in figure 4.3 have the highest Catalog Coverage. First, the model has too little data to make recommendations for the entire catalog and only recommend items that occur more often. When more data is added the model reaches a point where it can perform recommendation for most items in the catalog. Then, when adding even more data the model gradually learns to recommend the most popular items to a higher extent as these are more accurate, but as a consequence, novelty decreases.

In figure 4.7b the Novelty for the different datasets are presented as a function of the size of the training dataset. From the graph, it can be interpreted, the novelty is decreasing when adding more and older data to the dataset. This means that models trained on larger datasets, containing data gathered for a longer duration to a greater extent recommend the most popular products, while the datasets containing data from shorter periods tend to recommend less popular items to a higher extent.

4.3 Up-sampling using data from other markets

Figure 4.8 contains the computed similarities between markets using the sequence embedding method. It is possible to interpret a correlation between geography and the computed similarities, examples of this being that the most similar countries to Germany are Austria and Netherlands, while the most similar countries to China are South Korea and Japan. It is also possible to see a pattern in European countries is similar to each other while dissimilar to Asian countries.

Some examples contradict the geographical correlation. An example of this is that the two most similar countries to Canada are France and Switzerland, while the neighboring country, US, only achieves 69% similarity. Another example of this is the similarity between Poland and Spain, which is 92%, while the similarity between the neighbors Poland and Germany is 45%.

	Germany	Japan	Netherlands	South Korea	Austria	France	Poland	Italy	Switzerland	Canada	US	Russia	UK	Spain	Sweden	China
Germany		31,2%	88,5%	26,5%	90,0%	72,5%	45,3%	62,8%	82,2%	66,7%	67,7%	25,2%	77,4%	58,0%	64,7%	24,6%
Japan	31,2%		25,8%	88,3%	31,6%	38,9%	27,5%	32,3%	32,5%	43,1%	45,2%	70,2%	25,5%	30,5%	34,8%	84,0%
Netherlands	88,5%	25,8%		27,1%	86,8%	78,6%	52,0%	60,4%	81,3%	74,7%	58,5%	27,7%	69,5%	57,7%	68,5%	25,0%
South Korea	26,5%	88,3%	27,1%		23,0%	31,9%	15,7%	17,8%	22,2%	43,1%	42,3%	76,2%	21,8%	19,6%	31,3%	92,4%
Austria	90,0%	31,6%	86,8%	23,0%		86,5%	66,4%	70,7%	93,1%	79,8%	65,3%	25,4%	84,3%	75,1%	77,3%	20,0%
France	72,5%	38,9%	78,6%	31,9%	86,5%		79,3%	69,9%	85,6%	82,3%	60,9%	40,9%	72,5%	81,1%	83,8%	26,0%
Poland	45,3%	27,5%	52,0%	15,7%	66,4%	79,3%		83,0%	76,3%	72,5%	56,9%	22,2%	69,4%	92,0%	77,8%	10,9%
Italy	62,8%	32,3%	60,4%	17,8%	70,7%	69,9%	83,0%		83,1%	62,8%	58,0%	16,9%	69,1%	85,3%	74,8%	14,0%
Switzerland	82,2%	32,5%	81,3%	22,2%	93,1%	85,6%	76,3%	83,1%		81,6%	64,8%	26,1%	84,6%	85,2%	83,0%	22,7%
Canada	66,7%	43,1%	74,7%	43,1%	79,8%	82,3%	72,5%	62,8%	81,6%		69,6%	46,1%	77,9%	77,8%	80,6%	41,5%
US	67,7%	45,2%	58,5%	42,3%	65,3%	60,9%	56,9%	58,0%	64,8%	69,6%		38,6%	74,2%	61,2%	59,4%	42,0%
Russia	25,2%	70,2%	27,7%	76,2%	25,4%	40,9%	22,2%	16,9%	26,1%	46,1%	38,6%		12,7%	18,0%	44,2%	67,5%
UK	77,4%	25,5%	69,5%	21,8%	84,3%	72,5%	69,4%	69,1%	84,6%	77,9%	74,2%	12,7%		84,5%	67,5%	22,2%
Spain	58,0%	30,5%	57,7%	19,6%	75,1%	81,1%	92,0%	85,3%	85,2%	77,8%	61,2%	18,0%	84,5%		80,6%	18,1%
Sweden	64,7%	34,8%	68,5%	31,3%	77,3%	83,8%	77,8%	74,8%	83,0%	80,6%	59,4%	44,2%	67,5%	80,6%		27,3%
China	24,6%	84,0%	25,0%	92,4%	20,0%	26,0%	10,9%	14,0%	22,7%	41,5%	42,0%	67,5%	22,2%	18,1%	27,3%	

Figure 4.8: Similarities between markets obtained from cosine similarity on market embeddings.

To test the similarity metric, using the method described in 3.4, three suitable markets were selected. These three markets and the similar and dissimilar secondary markets that were selected are presented in table 4.3 where the corresponding similarities also are presented.

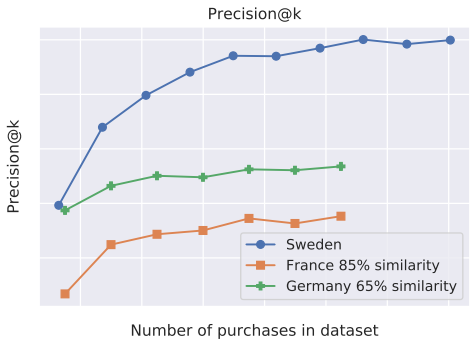
First, analysing the achieved Precision@k when validation models trained on data from the similar (**B**) and dissimilar markets (**C**) using validation data from the market (**A**). The results are presented in figure 4.9.

As a baseline, the result from section 4.1, when a model was trained on data from the same market as it is then validated on has been plotted. This is to be able to judge how well

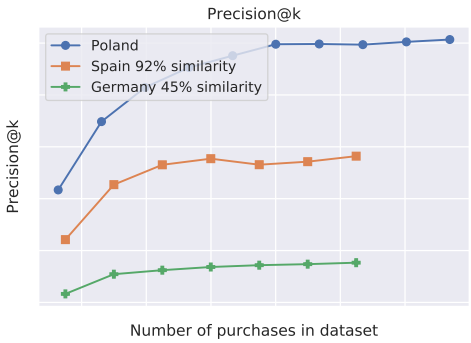
Table 4.3: The combinations of markets selected for testing the similarity metric between markets.

Market to test on (A)	Similar market (B)	Disimilar market (C)
Sweden	France (85%)	Germany (65%)
Poland	Spain (92%)	Germany (45%)
Germany	Netherlands (89%)	Spain (58%)

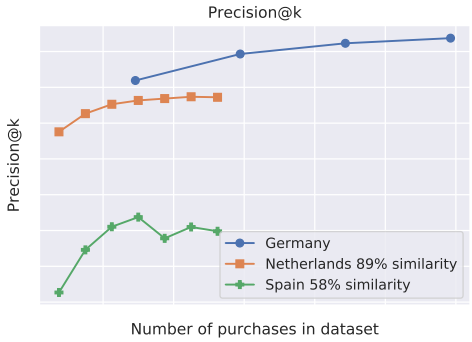
the models with data from one market and then validated on another perform, compared to using data from the same market where it was validated.



(a) Performance of three models, trained on datasets from Germany, France and Sweden, all tested on a validation dataset from Sweden.



(b) Performance of three models, trained on datasets from Spain, Germany and Poland, all tested on a validation dataset from Poland.



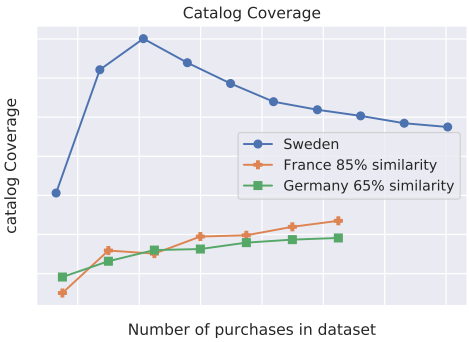
(c) Performance of three models, trained on datasets from Spain, Netherlands and Germany, all tested on a validation dataset from Germany.

Figure 4.9: Precision@k plotted as a function of dataset size when testing how well models trained on data from a secondary market performs on a local validation dataset.

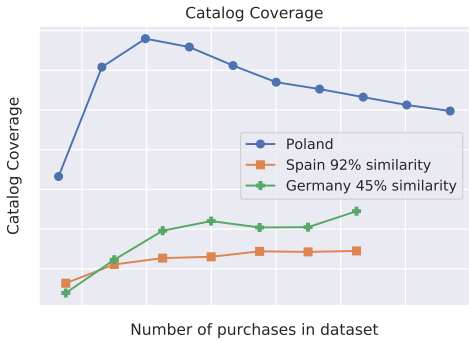
In figure 4.9b and 4.9c the models trained on data from more similar markets according to the suggested similarity metric achieves higher Precision@k than the models trained on data from a dissimilar market as was predicted by the proposed similarity metric. However, in figure 4.9a the model trained on data from the dissimilar market achieves a slightly higher Precision@k than the model trained on data from the similar market.

It is worth noting that there is a significant difference in achieved Precision@k between the baseline and the models trained on data from other markets, for all examples in figure 4.9. The exception is the case when data from the Netherlands has been used to train a model which has been validated on data from Germany, where Precision@k levels are very similar.

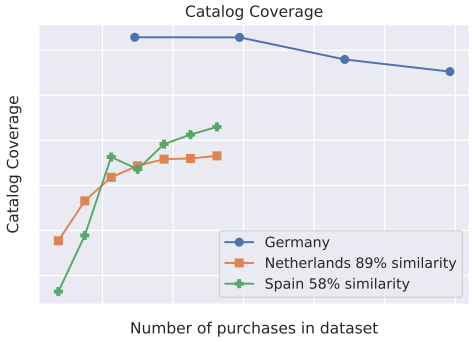
In all three examples in figure 4.9 there is a significant difference in the Precision@k levels between the two markets, confirming that from what market the training data is extracted is important.



(a) Performance of three models, trained on datasets from Germany, France and Sweden, all tested on a validation dataset from Sweden.



(b) Performance of three models, trained on datasets from Spain, Germany and Poland, all tested on a validation dataset from Poland.

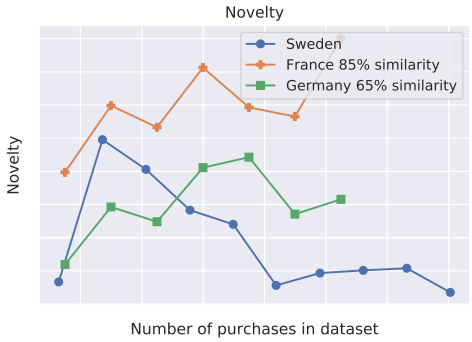


(c) Performance of three models, trained on datasets from Spain, Netherlands and Germany, all tested on a validation dataset from Germany

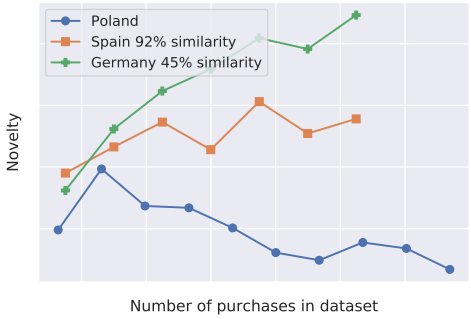
Figure 4.10: Catalog Coverage plotted as a function of dataset size when testing how well models trained on data from a separate market performs on a local validation dataset.

In figure 4.10 the resulting Catalog Coverage from the experiments is presented. Overall, the Catalog Coverage is unquestionably lower than what a model trained on data from the same market as to where the validation data originates from achieves. Most likely this is a consequence of the set of products available on each market varies. Therefore the model trained on data from another market will not be able to recommend products that only exist in the market where the validation is done as the training dataset did not include these products, while the model trained on data from the same market as to where the validation data originates from includes all products available on that market.

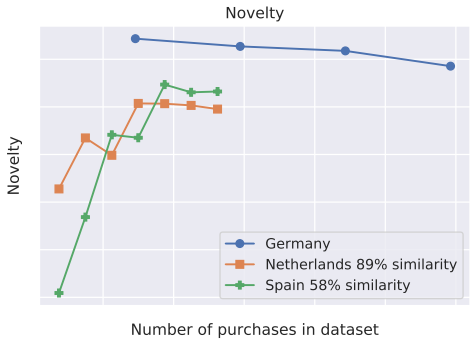
There is no significant difference in Catalog Coverage to be seen in between the models trained on data from similar or dissimilar markets.



(a) Performance of three models, trained on datasets from Germany, France and Sweden, all tested on a validation dataset from Sweden.



(b) Performance of three models, trained on datasets from Spain, Germany and Poland, all tested on a validation dataset from Poland.



(c) Performance of three models, trained on datasets from Spain, Netherlands and Germany, all tested on a validation dataset from Germany

Figure 4.11: Novelty plotted as a function of dataset size when testing how well models trained on data from a separate market performs on a local validation dataset.

In figure 4.11 the Novelty plots from the experiments are presented. There is no clear pattern to be found regarding if the model trained on data from a similar or dissimilar market performs better or worse compared to the baseline.

The plots all indicate that Novelty increases when a larger dataset from another market has been used to train the model, which differs from the results in section 4.1 when the model has been trained and validated on data from the same market. In those cases (figures 4.4 and 4.7) the Novelty tends to decrease when the size of the training dataset has been increased. Since Novelty indicates how well the RS recommends less popular items, this result could be an indication of that the most popular items differ between markets. When using a model trained on a large dataset from Germany it will according to our earlier results recommend the most popular items from Germany, which is considered a less novel recommendation. If these items that are popular in Germany are less popular in Poland, those recommendations will be considered more novel, in the Polish market.

Chapter 5
Discussion

In this section, we discuss the results presented in the previous section regarding the three main experiments.

5.1 Performance as a function of dataset size

The results in this experiment confirm that there is an irreducible error that the model cannot overcome. The precision achieved in this thesis can, however, most likely be reduced using tuning of the model and hyperparameter optimization. As a consequence of this there is also a maximum level of accuracy and Precision@k that can be achieved.

High scores in Catalog Coverage, Novelty, and Precision@k are desirable, and there seems to be a trade-off between the more traditional accuracy metrics such as Precision@k and the diversity metrics such as Novelty and Catalog Coverage. This is to be expected due to the long tail distribution in sales, as the most accurate recommender converges to only recommend the few very popular products, leaving the less popular products out and therefore decreasing the Catalog Coverage.

Precision@k and similar accuracy metrics cannot be the only metric considered when designing an RS as it only describes how well the model was able to predict what the customers in the dataset bought without the influence of an RS. Since a good RS shows the customer something he or she was not aware of but might be interested in, it is not desirable to only mimic current customer behavior. Instead, Novelty and Catalog Coverage, or other similar metrics, must also be considered to be able to include the less obvious recommendations, the ones that customers would not have found themselves. If only Precision@k would be considered, the results in this test indicate that more data is always better.However, since the performance gained per added data point is decreasing it might not be necessary to keep adding data as the improvements with ever-increasing datasets are negligible. The decreasing learning per added datapoint is expected due to the power-law behavior of the learning curves and the law of diminishing returns.

Instead, if only Catalog Coverage and Novelty would be considered, the optimal dataset size to use when training this model would be significantly smaller than if only Precision@k would be considered. However, at the optimal amount of training data for Catalog Coverage and Novelty, Precision@k has not reached near its maximum level. If taken to the extreme, the highest Catalog Coverage and Novelty would be achieved with a recommender sampling random products from the catalog without considering any data, however, those recommendations will in most cases be useless to the customer.

There is an apparent trade-off between the metrics, and as a consequence, each designer of an RS needs to choose what to prioritize. These results are consistent with previous research such as in [10, 30, 43]. The more accurate but obvious, or the less obvious, and accurate but more novel RS. If a company needs to sell more of less popular products a more novel RS is desirable. If the product catalog is less known, and even the most popular products are unknown to the average customer a more accuracy focused RS could be better, as even recommendations for very popular products will be novel and providing new information for the customer.

It is however clear that adding more data does not unambiguously improve performance. Depending on whats prioritized for that specific implementation an optimal size of a dataset can be found.

5.2 Down-sampling excess data

5.2.1 Down-sampling by removing shorter sequences

When down-sampling by removing shorter sequences and thereby making the dataset smaller, Novelty increased, Precision@k decreased but Catalog Coverage remained unchanged.

The Novelty increasing could potentially be due to these short sequences to a higher extent contains popular products, while the longer sequences contain some popular products but also some less popular complementary products. As a consequence of removing the shorter sequences, these less popular complementary products will become more prominent in the dataset. When the less popular products are more prominent in the training dataset, they will occur more often in the recommendations. This indicates that the long tail sales distribution is even more notable among the purchases containing few products.

Interestingly, Catalog Coverage was unchanged when the shorter sequences were removed, i.e the size of the training dataset decreased. This contradicts the results from section 5.1. Building on the reasoning behind why Novelty increased, one possible reason for this could be that the shorter purchases, in general, contain the most popular products. When removing these sequences, the most popular products are likely still very prominent in the dataset even if less prominent than if the shorter sequences were kept, which means that they still occur frequently. The RS will therefore still recommend these items, but not as often as if the shorter purchases were present in the dataset. While the less popular items become more prominent in the dataset, it is when these items occur too seldom be recommended Catalog Coverage decreases. This could explain why the Catalog Coverage is constant while novelty increases, all products are still present in a number large enough to be included in the recommendations by the RS.

The Precision@k decreasing when removing the shorter sequences is then a consequence

of the novelty increasing. The most popular products are recommended to a lesser extent in favor of the less popular products. These less popular products are likely less accurate, and thereby Precision@k decreases. The decrease could also be explained in part by the datasets being smaller, and as seen in the previous experiment this should lead to a decrease in Precision@k. It is not possible to tell, from this experiment, which of these effects that are the greatest cause for the decrease.

5.2.2 Down-sampling by removing older datapoints

When removing older data from the dataset and thereby decreasing the size of the dataset the Catalog Coverage and the Novelty increased. This is likely due to the datasets being smaller, which is a pattern that can also be seen in the previous experiment. These results are comparable due to the distribution of products present in these datasets being equal or very similar to the distribution in datasets in the previous experiment. Another explanation for the Novelty increasing when removing older data is that seasonal items might be more prominent in datasets gathered during a shorter period. These would be considered more novel as they might not be very popular if one is considering the sales during the full year, but during a short period, they might be very popular. If this period is prominent in the dataset, these seasonal items will be recommended to a higher extent and thereby increasing novelty.

Precision@k increases at first when adding older data, until the Precision@k peaks and starts to decrease when more data is added. This result is different from the results in the first experiment where Precision@k was increased for larger datasets. This implies that older data is less accurate for making up to date recommendations and that the buying pattern changes with time. This might be explained by the fact that the product catalog changes regularly when some products are removed and some new introduced, but also that preferences might vary with season, e.g red might be a more popular color in countries celebrating Christmas in the months of November and December than the other months.

All metrics considered in this thesis decrease when adding older data, if the previous amount of data was enough to reach the peak in performance, which confirms the hypothesis that recent data is important when training RSs.

5.3 Up-sampling using data from other markets

The proposed similarity metric shows some promising results in the sense that it captures geographical patterns. Neighboring countries having a similar taste for interior design and similar financial situations are prejudice one might have, which to some extent is being confirmed by this similarity metric created from market embeddings.

When testing this metric by training an RS model on data from one market and validating it on another the achieved Precision@k is higher on the market deemed similar by the similarity metric in two of the three cases. The RS model achieved spectacular results when using data from the Netherlands, validated on data from Germany, achieving almost as high Precision@k as if data from Germany had been used for training. In all cases there is a significant difference in achieved Precision@k between the models trained on data from secondary markets, indicating that it is of great importance which market the data is selected

from. In all cases using data from the same market for training and validation produces the best results.

The results regarding novelty are ambiguous, and it is difficult to draw any conclusions of significance here. A possible explanation could be that products which are very popular in one country could be less popular in another, therefore being considered as more novel recommendations. This could explain why the novelty is higher in some cases when data from a secondary market has been used, but the difference is not large enough to be significant.

The Catalog Coverage is lower when models trained on data from a secondary market is analyzed. This is an argument against using data from secondary markets since the product catalogs in all countries are slightly different. The items that do not exist in the market where the training dataset is taken from will never occur in the recommendations. However, in a situation where there is a lack of data from one market, the small amount of data available would be used in firsthand and then complemented with data from a similar market. This would make the Catalog Coverage better than in the test case presented in this thesis where only data from another market was used, instead of complementing another dataset.

The results, therefore, indicate that it is possible to use data from another market if there is a lack of data on some market, and it is of great importance which market the data is taken from. However, it is not possible from these experiments only to determine if this similarity metric is a sufficient method for selecting which market to use data from as our results are equivocal. Further evaluating this is left as a question for future research.

Chapter 6
Summary

6.1 Conclusions

In this section, the conclusions regarding all research questions in this thesis are presented.

RQ1: How is the performance of this particular RS affected by the amount of available training data?

Performance in this thesis has been defined as a combination of three main metrics: Catalog Coverage, Precision@k, and Novelty. Precision@k increases as a logarithmic function of dataset size, meaning that the information gained by each added data point is decreasing. When the dataset size is increasing Precision@k asymptotically approaches a theoretical limit due to the irreducible error in the prediction.

Catalog Coverage and Novelty decreases as a function of dataset size, as this particular RS learns to recommend the most popular products to a high extent, while not recommending the less popular items.

RQ2: Is there an optimal amount of training data which gives the best overall RS performance?

There is an apparent trade-off between accuracy focused metrics such as Precision@k and diversity-focused metrics such as Catalog Coverage and Novelty. Hence, an optimal amount of data for training the RS can be defined if one's notion of optimal is clearly defined. Depending on what is prioritized when an RS is designed, the optimal amount of data varies. In this case, if accuracy and Precision@k are of high importance more data is better, but if Catalog Coverage and Novelty are important less data could be used without sacrificing too much accuracy and Precision@k. In summary, there is an optimal amount of training data to

be used to achieve the best overall performance, but it's necessary to have a clear definition of performance.

RQ3: Is it possible to select a more relevant subset of this data to achieve even better performance?

If there is an excess of data, the results in section 4.2 suggest that removing older data improves Precision@k, Catalog Coverage, and Novelty. The results also imply that removing the shortest purchase sequences from the dataset increases Novelty while sacrificing some Precision@k. As discussed earlier, what to prioritize in this case is a question for the designer of the RS. It is however clear that more recent data is better, and if the data available is enough to reach sufficient levels of Precision@k, it is beneficial to remove the oldest data points in the dataset.

RQ4: Is it possible to complement the data using data from a secondary market?

The results in section 4.3 show that it is possible to use data from a secondary market. However, what market the data is selected from is of great importance. The performance depends on what market the data was taken from and varies greatly. In none of our experiments the models trained on data from another market were able to perform better or as well as a model trained on data from the same market as it was validated on.

To decide what market to select data from, a similarity metric between markets was developed using deep learning embeddings. This metric showed some promise, but more research is needed on the topic.

6.2 Future Work

In this section, a few future areas of research to build on the results from this thesis are presented.

6.2.1 Online evaluation

Offline evaluation of RSs, as conducted in this thesis, only measures the performance of an RS on a dataset of data from prerecorded sessions where customers browsed the website without the RS. How a customer would act if the RS was present cannot fully be concluded from the offline evaluation. To build on the results from this thesis, online evaluation, and A/B testing of RS performance as a function of the training data should be considered.

6.2.2 Biases introduced when customers opt-out of data collection

Giving customers the option to opt-out from personalized content and data tracking could introduce some biases in the dataset. For example, if young customers opt-out to a higher

extent than older customers, products that are often bought by young customers will be less prominent in the dataset. Which biases that might be introduced and how to handle these is a question for future research.

6.2.3 Further research on a Similarity metric between markets

The work in this thesis could not draw any significant conclusions about whether the proposed similarity metric between markets functioned as intended. More research is needed in this area. Several other techniques could potentially be well suited for the task, such as Doc2Vec and Meta-Prod2Vec.

References

[1] Shun-ichi Amari. A universal theorem on learning curves. *Neural Networks*, 6:161–166, 12 1993.

[2] James Bennett and Stan Lanning. The netflix prize. 2007.

[3] Pablo Castells, Saúl Vargas, and Jun Wang. Novelty and diversity metrics for recommender systems: Choice, discovery and relevance. *Proceedings of International Workshop on Diversity in Document Retrieval (DDR)*, 01 2011.

[4] Òscar Celma. *The Long Tail in Recommender Systems*, pages 87–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[5] Kyunghyun Cho. Introduction to neural machine translation with gpus. `https://devblogs.nvidia.com/introduction-neural-machine-translation-with-gpus/`. Visited on 2020-04-27.

[6] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. *CoRR*, abs/1907.06902, 2019.

[7] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 1775–1784, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[8] Kim Falk. *Practical Recommender Systems*. Manning Publications, 2019.

[9] Ibtissem Gasmi, Hassina Seridi-Bouchelaghem, Hocine Labar, and Abdel Karim Baareh. Collaborative filtering recommendation based on dynamic changes of user interest. *Intelligent Decision Technologies*, 9:271–281, 09 2015.

[10] Mouzhi Ge, Carla Delgado, and Dietmar Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. *RecSys'10 - Proceedings of the 4th ACM Conference on Recommender Systems*, pages 257–260, 01 2010.

[11] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[12] Google. Keras embeddings layer. `https://keras.io/layers/embeddings/`. Visited on 2020-04-27.

[13] Google. Keras losses. `https://keras.io/api/losses/`, note = Visited on 2020-06-11.

[14] Google. Measuring similarity from embeddings. `https://developers.google.com/machine-learning/clustering/similarity/measuring-similarity`. Visited on 2020-04-27.

[15] Ingka Group. Ingka group launches data promise to customers. `https://www.ingka.com/news/ingka-group-launches-data-promise-to-customers/`, 01 2020. Visited on 2020-03-17.

[16] Asela Gunawardana and Guy Shani. *Evaluating recommender systems*, pages 265–308. Springer, 01 2011.

[17] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory F. Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwarym, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *CoRR*, abs/1712.00409, 2017.

[18] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[20] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, January 2004.

[21] Se Won Jang, Simon Kim, and JeongWoo Ha. Graph-based recommendation systems : Comparison analysis between traditional clustering techniques and neural embedding. 2017.

[22] Daniel Johnsson. Composing music with recurrent neural networks. `http://www.hexahedria.com/2015/08/03/composing-music-with-recurrent-neural-networks/`. Visited on 2020-04-27.

[23] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Visited on 2020-04-27.

[24] Andrej Karpathy. Cs231n: Convolutional neural networks for visual recognition. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 08 2015. Visited on 2020-03-31.

[25] Yehuda Koren and Robert Bell. *Advances in Collaborative Filtering*, pages 145–186. Springer US, Boston, MA, 2011.

[26] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[27] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[28] Ian MacKenzie, Chris Meyer, and Steve Noble. How retailers can keep up with consumers. `https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers`. Visited on 2020-04-27.

[29] Benjamin Marlin. Modeling user rating profiles for collaborative filtering. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 627–634, Cambridge, MA, USA, 2003. MIT Press.

[30] Sean McNee, John Riedl, and Joseph Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. *CHI '06 Extended Abstracts*, pages 1097–1101, 04 2006.

[31] Prem Melville and Vikas Sindhwani. Recommender systems. In Claude JSammut and Geoffrey I. Webb, editors, *Encyclopedia of Machine Learning*, chapter 00338. Springer, New York, 2017.

[32] Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[33] Christopher Olah. Understanding lstm networks. `http://cs231n.github.io/neural-networks-1/`. Visited on 2020-03-31.

[34] Jeffrey Pennington, Richard Socher, and Christoper Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014.

[35] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. pages 175–186, 1994.

[36] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. In *Recommender systems handbook*, pages 1–34. Springer, 2015.

[37] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 791–798, New York, NY, USA, 2007. Association for Computing Machinery.

[38] Daniela Sanchez Santolay. Using recurrent neural networks to predict customer behavior from interaction data. *University of Amsterdam*, 06 2017.

[39] Stuart A. Thompson and Charlie Warzel. Twelve million phones, one dataset, zero privacy. *The New York Times*, Dec 2019.

[40] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec - product embeddings using side-information for recommendation. *CoRR*, abs/1607.07326, 2016.

[41] WolframMathWorld. Geometric centroid. `https://mathworld.wolfram.com/GeometricCentroid.html`. Visited on 2020-05-14.

[42] Liang Xiang and Qing Yang. Time-dependent models in collaborative filtering based recommender system. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, page 450–457, USA, 2009. IEEE Computer Society.

[43] T. Zhou, Z. Kuscsik, J.G. Liu, M. Medo, J.R. Wakeling, and Y.C. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

[44] Andrej Zwitter. Big data ethics. *Big Data & Society*, 1(2):2053951714559253, 2014.

Appendices

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-08
EXAMENSARBETE Data Optimization for a Deep Learning Recommender System
STUDENT Gustav Hertz
HANDLEDARE Patrik Persson (LTH), Emil Joergensen (IKEA)
EXAMINATOR Flavius Gruian (LTH)

Användardatans påverkan på ett rekommendationssystem

POPULÄRVETENSKAPLIG SAMMANFATTNING **Gustav Hertz**

Användardata krävs för att bygga personliga rekommendationssystem. I detta arbete har jag undersökt hur mängden data påverkar rekommendationsystemets prestanda, samt hur data bör väljas då det finns ett överflöd eller underskott av data för träning av rekommendationssystemet.

Rekommendationssystem är algoritmer som föreslår innehåll för användare, främst på internet. Systemen är generellt utformade för att ge en personligt anpassad användarupplevelse genom att rekommendera innehåll som det är sannolikt att användaren kommer att uppskatta. Inom näthandel på IKEA används ett rekommendationssystem som bygger på ett Long-Short Term Memory deep learning nätverk. Denna modell använder sig av en sekvens av produkter som användaren har lagt i varukorgen för att förutsäga vilka produkter som är mest sannolika nästa inköp. Dessa produkter är sedan rekommenderade till användaren.

Vilka produkter som är sannolika som nästa inköp baseras på stora mängder av data om tidigare användares köp. IKEA vill erbjuda användare valet om de vill att deras data ska lagras och användas. Det kommer att leda till en minskad insamling av data för träning av rekommendationssystemen. I detta arbete har jag undersökt hur minskad insamling av data påverkar rekommendationssystemets kvalitet.

Kvalitet definieras här som en kombination av träffsäkerhet i rekommendationerna, hur stor del av produktkatalogen som rekommenderas samt till vilken grad systemet rekommenderar produkter som inte tillhör storsäljarna. Det är önskvärt att rekommendera stora delar av produktkatalogen samt produkter som inte tillhör storsäljarna för att skapa rekommendationer som användarna inte hade hittat själva men som ändå är värdefulla. Rekommendationsssystem som endast rekommenderar de mest sålda produkterna tillför inte så mycket värde för användarna.

Undersökningarna visar att mer data inte är ensidigt positivt för rekommendationerna. Mer data ledde till att en mindre andel av produktkatalogen rekommenderades och att storsäljarna rekommenderades i högre utsträckning. Träffsäkerheten ökar visserligen med större mängder data, men förbättringen per tillagd datapunkt är avtagande.

Som en följd av att mer data inte entydigt är av godo undersöktes hur den mest relevanta mängden data kan väljas då det finns tillgång till mer data än nödvändigt. Undersökningen visade att när rekommendationssystemet tränades på äldre data så presterade det betydligt sämre. För att träna modellen bör alltså den senast tillgängliga data väljas, i tillräckligt stor mängd för att nå tillfredställande träffsäkerhetsnivåer.

Slutligen undersöktes det om det är möjligt att komplettera dataset från små marknader med data från andra marknader. Undersökningen visade att data från sekundära marknader fungerade för att träna rekommendationsystemet på marknader som saknar tillräckliga mängder data. Vilken marknad data togs från spelade stor roll för kvalitén på rekommendationerna, detta då köpbeteenden skiljer sig olika mycket mellan länder. Ett mått för att mäta likheter i köpbeteende mellan marknader utvecklades och visade lovande tendenser i att kunna förutsäga vilka marknader som hade liknande köpbeteenden, men ytterligare undersökningar krävs på området.