Designing and Evaluating a Board Game Recommender System

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Michael Ion, B.Sc.
Registration Number 1005233

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Hannes Werthner
Assistance: Univ.Ass. Dimitris Sacharidis, Ph.D.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Vienna, 12th December, 2018

Michael Ion | Hannes Werthner

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Michael Ion, B.Sc.
michael.antonius.ion@gmail.com

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. Dezember 2018

Michael Ion

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Board games have gained an unbelievable traction over the last few years. The number of board game releases grows exponentially and customers have a variety of choice like never before. This increases decision paralysis and makes it harder to decide what game to buy next. Consequently, recommendation systems were applied in various domains over the last decade to relieve this issue for customers. In this thesis, we will investigate and create various recommendation systems for this specific use case. We apply both collaborative as well as content-based approaches. We conclude the thesis with a thorough evaluation of the different recommender systems.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Abstract v

Contents vii

1 Introduction 1
1.1 Motivation . . . 1
1.2 What problem needs to be solved? . . . 3
1.3 Aim of the work . . . 3
1.4 Methodological approach . . . 4

2 Background 7
2.1 History of Recommender Systems . . . 7
2.2 Types of Recommender Systems . . . 8

3 History of Board Games & Data 23
3.1 History of Boardgames: Then to Now . . . 23
3.2 The Data - boardgamegeek.com . . . 26
3.3 Collecting the Data & Data Models . . . 28
3.4 Available board game recommendation systems . . . 29

4 Recommenders 35
4.1 Implementation . . . 35
4.2 Collaborative Filtering Approaches . . . 37
4.3 Content-Based Approaches . . . 42
4.4 Hybrid Recommender . . . 44
4.5 Baseline: Popularity-Based Recommender . . . 45

5 Software 47
5.1 Existing Recommendation Software . . . 47
5.2 Applied Software and Technologies . . . 58

6 Results 65
6.1 Evaluation Methodology . . . 65
6.2 Evaluation Metrics . . . 66

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

6.3 Training and Test Data Selection 70
6.4 Discussion of Results 72
6.5 Summary of Results 76

7 Summary & Future work 79
7.1 Summary of this thesis 79
7.2 Answered Research Questions 79
7.3 Opportunities for future research - open issues 80

List of Figures 83

List of Tables 87

Bibliography 89

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 1

Introduction

In this chapter, we will give an introduction to the topics that are tackled by this thesis. We will establish the motivation behind this research subject and the end goal of the thesis. We will also present the methodological approach taken to fulfill the research goals.

1.1 Motivation

1.1.1 The rise of the Recommender Systems

In the last few years, the practical use of Recommender Systems (RSs) has skyrocketed. They have become very common in a wide range of areas. Most commonly, they are associated with different consumer products like movies, music or news paper articles. There are a lot of companies that come to mind: Streaming provider Netflix is known for heavily including recommendations for movies and TV series in the User Interface (UI) of their product. The New York Times or Wall Street Journal (and nearly every major newspaper publisher) relies on the use of RSs to find interesting articles for their readers. E-Commerce pioneer Amazon was one of the first companies to introduce recommendations for general items, based on the sales habits of its customers("Customers who bought this...").

RSs have become ubiquitous tools, aiming to save time and energy for users, while increasing traffic and sales for providers. The economic success of these systems has been shown in increasing sales volume and sales diversity [LH14]. For consumer media, there is a lot of academic research available on the technical features of these media recommender systems, as a simple search on Google Scholar shows [OTTK12] [SKKR01]. It is evident that Recommender Systems can provide a valuable service in domains where users are faced with decisions between numerous options. For these reasons, we think it

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

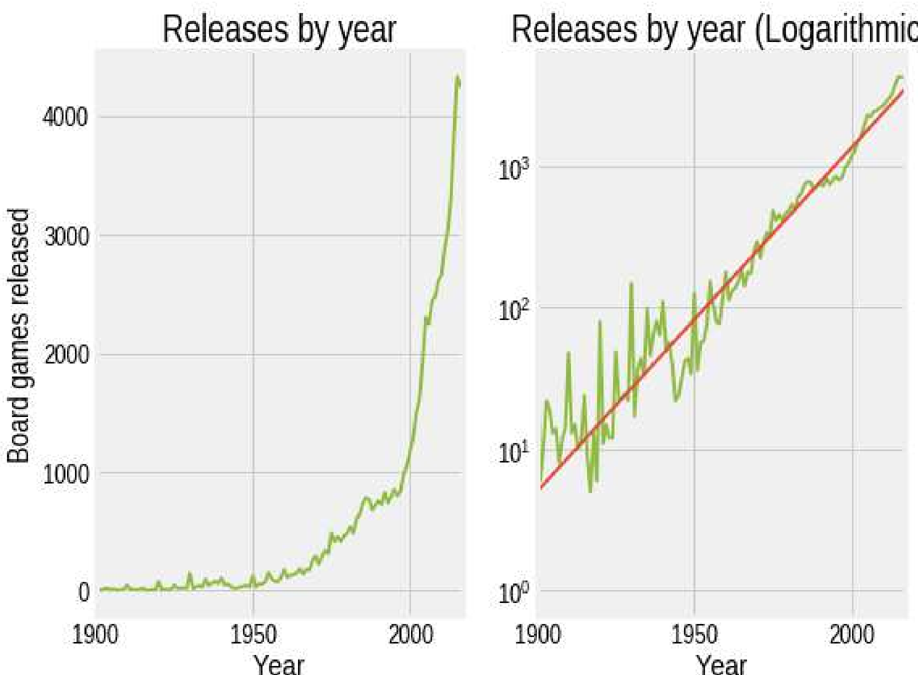


Figure 1.1: Released games per year 1900 - 2010. Left: Linear Scale, Right: Logarithmic Scale [Vat17]

is interesting to investigate the use of a RSs in the domain of Board Games. But firstly, we need to specify the current development of this consumer media.

1.1.2 The comeback of board games

Sales and releases of new board games have been increasing proportionally with the rise of video games, which is about 25% per year [Duf14] [Win14]. Never before in history have there been as many board game releases as today. There are a variety of theories why this has happened, despite the still rising popularity of digital games. For one, digital versions of classic board games might have helped to keep this form of entertainment alive. But the other, probably more relevant reason is, that, as the modern working environment and everyday life is being pervaded more and more with Computers, smart phones and other new technologies, people might look stronger for a pastime that they can enjoy with friends and families that doesn't involve "staring at a screen".

The total market is estimated at around $750 million - if we would include collectible tabletop games, this number would double [Gri17]. Even new research has been made about this board gaming "phenomenon" [Woo12]. In fact, never before have there been so many board games released as today [Vat17]. Figure 1.1 highlights the yearly developments.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

1.2 What problem needs to be solved?

The increasing popularity is most certainly a great development for the board game community. As for every type of item, good or form of popular art or entertainment, more choice implies more supply, which further implies, overall, a quality improvement in order to survive against the competition. But it has been shown, that too much freedom of choice and too many options can make people feel lost and overwhelmed. Customers can experience what is called “decision paralysis” [Sch09]. This can help the customer facilitate their choices, save them time and protect them from feeling burnt out. It is also one of several reasons why streaming providers like Netflix or Spotify use RSs.

To solve this, we propose the application of a RS in the domain of Board games. We develop a board game recommender system, where the users specify one or multiple games they like as an input. The system processes this information based on game characteristics and existing user ratings. It outputs a list of games they will likely enjoy as well. To properly construct a reliable system, we need appropriate data, techniques to compute predictions and methods for evaluation. We discuss these in the following.

However, there is no straight answer to which techniques are best suitable to our problem of recommending board games. As we have both user ratings as well as user-provided data about the games themselves available, we can develop a new approach for this specific use case. Especially for content-based techniques, where we use multinominal attributes like “Mechanics” or “Family” of board games (where one item can have multiple values in one attribute),we need to investigate fitting methods and adapt them, for example TF-IDF (Term Frequency - Inverse Document Frequency).

1.3 Aim of the work

The aim of this work is to create a RS that provides board game suggestions. On top of that, we test and evaluate different approaches of doing so. It should execute as follows: The recommender system takes one or several games as an input. The output consists of several board game titles that is very likely enjoyed by a user who also liked the games provided as the input. The computation of the various recommendations is based on different strategies, which need to be explored. Additionally, we use our data for the evaluation and comparison of the performances of our various recommendation techniques.

1.3.1 Research Questions and Expected Results

With our work, we aim to answer following research questions.

- **Q1**: Which recommendation methods are appropriate for recommending board games?
- **Q2**: Which attributes can be used for recommendations?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **Q3**: Which metrics can be applied for measuring the quality of board game recommendations?

  In addition to providing answers to these questions, we build models that can take games and compute recommendations, based on various approaches. We also develop metrics to evaluate, validate and justify our findings.

1.4 Methodological approach

1. Literature Review

   Relevant literature that informs us about the best way to accomplish the tasks at hand is gathered and dissected.

2. Data

   To compute similarity between board games, we use two main data types: Information about the games themselves, as well as users who rated these games. We retrieved this data by using the API of the most important board game review site, BoardGameGeek.com. We have collected:

   - 71.343 Board games with a multitude of attributes. Attributes include genre, category, playing time, player count, mechanics used in the game and many more.
   - 201.983 Users who provided a total of more than 20 million ratings. About 40% of users rated more than 100 games each, giving a reliable basis for recommendation computing.

3. Application of Recommender Systems

   The variety of the data allows us to employ different recommendation techniques.

   - Collaborative Methods - recommendations based on users ratings (item-to-item CF, Matrix Factorisation,...) [HKTR04].
   - Content Based Methods - recommendations are computed based on similar attributes from other games (using tf-idf, Self Organizing Maps, Decision Trees...)
   - Hybrid Approach - A combination of the two above [Bur02]

4. Evaluation and Findings

   We measure the performance of our recommendation engine by employing metrics utilizing data on existing users. Evaluation proceeds in following steps:

   a) Use one (or several) of the user's well-rated items as an input.
   b) Calculate a list of games the user likely enjoys as well.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

c) Compare these results to the rest of the user's rated items (not including the withheld items that were used in step 1 as an input). Ratings of the user are normalized to ensure comparability. To achieve this, we employ a range of selected metrics [SG09]:

- Precision is a metric that measures how many games that are relevant to the user have been found in the recommendations.
- Recall is a metric that is defined as the ratio of relevant items selected to total number of relevant items available. In our case, this means: How many of the user's well-liked items is the system able to recall, based on a small subset of input? Evaluation can be based on prediction of the actual rating or on a threshold: If a game is predicted to be "good" (rating higher than 7) or "bad" by the user.
- NDCG (Normalized Discounted Cumulative Gain) is a ranking metric that measures how good the recommendations are compared to the optimal set of recommendations.
- MAP (Mean Average Precision) is another accuracy metric. It measures how good Precision results are at different Recall levels.
- Diversity and Novelty: One shortcoming of recommender systems is that they can recommend items that are too similar to each other. However, there exist metrics to measure this. For deciding if a list of recommended items is too similar or not very diverse, we would compare these to the collection of a user's rated games.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Background

In this chapter, we will look at the current available research. We will collect and present available literature, as well as analyse and compare the most common approaches, their implementations and key findings.

2.1 History of Recommender Systems

A lot of ideas in RS stem from the discipline of Information Retrieval (IR). The term IR was coined in 1950 by Calvin Mooers [Moo50]. The necessity of such systems was proposed around that time due to the concern of the abundance of (scientific) information available and the inability to process all of it. This was closely related to the scientific race between the US and USSR during the cold war [W+63]. The first formal IR research groups were formed in the US. Not very long thereafter, IR systems in all kinds of domains were developed, e.g. the MEDLARS project [Bec64] for medical literature or the SMART system for general scientific literature [SM71]. First online systems appeared in the 70s and improvements in models and evaluation methods continued. With the emergence of the World Wide Web in the 90s, search engines for the web were developed and became the nowadays most famous examples of IR application — which are arguably also RS.

The first literal mention of a "Recommender System" was in a paper by Jussi Karlgren in 1990 [Kar90], where a proximity measure for text documents to retrieve relevant documents for readers was proposed - a classic application of IR methods. In 1992, the term "collaborative filtering" was coined. It described the operating process of the PARC Tapestry system [GNOT92]. "Tapestry" was developed by Goldberg et al. and used both explicit metadata and implicit behavioral data to create personal recommendations in an internal mailing system. Just two years later, Paul Resnick and other researchers at MIT showed with the "GroupLens"-system that recommendations for news on the net can be automated and served over a network [RIS+94]. In 1995, the research efforts

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

targeted multimedia system: Shardanand et al. developed the “Ringo” system at MIT for recommending music albums [SM95]; Hill et al. created a video recommendation system at Bellcore [HSRF95] [RV97].

A recent popularity boost was the announcement of the Netflix Prize in 2006. Netflix would award prizes of up to one million dollar to the individuals who could come up with effective recommendation algorithms for movies. This suddenly produced a lot of interest into RSs. Even if the algorithm by the grand prize winners [Kor09] was not used by Netflix due to engineering challenges [Ama12], the contest produced a lot of valuable insight into RS architecture. As Netflix moved their business model from lending DVDs to streaming and partly due to privacy concerns, they stopped the yearly contest after awarding the prize money. Due to their success, various matrix factorization techniques became the gold standard among the research community.

The increasing popularity of Deep Learning in recent years, most notably the successes of the AlphaGo AI by Deep Mind [Che16], and general application of neural nets in various domains might have pushed the RS community to investigate similar techniques.

Autoencoders are unsupervised artificial neural networks which create efficient encodings from input data. With slight modifications, they have been shown to be applicable as RS. By applying collaborative filtering techniques to Autoencoders, it was possible to beat traditional approaches of recommendations [SMSX15]. Various improvements for this exist by adding slightly noisy input during the training phase [WDZE16]. The most recent developments is the Variational Autoencoder, which learns the parameters of a normal distribution which approximates the distribution of latent variables [LKHJ18].

2.2 Types of Recommender Systems

In this section, we will investigate concrete State-of-the-Art recommendation techniques. We can distinguish different types of RSs. Collaborative (or Social) Recommendation systems rely on the ratings information provided by users. Content Based Recommenders, on the other hand, use descriptive terms of the items. They work on the principle that users will prefer items that have similar characteristics as items they previously enjoyed. Both approaches can be used in connection with Autoencoders, which we discuss later.

2.2.1 Collaborative Filtering

Approaches that take into account user ratings can be split into two general classes: Memory based techniques and model based techniques. In Neighborhood-based approaches, the ratings of items that are saved in the system are directly taken to predict ratings for users. A defined neighborhood determines which ratings are used for a predicted rating. Contrary to using saved user ratings, Model Based Collaborative Filtering aims to build an abstracted representation of the user-item ratings. As a result, these model-based approaches produce a predictive model that is tuned in the training phase according to a set of training data. This model is then used to generate new ratings.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Memory Based Collaborative Filtering

Neighborhood methods are a type of Memory Based Collaborative Filtering technique where ratings that are part of a previously defined neighborhood are considered for rating prediction. The neighborhood can be defined either based on users or based on users:

- **User-based** neighborhood calculates the rating prediction of an item based on the ratings of the most similar users to the user the prediction is calculated for.
- **Item-based** neighborhood calculates the rating prediction of an item based to the most similar items [LSY03] [SKKR01].

A common way to determine similarity is to use Pearson Correlation similarity. In the case of User-based similarity, the similarity of a user u to a user v based on their commonly rated items $i \in I_{uv}$ is shown in equation 2.1.

$$\text{sim}(u,v) = \frac{\sum\limits_{i \in I_{uv}} (r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{u,i} - \bar{r_u})^2} \sqrt{\sum\limits_{i \in I_{uv}} (r_{v,i} - \bar{r_v})^2}} \tag{2.1}$$

$r_{u,i}$ is the rating of an item i by a user u. $\bar{r_u}$ is the average rating of all items rated by user u. As you can see in equation 2.1, the average is always subtracted before every rating. The reason is to eliminate user **rating bias**: Every user has different standards — some users give generally higher ratings. Others are more critical, they have higher standard and their scores are usually lower. If these users give an item a very high rating, it is more significant than when another user, who usually gives high ratings does. This is why we only use the offset - the difference of the rating to the mean [OTTK12].

Equation 2.2 shows the prediction formula. To calculate the prediction of an item i for a user u, we take $\bar{r_u}$ and then add the ratings of i by the other users. However, the ratings of other users is weighted by a similarity factor (2.1). The factor k is one over the sum of total similarities, used for normalization (see 2.3).

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \text{sim}(u,u')(r_{u',i} - \bar{r_{u'}}) \tag{2.2}$$

For efficiency and practicality reasons, not all users are considered for rating prediction. Often, a neighborhood is defined: Only the 5, 10 or 20 most similar users are taken into account.

$$k = 1 / \sum_{u' \in U} |\,\text{sim}(u,u')| \tag{2.3}$$

Figure 2.1 demonstrates the user-based collaborative filtering technique with a neighborhood of 2.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

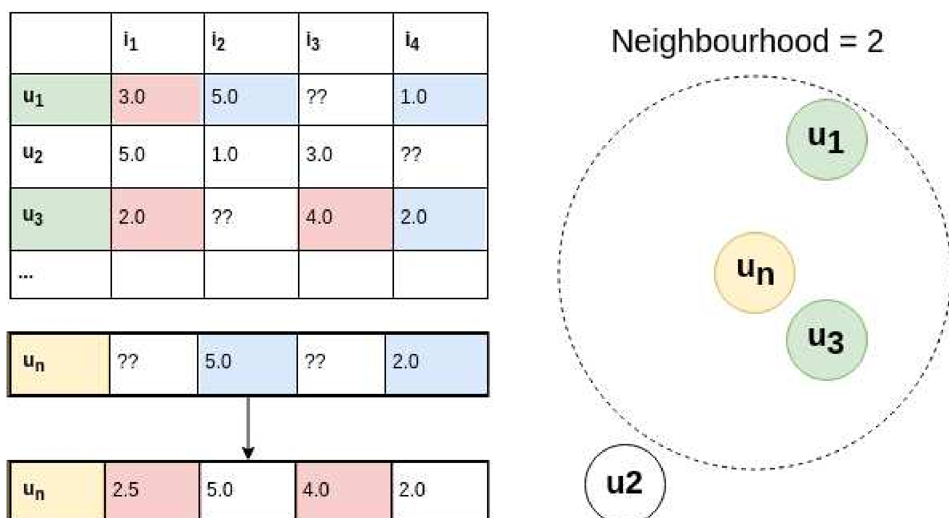


Figure 2.1: We want to predict movie ratings for user u_n (yellow). We see that u_n shares similar ratings on items i_2 and i_4 with u_1 and u_3 (highlighted blue). Consequently, these are the users that are taken for predicting the ratings of items i_1 and i_3 - items that u_n has not rated yet (highlighted red). User u_2's ratings are not very similar to u_n, therefore, u_2 does not appear in the neighborhood of u_n.

Model Based Collaborative Filtering - Matrix Factorization

Matrix Factorization involves building reduced representations of both users and items. It maps users u and items i to a latent factor space of dimensionality k. Each dimension of k represents a latent feature - for example, genre or difficulty or a completely uninterpretable dimension. One entry in the user matrix mxk then defines the correlation off this particular user with this feature (if the user prefers that items include this feature). Similarly, an entry in the item matrix nxk specifies how much this item is correlated with the feature k. Figure 2.2 shows an example of this reduced dimensionality approach [JZFF10] [KBV09].

How to get user and item matrices? Singular Value Decomposition (SVD) is a method to factorize matrices. However, it does not work with sparse matrices like the ratings matrix. To solve this problem, we approach it backwards: Instead of computing the decomposition starting with the ratings matrix R, we generate first the two factors, P and Q, calculate the derived rating matrix R' and then compare to the original ratings matrix. The difference between the original and the newly generated ratings matrix is the error. To make it smaller, we modify the two factors accordingly. This modification is given to us by computing the gradient. Then, this process is repeated. We realize this process by applying Stochastic Gradient Descent (SGD) which we will describe in the following.

The general idea behind SGD is to predict the ratings with our reduced matrices P and Q, calculate the difference to the real ratings (which we call the error) and then adjust

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

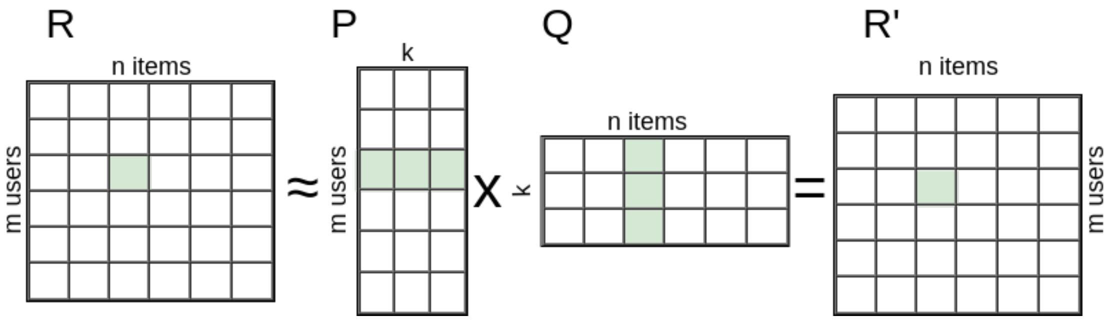


Figure 2.2: R is original user-item ratings matrix. It is sparse, since usually every user only rates a handful of all existing items. The reduced dimensionality k is 3 in this example. P and Q represent the reduced matrices of users and items. These are derived by applying an algorithm like SVD. When P and Q are multiplied, we get a matrix R'. It includes derived ratings for all items by all users.

the values (based on the error) of the matrices in order to approach a better prediction and reduce the error in the next step. This is repeated for many times, as SGD is an iterative procedure. One scan over all the ratings in the training set is called an epoch and it takes multiple epochs to perform SGD. Before the start of each epoch, the order of the ratings is shuffled.

The factors P and Q are initialized with random values. To begin, they are multiplied together to get an initial prediction of ratings. The difference of these to the actual ratings (in the training set) is called the error term. We also need a regularization term, which is added to avoid overfitting (see figure 2.3). The total cost function consists of an error term and a regularization term and is shown in equation 4.4.

To detail how the parameter adjustement works, it is helpful to regard a single rating. The error term for one predicted rating is computed with 2.5. Adding a regularization term, this results in the cost function shown in equation 2.6.

$$Tot.Cost = \frac{1}{|R|} \sum_{r_{ui} \in R} (r_{ui} - p_u^T q_i)^2 + \lambda(||P||^2 + ||Q||^2) \tag{2.4}$$

$$e_{ui} = r_{ui} - p_u^T q_i \tag{2.5}$$

As already mentioned, the matrices P and Q are step-wise modified: Concrete ratings are iterated through and compared to their representation in P and Q. To approach the minimal error, we will be using the inverse of the gradients of the cost function. The process to reach a global minimum in this way is illustrated in Figure 2.4.

The derivatives of equation 2.6 are shown in 2.7 and 2.8, for user and item gradient respectively. We use a similar value to adjust the values in the user matrix P and item

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

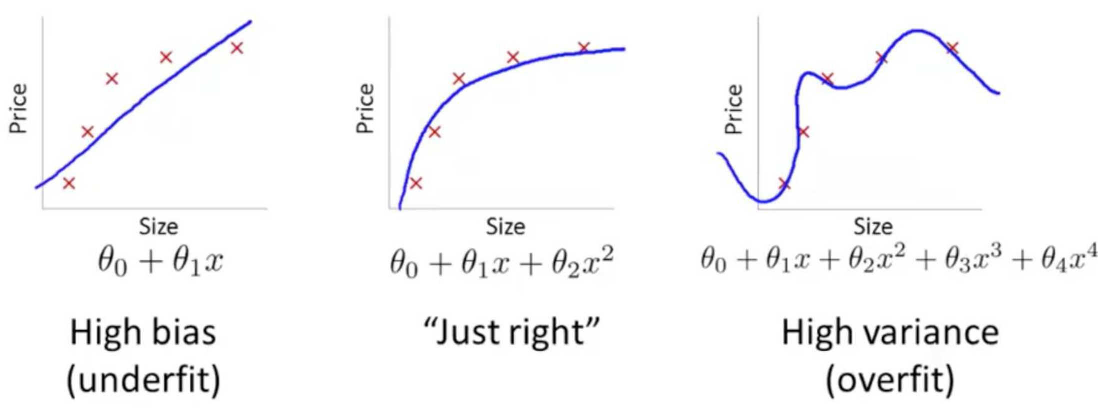


Figure 2.3: When we only use the error term during learning, the problem emerges that the model is tailored too much after the training data and can generalize badly (right image). Introducing a regularization term that depends on the parameter can improve generalization (middle image). The impact of this term can be controlled by a factor and should not be exaggerated, otherwise the model might not be accustomed at all (left image) [Ng08].

matrix Q respectively. The update of the user and item values are shown in equations 2.9 and 2.10. These are the negative gradients. The factor 2 has also been replaced by a variable factor η, which is the learning rate of the algorithm and determines the step size of each iteration.

It is possible to process multiple training examples in one step to reduce time complexity, this is called "Mini-Batch" Gradient Descent. This also reduces variance during training [BCN16].

$$Cost^{(u,i)} = e_{ui}^2 + \lambda(||P||^2 + ||Q||^2) \tag{2.6}$$

$$\frac{\partial Cost^{(u,i)}}{\partial p_u} = -2e_{ui}q_i + 2\lambda p_u \tag{2.7}$$

$$\frac{\partial Cost^{(u,i)}}{\partial q_i} = -2e_{ui}p_u + 2\lambda q_i \tag{2.8}$$

$$p_u \leftarrow p_u + \eta \cdot (e_{ui}q_i + \lambda p_u) \tag{2.9}$$

$$q_i \leftarrow q_i + \eta \cdot (e_{ui}p_u + \lambda q_i) \tag{2.10}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

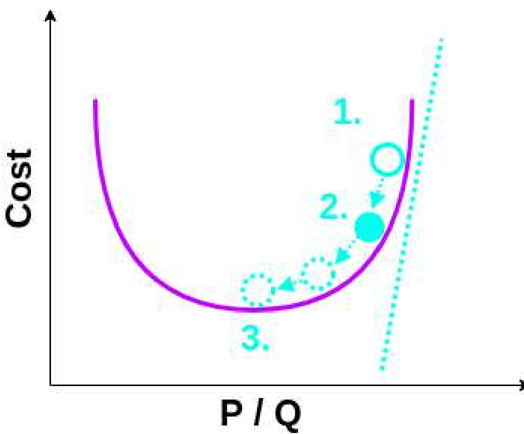


Figure 2.4: When performing SGD, we initialize random parameters for P and Q and calculate the cost (1). The negative gradient is added to the parameters, which decreases the cost (2). After a certain amount of iterations, a minimum value is reached (3).

2.2.2 Content based Recommenders

Content Based Recommender System (CBRS) use descriptions of items to create representations thereof and then suggest items to users based on items they preferred in the past. This happens by storing what attributes a user prefers and then retrieve items that possess these attributes accordingly. These attributes can be explicitely given descriptive terms (metadata) or extracted directly from the items, which can be movies or text articles, by application of various algorithms.

We can represent a CBRS on a high level by abstracting three key modules [RRS15].

- **Content Analyzer** In this step, features are extracted from the item in question to represent in a retrievable manner.
- **Profile Learner** Based on the interactions with items, a profile is constructed that defines the preferences of a user. These interactions can be explicit feedback, like giving user ratings, or implicit based on clicks or views.
- **Filtering Component** This part matches the user representation with existing items. A subset of items is selected to recommend.

These modules are usually highly interdependent. Based on the type of recommender, interactive user feedback is used to optimize recommendations. This concept is shown in figure 2.5.

In the following, we present some types of common CBRS techniques. We will give examples of different Content Analyzers and Profile Learners that have been successfully applied.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

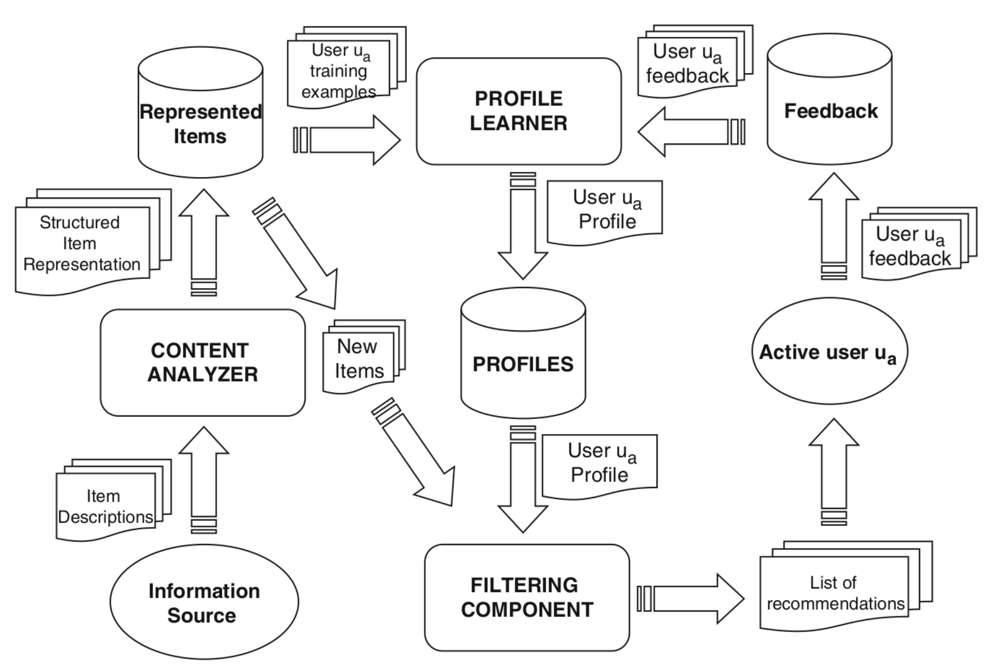


Figure 2.5: High level architecture of a CBRS: Features are extracted from items via a content analyzer. User are represented in user profile. Based on the feedback of an active user, the profile learner adjusts this user's profile. The filtering component matches users to fitting items and composes a list of recommendations [RRS15].

Types of Content Analyzers

These are some techniques that are applicable to text documents, but also to items that can be sufficiently described through text attributes.

These items can be represented in a **Keyword-based Vector Space Model**. This model represents text documents in a vector in n-dimensional space, of which each item is associated to one term in the vocabulary of the overall document collection. Text reduction techniques like stemming and removal of stopwords can reduce the dimensionality of this vector. Each item in the vector corresponds to one weighting value.

A very naïve approach to weight document terms is the **Bag of Words Model**. This generates a feature vector from a text corpus. This vector contains each present word in the document and assigns it the occurrence of this word. Very frequent term are regarded as representative for a document.

Two commonly used weighting schemes are **Term Frequency (TF)** and **Inverse Document Frequency (IDF)**, which stem from IR research and are also applied in web search engines. TF-IDF weighting aims to link each term to a sensible measure of importance: Words that are not that frequent are not less relevant than frequent terms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- in the contrary, if a term that is rare across the whole document collection appears in a document, it is probably quite characteristic for it. This is what IDF describes. Additionally, if a term occurs more frequently in a single document than in others, it is again more descriptive of the document. This is characterized by TF. To combine these two measures, we take the product as shown in Equation 2.11. df_k is the size of the document collection in which the term t_k occurs at least once, N is the total number of documents in the collection and d_j is a document from the collection.

$$TF - IDF(t_k, d_j) = TF(t_k, d_j) \cdot \log \frac{N}{df_k} \tag{2.11}$$

Types of Profile Learners

Machine Learning algorithms that are used in text categorization are applicable for learning user profiles. Building a user profile can be viewed as a task of building a binary classification (either relevant or not relevant), a classification with multiple categories or actually predicting numerical ratings (for example by using regression based methods).

The most common algorithms for learning in CBRS are:

- **Probabilistic Methods**

  This creates a method based on previously observed items. An example is the Naïve Bayes method. Given the probability of observing any document of class c $P(c)$, the probability of observing a document d $P(d)$ and observing a document d given the class c $P(d|c)$, we can calculate $P(c|d)$ using the Bayes Theorem. This is the probability of an observed document d belonging to class c and is shown in equation 2.12.

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{2.12}$$

  After a probability has been calculated for each class c, the highest one is taken as a label for an document d. $P(d)$ is usually uniform, thus removed (each document only occurs once), $P(c)$ and $P(d|c)$ are estimated. To enable better estimation of $P(d|c)$, all occurring words in a document are supposed to be independent of each other given the class and probabilities are estimated word for word, which does not hinder performance [BP97]. Two commonly used models of this approach are th multivariate Bernoulli event model and the multinomial event model [MN+98].

- **Nearest Neighbors**

  Approaches based on Nearest Neighbors store all available training data into memory. There is no actual training step. In the prediction phase, the algorithm compares the new item to all available items and calculates a similarity. Therefore,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

a similarity measure like cosine similarity or other distance functions must be used. The label of a new item is then based to the nearest existing items in the collection.

The algorithm performs generally quite well, despite the simple concept. However, the high computational cost at classification time, which is linearly dependent to item count and the cost of computing the similarity, is a major drawback. Most RSs prefer to optimize the cost at classification time, but allow for high computation cost at training time.

- **Relevance Feedback**

  Relevance Feedback allows for continuous refinement of a search query, in order to optimize a user's search results. This is realized by letting users rate recommended documents, which is used to refine the user profile. For this end, the Rocchio Formula is used and this classification uses also TF-IDF weighting [SM71].

Semantics-Aware CBRS

Semantic techniques can be separated roughly into top-down and bottom-up approaches. The overall intention is to being able to include linguistic intricacies and inherent meaning of words in the systems.

Top-Down Approaches use external knowledge, which can be derived from various information sources to enrich items and represent interests more accurately in the user profiles. Linguistic External knowledge is derived from various sources: **Ontological Resources** like WordNet [Mil95], **Encyclopedic Knowledge** derived from Wikipedia [BRG07] [EGM08] or **Linked Open Data**, an initiative to publish interlinked data sets, which already counts more than 2300 data sets from various backgrounds [Biz09].

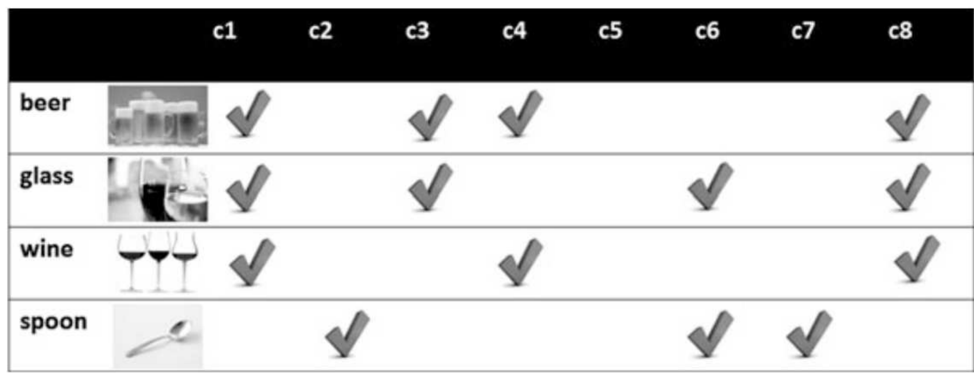
An interesting resource is BabelNet, which combined the encyclopedic knowledge contained in Wikipedia with the linguistic knowledge of WordNet to create a semantic network [NP12].

Bottom-Up Approaches do not rely on predefined constructs of semantics. It derives the meaning of terms directly by analyzing its usage and the context of its occurrences. This results in a semantic representation of items and user profiles. **Discriminative Models** compute term-context matrices, which allows for discovery of related terms, as shown in Figure 2.6. These are also called geometrical models, as they can be represented in vectors and usual similarity measures can be applied [TP10].

As the vector space needed by these techniques can get very big, incorporation of dimensionality reduction techniques are necessary. Latent Semantic Indexing (LSI) is one approach which is based on SVD [DDF+90] and decomposes the term-context matrix into matrices of reduced dimensionalities. Some applications of LSI in CBRS are an algorithm for recommending TV-shows [BCT11] and source code examples [MCK06].

Another more scalable reduction approach is based on Random Projections (RP) [Vem05], where high-dimensional vector spaces can be randomly projected into a lower-dimensionality

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub



	c1	c2	c3	c4	c5	c6	c7	c8
beer	✓		✓	✓				✓
glass	✓		✓			✓		✓
wine	✓			✓				✓
spoon		✓				✓	✓	

Figure 2.6: Due to the co-occurrences of the terms *glass* and *wine* with the term *beer* in documents c1, c3, c4 and c8; the term-context matrix shows them as being related to each other [RRS15].

spaces, without losing distance information. The discriminative model built on RP is called Random Indexing [Sah05] and has been used in a multimedia RS for TED talks [PPB15], a knowledge recommender [SLBdG09] and as an enhancement of collaborative filtering [SCJ11].

2.2.3 Autoencoders

In the past few years, there have been breakthroughs in the application of deep learning algorithms in both academia and industry, across various domains [CAS16]. Consequently, there have been similar advances in applying deep learning in recommender architectures. Deep learning can reveal here non-trivial features and hidden relationships between data items [ZYS17]. One of these deep learning techniques we shall focus on is the autoencoder.

The autoencoder is an approach to **representation learning**. This allows for discovering representative features of the samples in a data set. In this way, it is similar to other mentioned techniques like SVD for matrix factorization.

An autoencoder is a neural network that is designed to recreate the input it receives using back-propagation. If the number of hidden nodes is smaller than the number of nodes in the input and output layer, we speak of an undercomplete autoencoder. In the case of the number of hidden nodes being equal or bigger than the input nodes, the autoencoder is called overcomplete. As we are interested in compact feature representation, we are only discussing undercomplete autoencoders [GBCB16].

An autoencoder consists of two parts, an **encoder** and a **decoder**. The encoder function compresses the input data set into an intermediary representation, writing information into the hidden node layer. The decoder transforms it back into the original input format.

TU Bibliothek WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

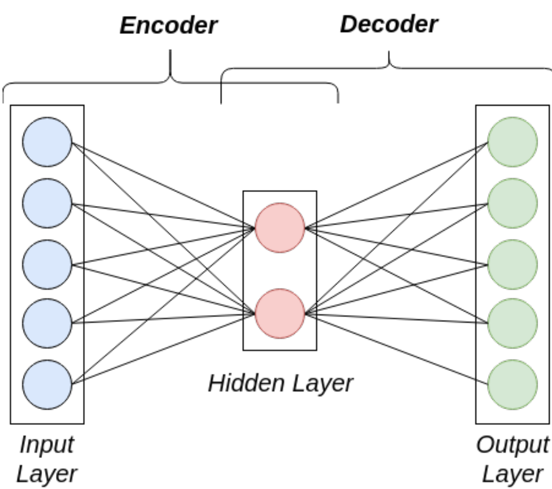


Figure 2.7: Example of an autoencoder. The encoder part converts input data (blue circles) to a more compact representation (red circles). The decoder processes it back into the original representation (green circles)

A simple autoencoder consists of only one hidden layer, as shown in figure 2.7. Stacked autoencoders with multiple hidden layers are used for complex input data and are trained layer by layer [GP17]. The design of an autoencoder is very flexible: The amount of hidden layers, number of hidden nodes and various activation functions can be used for the encoding or decoding process. When using an activation function like ReLu or Sigmoid, it is necessary to scale the input data into a usable range.

Autoencoders are by design very flexible. While they have been mostly used for application of collaborative filtering in regard to using recommendations, it is possible to apply them to any kind of data. This leaves room to explore approaches like content-based recommendations.

Collaborative Autoencoder

Autorec was the first application of collaborative filtering using an autoencoder [SMSX15]. It was implemented in both an item-based and a user-based model. In evaluation, item-based Autorec showed the lowest RMSE compared to collaborative filtering using restricted boltzmann machines (another deep learning method). Additionally to effectiveness, collaborative autoencoder also beats other approaches in efficiency: The memory footprint for training as well as for testing scales linearly, as only items are needed to be represented in latent space (compared to factorization where items and users both are transformed into latent matrices). These results apply to autoencoders with one layer, although the authors hint at possible performance gains by experimenting with multiple hidden layers.

Autorec, as any autoencoder, aims at minimising the difference between input and output layer during its training phase. This is shown in equation 2.13, which describes

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

the training function of minimising the RMSE of all items of S, the data set used for training, and $h(r;\theta)$, which represents the output of the autoencoder, i.e. the decoded latent representation of r. The output is the end result of the process of encoding and decoding, which is shown in equation 2.14. In this case, $\theta = W, V, \mu, b$, where W and V depict transformations for decoding and encoding; μ, b are added biases after each transformation.

$$\min_{\theta} \sum_{r \in S} ||r - h(r;\theta)||^2 \tag{2.13}$$

$$h(r;\theta) = f(W \cdot g(Vr + \mu) + b) \tag{2.14}$$

Denoising Autoencoder with Collaborative Filtering

Whereas the classic autoencoder aims to reconstruct provided inputs during training, the denoising autoencoder modifies the actually used data that is feeded into it. It corrupts a fraction of an input $x \in [0,1]^d$, creating a modified input vector $\tilde{x}$. This added noise can either be reached by randomly masking out data entries (e.g. with binary data) or adding Gaussian noise (when using continuous data). This aims to mitigate one shortcoming of autoencoders, where the hidden layer just learns the identity function to replicate the input and forces the neural net to discover more consistent features [VLBM08].

In order to apply collaborative filtering with a denoising autoencoder, it has been proposed to include an additional node for the user whom the ratings belong to. This means that there is a separate weight vector V_u associated with the user input node, which is unique for each user, encoding the user's preferences [WDZE16]. Similarly, a bias node is added to the hidden layer for each user as a regularization parameter to avoid overfitting as shown in figure 2.3. This concept is visualized in figure 2.8.

Variational Autoencoder

The Variational Autoencoder does not just learn an encoding function to transform the input into a latent representation. It creates a latent variable model from the input data and learns its parameters during training. These parameters describe a probability distribution of a statistical model that is derived from the observed data. The latent vectors that are generated follow this normal distribution [RMW14]. The concept is shown in figure 2.9.

It is very interesting to note that the variational autoencoder can be used as a generative model. By supplying arbitrary values for mean and variance to the decoder, we can generate new instances in the output layer [KW13].

In order to optimize results from collaborative filtering with a variational autoencoder, it has been proposed to use a multinomial distribution to describe the data (as opposed to gaussian distribution like in figure 2.9). These multinomial likelihoods have been

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

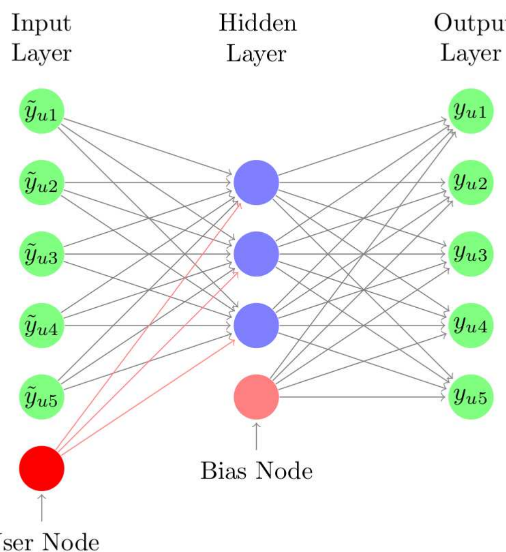


Figure 2.8: A depiction of a Collaborative Denoising Autoencoder. The red links are the user specific weights V_u. The other links are shared and user independent [WDZE16].

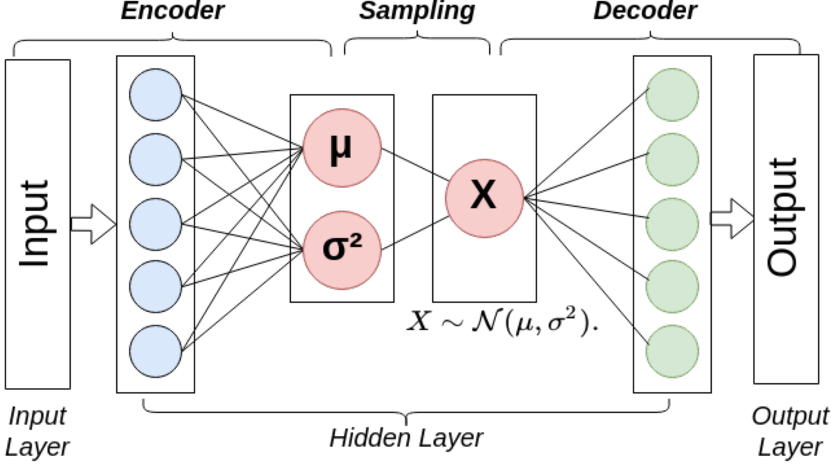


Figure 2.9: Like a normal Autoencoder, the Variational Autoencoder possesses an arbitrary number of hidden layers. However, at the core, there is a layer that transforms the inputs to parameters of a normal distribution, then transforms the sampled distribution back through one or multiple dense neural layers.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

used for modeling implicit feedback data in a more accurate way, for example mouse clicks by users [LKHJ18]. Additionally, there are also adjustements with regard to the regularization parameters.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

History of Board Games & Data

In this chapter, we reflect on the subject of Board Games. We highlight briefly their history, discuss their characteristics and we explain in-depth the technical challenges of building an extensive collection of data on top of which to build a recommender system.

The term 'board game' generally refers to all sorts of tabletop games, card games, miniature games. In essence, a game is a system where participants strive to reach goals, acting within a certain framework of rules.

3.1 History of Boardgames: Then to Now

Board games are not a modern pastime at all — they have been around for more than 5000 years, their creation coinciding with the cultural heights of the first advanced civilizations. The earliest found artifact is the game of *"Senet"*, found in a tomb of Predynastic Egypt dating back to 3500 B.C.E. [CDVdV16]. Other ancient examples across the globe include: *"Backgammon"* from Persia, *"Go"* from China, *"Mancala"* across Eastern Africa, "Chess" and *"Pachisi"* from India and "The Royal Game of *Ur*" from Mesopotamia. They all have in common that they were played on a board that indicated a grid-like playing field, included tokens to mark the current game state and were enjoyed by people of all social strata - commoners and noblemen alike [Mur52] [F+07].

These games spread around the world and transformed during the centuries. Chess evolved into its current form in Europe in the 15th century. Backgammon spread widely, originating from Persia and Mesopotamia: To Greece and the Roman Empire in 500 C.E. ; to the Ottoman Empire, Western Europe and East Asia in the 11th Century. A similar game to Senet and The Royal Game of Ur emerged in the 10th century under the name "*Quirkat*" in the Middle East and was later brought to Spain and France, where it was played with backgammon pieces on chess boards. Therefore, it was named after the queen piece in chess: "Dames". Today, it is known as "Checkers" or "Draughts" [Par99].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Figure 3.1: Examples of historic board games. From left to right: *Senet* with game pieces found in the tomb of Tutankhamun (1332 B.C.E.), from [Wik18b]; *The Game Of The Goose* (1880s), from [Wik18a]; Gaming pieces of the *Kriegsspiel* by Lieutenant Georg Leopold, Baron of Reiswitz (1812) from [Hil00]. Modular game pieces like the ones depicted are a mechanic featured in lots of contemporary games.

This evolutionary process that occurred over centuries is still very evident in modern games today. To innovate, mechanics and components from different games are mixed and refined: Some combinations that work well are integrated into new games, others are abandoned.

From the late 17th century onwards, so-called *"Kriegsspiel"* games were used in german military circles to demonstrate tactical maneuvers and political principles. During the following 200 years, rules were increasingly adapted to fulfill the needs of sophisticated simulation of combat. In the 20th century, these games evolved to "War Games", featuring meticulous rule sets to realistically simulate historical battles. They still enjoy a niche following by history enthusiasts. [Pet12]

"The Game of the Goose" originated in the 16th century in Europe, which established the genre of racing games. These became very popular in the late 18th century - not primarily as a fun pastime, but rather as a tool for teaching morals and behavior for children. Spaces on the board featured vices to avoid (e.g. the pub, or, ironically, gambling) and noble deeds to collect (e.g. going to church or school). As industry advanced and the middle class grew stronger, these kind of board games featured in numerous households [Car65].

Milton Bradley created the "Game Of Life" (still featuring virtues and vices) in 1860 and launched the commercialization and wide adaption of board games. Antique games like *"Pachisi"* and *"Gyan Chauper"* of Indian origin were transformed to the more commercially friendly games of "Ludo" and "Snakes and Ladders", which became a success.

Starting in the 1880s, games that emulated the famous rags-to-riches story became very popular in the United States. The most notable example is "Monopoly", published in 1935 by "The Parker Brothers". Ironically, it was designed originally by Elizabeth Magie

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

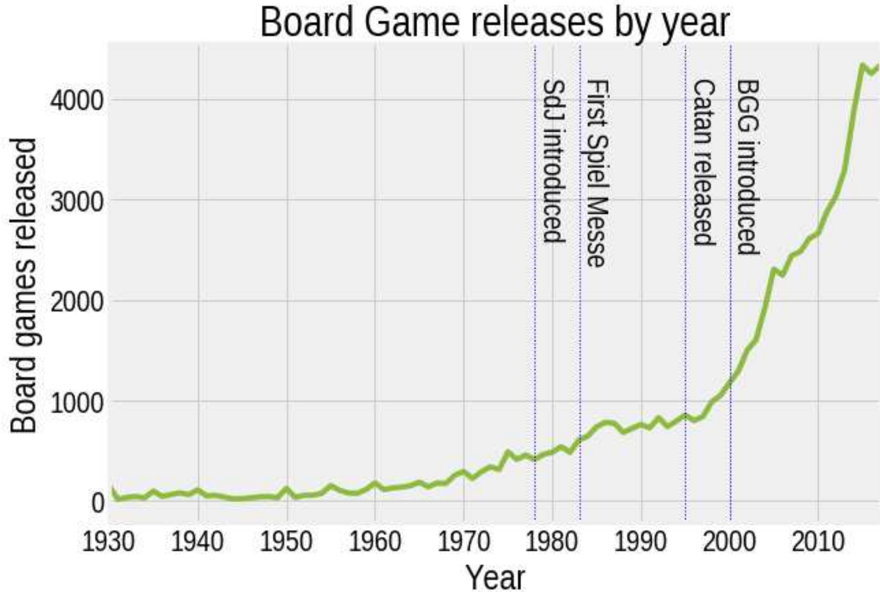


Figure 3.2: New releases of Board Games per year since 1930 with blue lines indicating certain industry-shaping events.

as a critique of unregulated capitalism under the name “The Landlord’s Game”. Not long thereafter, the games of “Risk” and “Clue” were released and enjoy huge commercial success to this day. For decades, the board game market was characterized by a few different games that made up the majority of sales. What all these games have in common is the heavy use of dice and thus the big factor of luck involved in winning.

In 1960, the “3M” company released “The Bookshelf Game Series”, which included classics like Chess and Go, but also new designs, most notably “Acquire”. These strategic games were targeted to adults. Acquire became very popular in Germany, and the term “Eurogame” (also known as “German-style board game”) was coined [Woo12]. These types of games rely less on luck but more on planning and strategical thinking. Starting in the 70s, a lot of games in this vein were created and sold in Germany. Another common theme of these games is indirect competition. Players compete for resources, but have no means of directly inflicting damage on each other. This proved to be popular due to the tendency of conflict aversion in education of post-war Germany: Parents didn’t want their children to play games featuring combat, war and fighting. Most of modern board games released today have their roots in these classic “Eurogames”. The most famous and successful example of this category is “The Settlers of Catan” from 1995. It is attributed with popularizing the genre across the world, selling more than 22 million copies in more than 30 languages.

Meanwhile, the board game scene in the United States focused more on vivid storytelling.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

"War Games" started to feature in Fantasy or Sci-Fi settings and would increasingly add elements of chance. They evolved to "Role Playing Games" like "Dungeons and Dragons" and "Collectible Card Games" like "Magic The Gathering" [Pet12]. It is increasingly common to find games that modeled after famous book series or movies [Boo15]. Figure 3.2 depicts the meteoric rise of board game during the last decades. The number of releases rises exponentially.

The increase from the 1970s onwards is heavily attributed with the popularity in Germany. The blue lines indicate certain events: Introduction of the "Spiel des Jahres" award (1978), known as the "Oscar of Board Games"; the inauguration of the first dedicated trade fair, the "Spiel Messe" in Essen, Germany (1983); the release of the game "Catan" (1994); and launch of the site boardgamegeek.com (2000). All these events are attributed to increasing the popularity of board games. As a source of data, the website boardgamegeek.com is very important for our purposes. We discuss it in the following section.

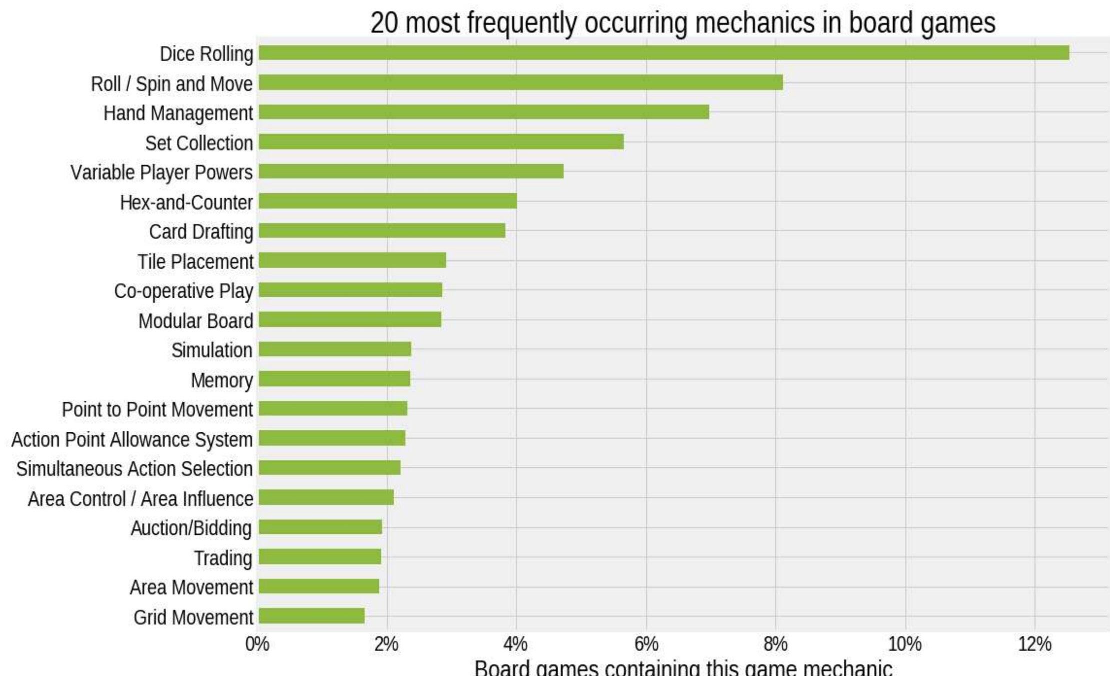
Nowadays, there are plentiful of examples from each type of game mentioned until now. A lot of games involve dice in some way, but lots of other mechanics have also emerged. Sometimes it is hard to draw strict lines between genres. Board Games seem to win popularity everywhere, and new Game Designers from countries like Japan and Taiwan try their luck in the board game industry. An overview of the most popular mechanics and categories is shown in Figure 4.3.

Due to the vast, but finite number of different game states, board games - especially Chess with its $\sim 10^{46}$ different combinations [Chi96] - were a good application domain for Computers. In fact, Alan Turing designed one of the first heuristics for chess playing in 1948 and tested it some years later [Tur53], as did Claude Shannon in 1950 [Sha88]. Performance in the game of Chess, and later "*Shōgi*" and Go (having an even bigger number of valid plays) was used as a measuring tool for the advances of artificial intelligence. When the algorithms of IBM's Deep Blue beat Chess Grandmaster Garry Kasparov in 1997 (see Figure 3.4) or Google's neural net AlphaZero beat Go world champion Lee Sedol in 2016, it made waves in the media.

3.2 The Data - boardgamegeek.com

The most popular existing website on board games is BoardGameGeek.com (BGG) [DSa]. Since its inception in 2001, more than 250.000 users have rated over 84.000 games and provided a wealth of information about these games. Ratings can be given in the inclusive range of 0 to 10 (= worst to best) points. The site has 1.5 million registered users (the majority of users doesn't rate games) and 4 million unique monthly visitors. It is a very active community where users also contribute in forums, share news and upload images. The characteristics of each game are presented in a very condensed format, as you can see in Figure 3.5. Ratings and a wealth of other information like complexity and playing time is clearly visible, which we collect and use in our recommendation algorithms.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub



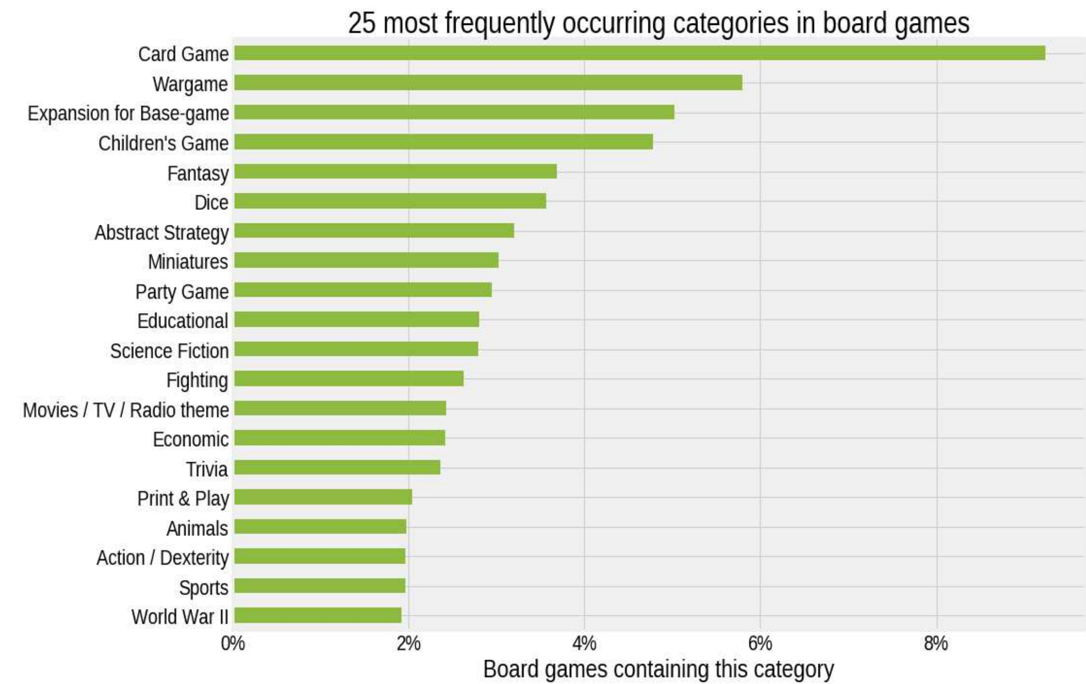


Figure 3.3: Top: Histogram of the most common mechanics in board games. Bottom: Histogram of the most common categories.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Figure 3.4: Chess enthusiasts watch Garry Kasparov (lower right) play his final game against the Deep Blue Computer, New York, 11 May 1997, from [Sta18]

3.3 Collecting the Data & Data Models

To get the data from the site into an easy-to-use format for our purposes, we need to collect it in some way. BGG does not provide downloadable snapshots of their data, so we need to collect it ourselves. The site, however, provides an Application Programming Interface (API) [McG], over which it is possible to extract the information visible in Figure 3.5. The API allows for access on the games as well as the users.

In total, we have collected information on 80.474 games. They accumulate over 13 million ratings in total, from about 249.186 different users. About 40% of users have rated 250 games or more, as can be seen in Figure 3.6. An interesting observation is that the ratings per user are not given evenly distributed. About 50% of ratings per user are given in the range of 6 - 8, as shown in Figure 3.7. Another peculiarity is that the average rating of games increases in average over the years, implying released games get better with time (see Figure 3.9). This might be linked to growing competition and better understanding of customer needs.

We implemented the scraper using best practices and Python's standard libraries described in [Mit15]. We extended our web scraping efforts over many hours in order to not bombard the BGG servers with too many requests (and to avoid the possibility to get blacklisted). We iterated over all the existing games and users, saving their respective attributes and ratings. A detailed diagram of the data retrieval processes involved is shown in Figure 3.8. The usage of this API is unrestricted for non-commercial purposes as specified in the Terms of Use [Boa18] and would additionally fall under the case of fair use [Ger06].

This careful approach has one side effect: By spreading the collection over a long timeframe, the data we collected in the beginning of the process is not as recent as the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

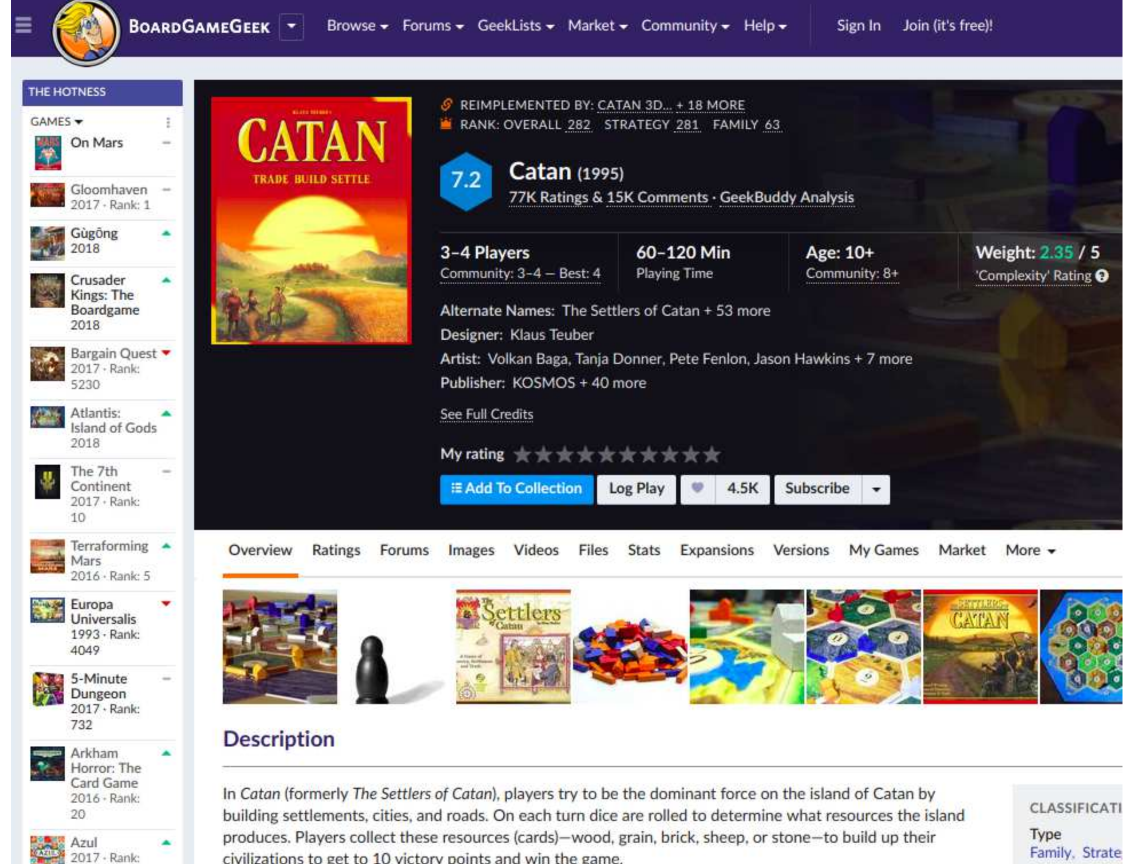


Figure 3.5: A screenshot of the BGG web site with an example game selected. Users can easily see average rating, rankings, categories, complexity and more details on this game (site accessed October 2018) [DSb].

data collected later. They are snapshots off different parts of the database at different points of time. However, we are sure that this does not invalidate any collected data and their ratings. The variation should have no effect on our recommendation techniques.

The collected information is stored in databases by using a normalized data model described in Figure 3.10. It is designed in order to avoid redundancies and duplicate data. It features a multitude of many-to-many relations expressed through IDs. In order to use the data for certain applications (especially content-based recommenders), appropriate joins of needed tables are necessary in a pre-processing step.

3.4 Available board game recommendation systems

The only recommender system of board games we are aware of is boardgamefinder.net, which operates on using a combination of Exponential Family Embedding (EF-EMB) in

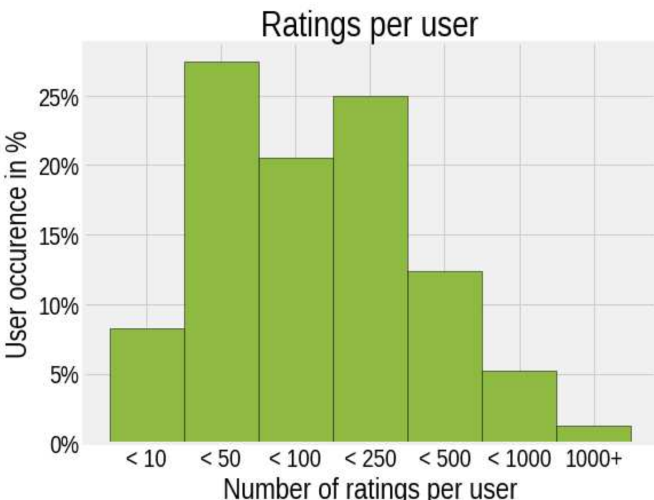


Figure 3.6: Number of total ratings per user in the dataset. More than 85% of users rated more than 50 games, giving a healthy data set for recommendation purposes.

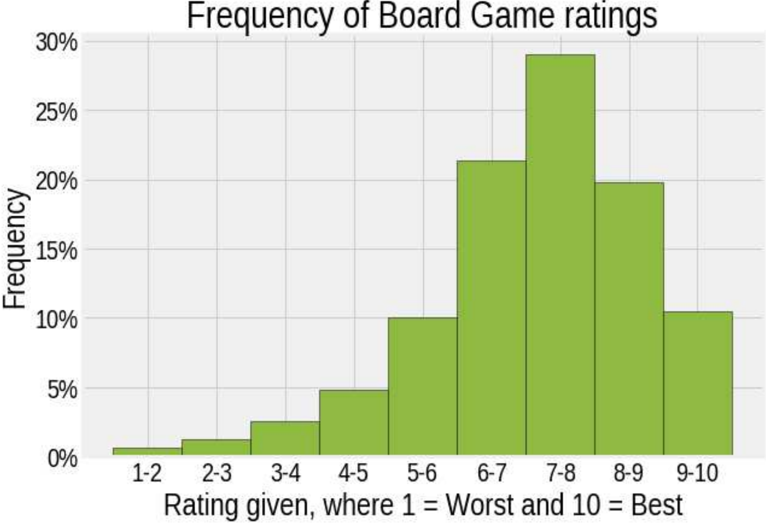


Figure 3.7: Histogram of game ratings per user. It is more likely that users give ratings ranging from 6 to 9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

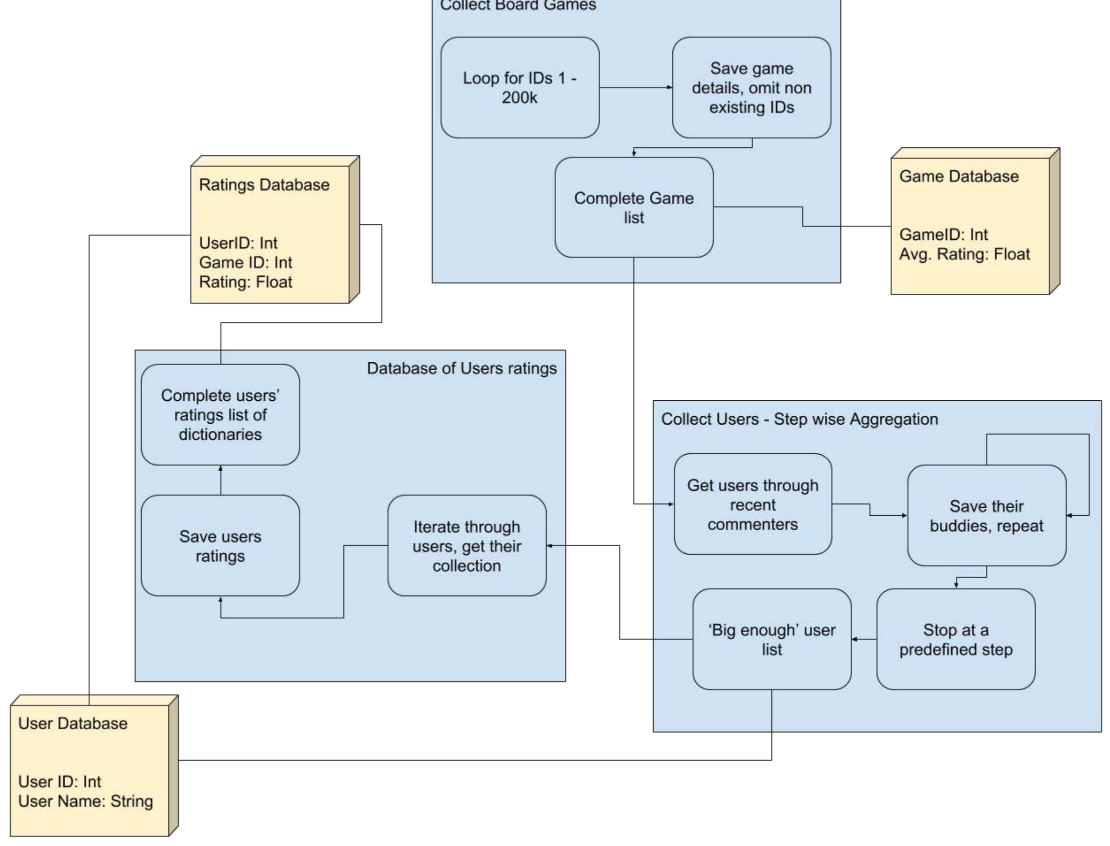


Figure 3.8: Board game collection process view. The API does not allow to iterate directly over users IDs, but game IDs. This is why we start with iterating over the games: The top process "Collect Board Games" loops over all existing board game IDs and extracts descriptive attributes. At each game, we also loop over all associated comments. This is how we collect user information and is shown in the blue box "Collect Users - Step wise Aggregation" on the lower right. The user ID is associated with the comments, which is why we can iterate now over the ratings and comments from each user by sending requests with user IDs. The data is saved in a separate database.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

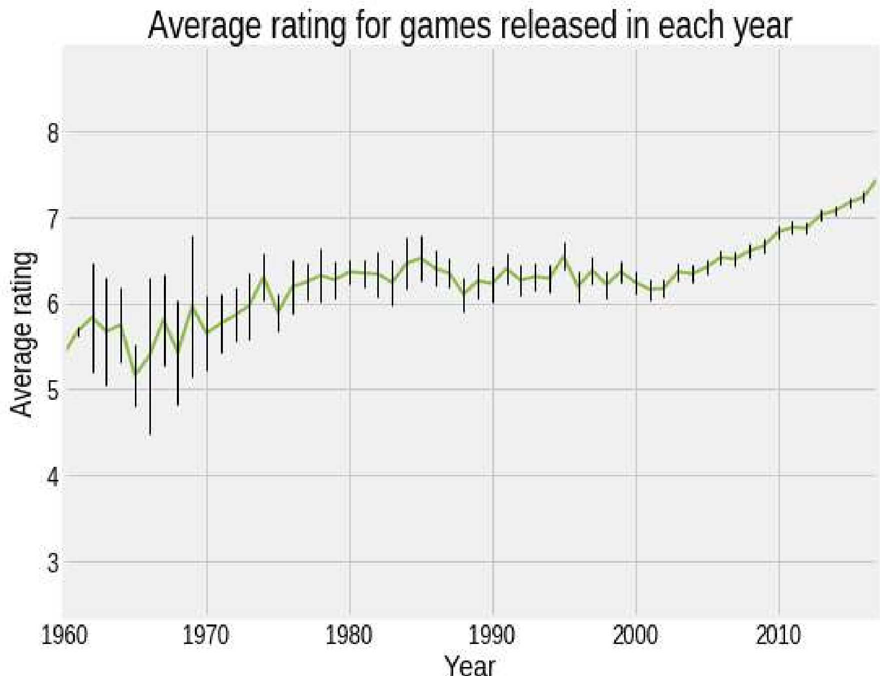


Figure 3.9: Average ratings of games in the years 1960 - 2018 with standard error bars.

combination with matrix factorisation in their recommendation approach as stated on their website [R+16]. However, they do not provide a paper containing details on their implementation or any evaluation. On their website, they announce a paper with results to "be coming soon". However, we present now the concept of EF-EMB for purposes of completeness.

EF-EMB are a form of word embeddings, which aim to model similarity of terms in a vocabulary [RRMB16]. The principle of word embeddings stems from Natural Language Processing, but EF-EMB can be used for any form of multidimensional data, so they are applicable to many domains.

To create EF-EMBs, you need three parts: A context function, which decides what items are relevant to a given data point. An example would be items on an e-Commerce site. Items in the same shopping basket are in the context of each other. For boardgamefinder.net, the context is created from games which were rated by the same people.

Then, you need a conditional distribution which takes a function of the embeddings of both the items and its context items as a parameter. The third part is the embedding structure, which finds embeddings and context vectors to describe the given data point. The model trains the latent vectors for both embeddings and context for each example. The interesting aspect of this recommendation approach is that it considers both complement items as well as exchangeable items. Complements are found through the context vectors,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

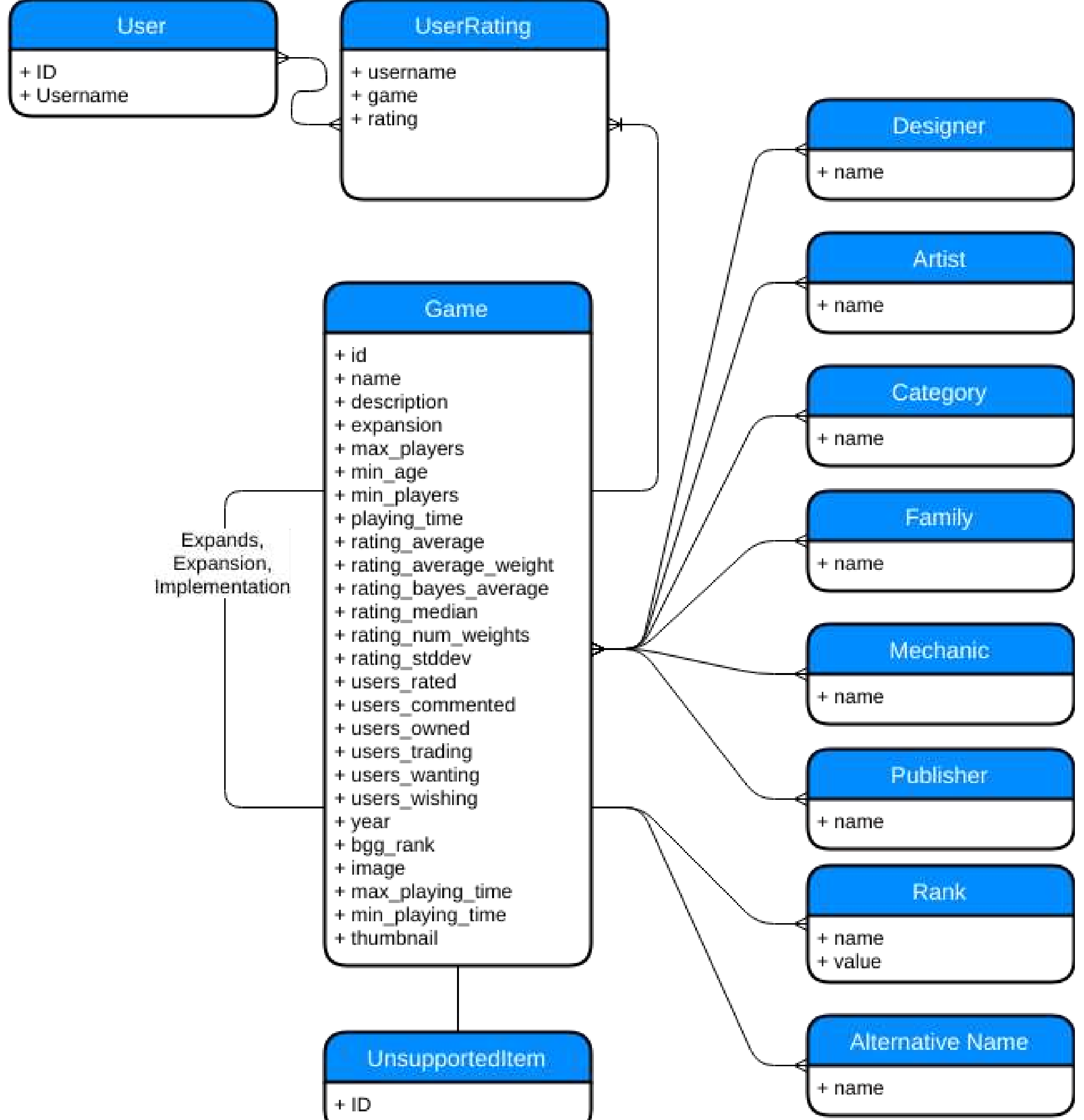


Figure 3.10: The data model of the collected board games and users. Users are associated to their respective ratings. A game contains all critical information and has many-to-many relations to other tables containing information to categories or mechanics in order to save storage space.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

which return the most probable items that appear in the context of an item. Exchangable items are found thanks to their similar context functions.

We investigate several recommendation methods, including collaborative methods, and then compare in an exhaustive evaluation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 4

Recommenders

In this chapter, we investigate specific details about our implementation of the discussed recommendation techniques and approaches taken to fit our use case.

4.1 Implementation

Most parts of this project are written in the Python programming language [VRDJ95]. There are numerous available modules for data intensive applications. These include modules for data cleansing, processing and performing machine learning or other tasks related to Data Science. The availability of these modules, combined with the relative ease of learning the Python programming syntax made the language very popular among professionals with a non-computer science background. It is now among the most commonly used programming languages worldwide, and the most popular one taught in introductory university programming courses [Guo14].

The resulting artifacts of this thesis are delivered in standard Python (.py) files and "Jupyter Notebook" (.ipynb) files, which is a format for interactive scientific computing in Python [PG07].

As mentioned in the previous section, the web scraping was done by using the xml parser class contained in the Python standard modules. The contained xml structure made it easy to encapsulate the scraped data in object instances that translated easily into database models. The data was collected during the time frame of several weeks, as the requests to BGG were not too excessive in order to avoid being blacklisted by server protection mechanisms.

Apart from the various standard modules contained in Python, which we mention in the following.

4.1.1 Data Management

For storing the collected data we use the relational database management system SQLite. Advantages for this choice include that it doesn't have to be run as a service, databases can be saved as simple text-based files and it is straightforward to export it to other formats. Setup and integration is fairly easy as it is a weakly typed database engine and offers interfaces for a variety programming languages. Our data model fulfills the requirements of the third normal form.

4.1.2 Data Processing & Visualization

We cannot rely on standard data structures like arrays and lists, we use specific modules in order to manipulate and transform the data more easily. These are **Scipy** [JOP+], **Numpy** [Oli06] and **Pandas** [McK10]. Scipy is a module for scientific computing, for example, it includes functions for calculating distance measures. Numpy makes it easier to handle and manipulate vectors and matrices. Pandas extends on this by including dataframe objects which are constructs containing different types of data, similar to tables in a database. It also includes transformations similar to SQL for adapting data in-memory.

For visualizing purposes in this thesis, we utilize the **Matplotlib** module [Hun07].

4.1.3 Recommendation Computation

We engineered our recommender systems in most cases with standard modules and already mentioned data processing focused modules like pandas and numpy.

For constructing the Autoencoder, we utilize the **Keras** library to build appropriate neural net models [C+15]. Keras consists of mappings to lower level modules like Tensor Flow and facilitates building machine learning models, so we do not need to worry about the concrete, low-level implementation of our neural nodes and their activation functions.

For creating the kNN classifier, we utilize similarity measures contained in the **scikit-learn** module [PVG+11].

4.1.4 Hyperparameter Tuning

Our RS have attributes which can be configured and tuned. In order to find the configuration with the best performance, we apply Hyperparameter tuning on a subset of the data (about 500.000 ratings).

The best practice is to create applicable value ranges for each parameter and then run a certain number of training sessions with evaluation. For each run, the values for parameters are picked at random from these ranges ("grid search"). The best performing configuration is then the basis for the full evaluation with the complete dataset.

In the following sections, we highlight some details about our RS implementations, while not repeating the general information already mentioned in chapter 2. In a later chapter,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

we describe the evaluation results. We also note which values were used and tested during hyperparameter tuning.

4.2 Collaborative Filtering Approaches

The approaches discussed in the following section have one thing in common: The similarity estimation for making recommendations is based solely on user ratings.

Due to this, the techniques vary in the way they process these ratings, and as a result, they all differ in complexity, computation time and effectiveness. They stem from collaborative approaches discussed in the previous chapter on the State of the Art.

For all their conceptual differences, there are certain parallels between the RSs. In order to compare our RSs to each other in the evaluation step, they have a common input and output protocol: Input is a list of Game IDs that a user enjoyed, which are games that the user rated with a 7.0 or higher. The output is a list of recommended game IDs. Another commonality is the use of regularization parameters.

For the training and test set, we split up each users ratings as described in section 6.3. They consist of user ratings equal or above 7.0.

4.2.1 Memory Based Collaborative Filtering

User - User based

Our user based collaborative filtering approach was adapted to fit with the requirement of input of game IDs.

For all ratings $\geqslant 7.0$, a similarity matrix of all users to each other is computed. It is calculated using mean-adjusted cosine similarity. The similarity of a user u to a user v based on their commonly rated items $i \in I_{uv}$ is shown in equation 4.1.

$$\text{sim}(u,v) = \frac{\sum\limits_{i \in I_{uv}} (r_{u,i} - \bar{r_u})(r_{v,i} - \bar{r_v})}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{u,i} - \bar{r_u})^2} \sqrt{\sum\limits_{i \in I_{uv}} (r_{v,i} - \bar{r_v})^2}} \tag{4.1}$$

For each game in a given input of games, a neighborhood of users which provided the highest rating to these games are found. We experimented with different sizes of neighborhoods, i.e. how many other users' ratings are considered. We tried the sizes 1, 2, 5 and 10 in hyperparameter testing. We have found that a neighborhood of 5 users gave the best result.

All these ratings are merged together and weighted. The weighting depends on the rating the user gave to the game used in the input. Then, a weighted average of the ratings of the users (if one game appears in several users' ratings) in the neighborhood is given back as a recommendation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Item - Item based

For the item based collaborative filtering, it was enough to calculate the cosine similarity between the ratings of all games. The recommendations are the games that match the rating patterns of a game in the input the most. If multiple games are given as an input, it can happen that the same game is returned with different similarities in the output - in that case, the average is calculated.

Similar to the User - User approach, we consider all ratings $\geqslant 7.0$ and calculate a similarity matrix of all games to each other. We use here mean-adjusted cosine similarity as well. The similarity of a game i to a game j is based on their rating score by the same user $u \in U_{ij}$, which is shown in equation 4.2.

$$\text{sim}(i,j) = \frac{\sum\limits_{u \in U_{ij}} (r_{i,u} - \bar{r_i})(r_{j,u} - \bar{r_j})}{\sqrt{\sum\limits_{u \in U_{ij}} (r_{i,u} - \bar{r_i})^2} \sqrt{\sum\limits_{u \in U_{ij}} (r_{j,u} - \bar{r_j})^2}} \tag{4.2}$$

4.2.2 Model Based Collaborative Filtering - Matrix Factorisation

As mentioned in section 2.2.1, we generate latent matrices P for users and Q for items from ratings matrix R. The multiplication of P and Q results in generated item predictions in the matrix R'. An example for the predicted rating of item i for user u is shown in equation 4.3. p_u is the column for user u in P and q_i is the column i in item matrix Q.

$$r'_{ui} = p_u^T q_i \tag{4.3}$$

These matrices are initially random and step wise adjusted to approach values that approach the existing ratings in the training set the best. To determine how much adjustement is needed, we need a cost function as shown in 4.4.

$$Tot.Cost = \frac{1}{|R|} \sum_{r_{ui} \in R} (r_{ui} - p_u^T q_i)^2 + \lambda(||P||^2 + ||Q||^2) \tag{4.4}$$

The appropriate user columns, respectively, item column is updated by adding the error gradient after each step. This is shown in equations 4.5 and 4.6 respectively. The error gradient is derived from the cost function.

$$p_u \leftarrow p_u + \eta \cdot (e_{ui} q_i + \lambda p_u) \tag{4.5}$$

$$q_i \leftarrow q_i + \eta \cdot (e_{ui} p_u + \lambda q_i) \tag{4.6}$$

Our Matrix Factorisation approach provides five modifiable parameters which were adjusted via Hyperparameter Tuning.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

The **Encoding Dimension** specifies the dimensions of our factorised matrices and was set to 5. The values tested during hyperparameter tuning were 1, 2, 5, 7 and 10.

The **Epochs** specifies how many iterations we need to adjust the factorised matrices and was set to 20. The tested values were 10, 20 and 50.

There is a **Regularization** parameter to avoid overfitting, set at 1e-4. Values tested were 1e-2, 1e-3, 1e-4 and 1e-5,

The **Learning Rate** determines the step size during SGD and is set at 1e-4. Additionally, we tested 1e-3, 1e-2 and 1e-1, which returned worse values.

4.2.3 Basic Autoencoder

For all the Autoencoders, the input and expected output are a vector of the length of all existing games. This means during training, the input consists of the ratings of all items a user has rated (without the ratings that are reserved for the evaluation set as explained in section 6.3).

Among all Autoencoders, we have a set of common parameters: The number of **Hidden Nodes** in the intermediate layer, the **Epochs** i.e. the iterations for training the hidden layer, the **Batch Size** determines the amount of throughput during each iteration.

For the basic autoencoder, we tested 512, 1024, 2048 and 4069 hidden nodes; 10, 20, 50 and 100 epochs; batch sizes of 32, 64 and 128; regularization of 0.01, 0.001 and 0.0001.

After tuning, we found that best results are given by using 1024 hidden nodes, 50 training epochs processing batch sizes of 128. The Regularization term is 0.01.

During training, each node learns a weight value W to apply to an input x. It adds a bias b and then feeds the result to an activation function σ. The process is shown in equation 4.7.

$$z = \sigma(Wx + b) \tag{4.7}$$

We experimented with both *ReLu* and *Sigmoid* activation functions and *adadelta* and *adam* optimizer functions. The activation functions are used for encoding to the neural layer. It determines the output range of every node. The range of *ReLu* is between -1 and 1 and the range of *Sigmoid* is from 0 to 1. The functions are shown in figure 4.1. For the sigmoid input function, we needed to normalize ratings to have a mean of 0.

The optimizers *adam* [KB14a] and *adadelta* [Zei12] were also included in hyperparameter tuning. They are responsible for adjusting learning rates based on the most recent gradient updates.

We receive the best results by using the *ReLu* activation function and the *adadelta* optimizer.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

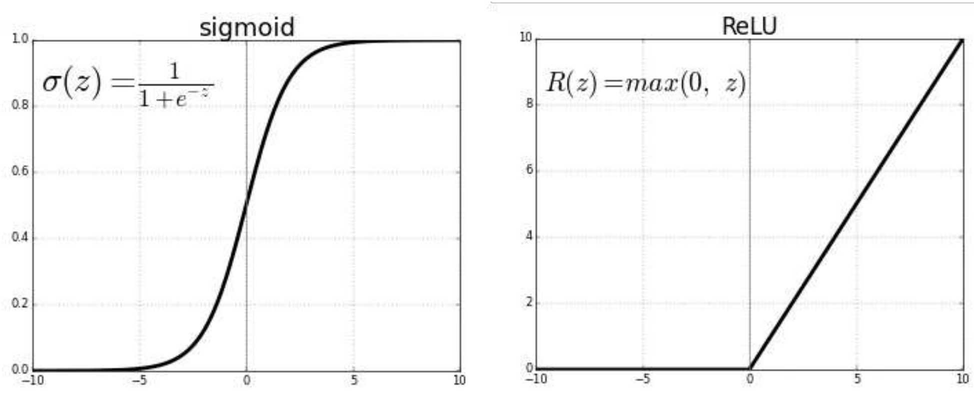


Figure 4.1: The sigmoid and ReLu functions, often used as activation functions in neural networks. The sigmoid function maps inputs to a range of [0,1]; the ReLu function maps inputs to a range of [-1, 1]

We needed to modify the loss function of the Keras library, so that it calculate the Mean Squared Error for each step, but ignores the nodes where we do not have input values for. This is shown in equation 4.8, where n is the number of samples and Y_i and $\hat{Y}_i$ are the input and output values respectively.

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2. \tag{4.8}$$

4.2.4 Denoising Autoencoder

In addition to the parameters in the Basic Autoencoder, we have a denoising factor which distorts the provided input during training. We sample values from X and add a noise factor u to our input Y_i. This is shown in equation 4.9. X follows a standard normal distribution as shown in equation 4.10.

$$Y_i' = Y_i + uX \tag{4.9}$$

$$X \sim N(0, 1) \tag{4.10}$$

We tested denoising factors of 0.05, 0.1, 0.2 and 0.5 during hyperparameter testing.

Regarding the other parameters, we tested 512, 1024, 2048 and 4069 hidden nodes; 10, 20, 50 and 100 epochs; batch sizes of 32, 64 and 128; regularization of 0.01, 0.001 and 0.0001.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

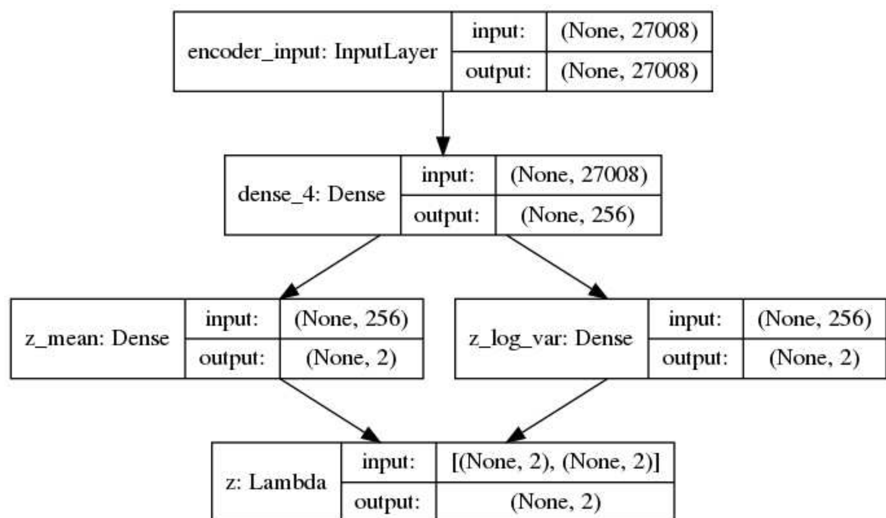


Figure 4.2: The encoder part of the variational autoencoder. It features a hidden layer for encoding the mean and another for the variance.

Setting the denoising factor to 0.1 seems to give the biggest performance boost compared to the Basic Autoencoder. For the rest of the configurable parameters, we found an optimal configuration by using 1024 hidden nodes, 100 training epochs processing batch sizes of 128. The Regularization term is 0.1. As in the basic autoencoder, we found that *ReLu* activation functions and *adadelta* optimizer gave the best result.

4.2.5 Variational Autoencoder

During hyperparameter testing, we tested 256, 512, 1024 and 2048 hidden nodes; 10, 20, 50 and 100 epochs; batch sizes of 32, 64 and 128; regularization of 0.01, 0.001 and 0.0001.

The variational autoencoder has a hidden layer that encodes the parameters of a normal distribution. These are again two hidden layers for mean and variance respectively. This is shown in figure 4.2. The variational autoencoder needs less nodes than the previous autoencoders. We found that 256 hidden nodes, 50 epochs, batches of 64 and regularization of 1e-05 give the best result. Additionally, the optimizer function used is *Adam* [KB14b]. The used loss function between in- and outputs is *Binary Crossentropy*, as it gave better results than MSE.

We based our approach on a proposed variational autoencoder on implicit feedback data sets [LKHJ18], so we adapted the activation functions to work with our input data accordingly. The training phase is the same as in the basic or the denoising autoencoder.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

4.3 Content-Based Approaches

In contrast to the ratings based approaches, the methods in this section only use characteristics of games to compute recommendations. These can be the used mechanics in the games, their genre and other descriptive information like playing time or number of players involved. We name the explicit attributes used in the recommenders below.

So, for a game that the user picked out, the algorithms recommend games that share certain attributes with it. This is facilitated by the fact that certain combinations of attributes occur more often together, as shown in the correlation matrix in figure 4.3.

4.3.1 k-Nearest-Neighbor

The principle of our kNN recommender is as follows. We use all the attributes that make sense to compare with a distance function. This means, we use *Categories*, *Mechanics*, *Playing Time*, *Weight*, *Minimum Players* and *Minimum Age*. There exist 84 different possible category values and 51 different mechanics. Both of these are turned into a binary attribute vector for distance calculation. Playing time can have any value, but is most commonly around 30 - 180 (minutes). The same goes for the other attributes: Weight is a measure of the games complexity or difficulty and is approximately normal distributed around the mean value 2.5 (on a scale of 0 to 5). Minimum Players are often 2 or 3, minimum age is most commonly in the range of 8 — 14.

All these attributes are equally weighted in the distance calculation. After the distance is measured, the results are sorted and the games with the k smallest distances returned.

We do not include the attribute *Family*, as it has 2761 different values and they are not very consistent - they can be very specific or very vague.

In our k-nearest-neighbor algorithm, we experimented with different functions (either euclidean or manhattan distance). We found the best results with euclidean distance, although the results are all very similar. The euclidean metric for calculating the distance d between two vectors X and Y is shown in Figure 4.11.

$$d(\mathbf{X}, \mathbf{Y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_i - y_i)^2 + \cdots + (x_n - y_n)^2}. \tag{4.11}$$

4.3.2 IDF-based recommender

The IDF-based recommender calculates the importance of each attribute, which is inversely related to its occurrence frequency (fitted to a logarithmic scale, as discussed in chapter 2): Rare categories are usually more descriptive of a game. For calculating IDF, we only use the *Category* and *Mechanic* attribute vectors of each game because of their categorical nature. We can't calculate IDF based on continuous measures like *complexity* or *playing time* or non-distinct values like *minimum player age*.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

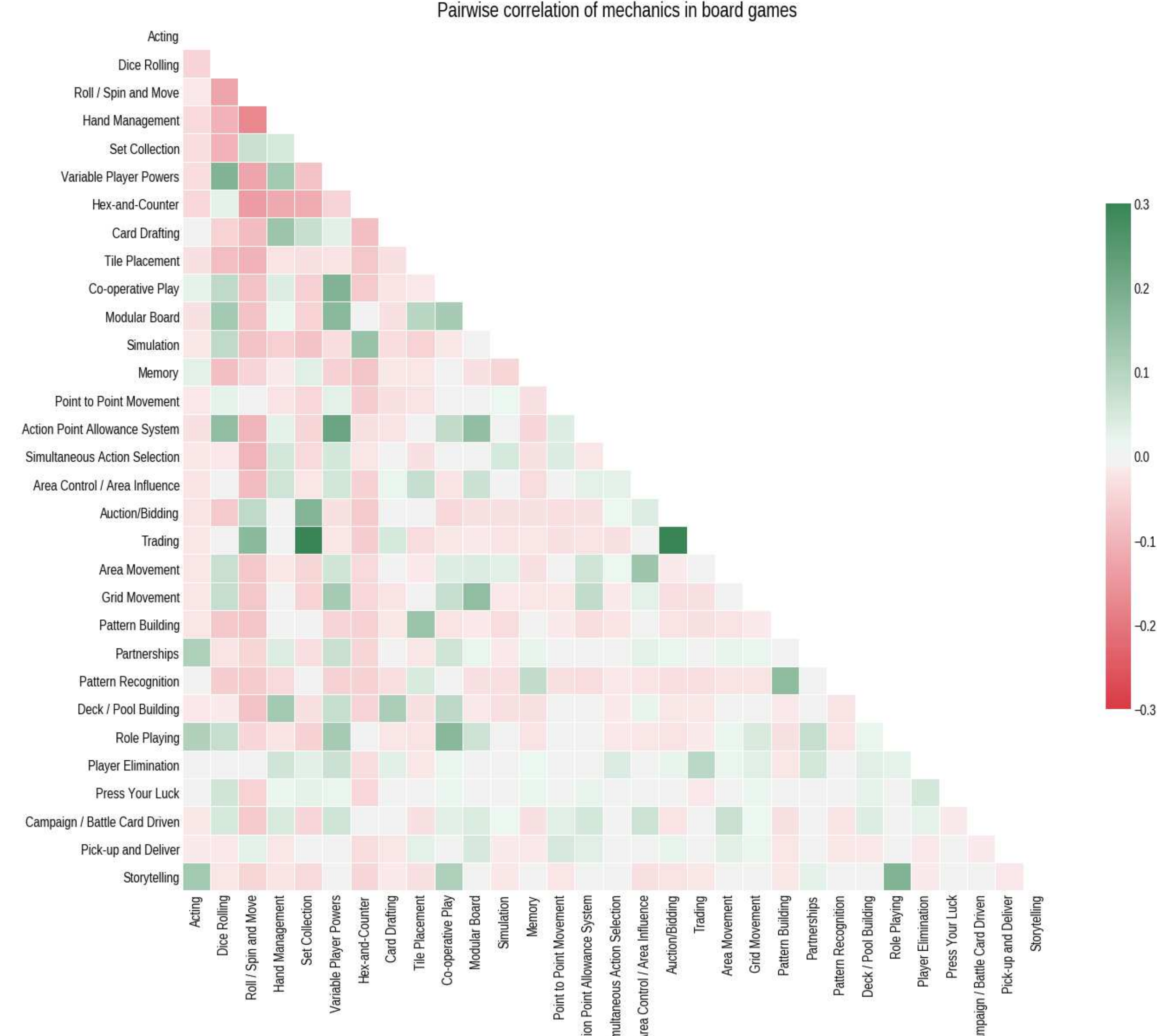


Figure 4.3: Correlation of mechanics in board games. The same mechanics are listed horizontally and vertically. A green field indicates that mechanics are more likely to appear together, a red field implies they are more likely to not occur together.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

The IDF formula for one attribute is shown in equation 4.12. In our case, N is the number of all existing games. df_k are the occurences of an attribute across all games. Therefore, IDF is very high for rare attributes and low for frequent attributes. This IDF value is assigned to each existing category value. This means, that all 84 distinct category values and each of the 51 mechanics values have an IDF value attached to them. Therefore, we have an IDF vector for categories and one for mechanics.

$$IDF = \log \frac{N}{df_k} \tag{4.12}$$

The recommendation process is as follows. Let's assume we want to have recommendations based on one input game. First, we get all the existing mechanics and categories of this game as a binary vector. We then proceed to create a bitwise AND operation with the binary mechanics and categories vectors of all other games. Now, we have all the mechanics and categories all games have in common with the input game. Then, we replace these with the IDF values for mechanics and categories. When we sum these together, we have a score for each game based on one input game. The higher the score, the higher the ranking of the recommended game. The games with the highest scores are output as recommendations.

As we have multiple games as inputs, we compared two approaches of merging the games in the input. In the first, we merged the occurring attributes of all input games and only after, based on this merged representation, calculated a score for the recommendations. In the second, we calculated scores for each single input game and got recommended sets of games for each separate game. The recommended results were sorted after overall score. We found that this latter approach gave vastly superior results.

4.4 Hybrid Recommender

In this section, we present a hybrid autoencoder approach. This method aims to combine some of the best properties of the recommendation techniques presented in the previous section.

This RS can be viewed as an extended version of the previously presented autoencoder approaches. We intend to merge ratings data together with data descriptive of the content of our items. In the training phase, the parameters of the hidden layers are still trained on a per user basis. The target is to most closely resemble the output layer, which still consists of the user's real ratings of their chosen items. However, there is an extra layer now that abstracts additional information on the input side: This is categorical information describing the overall type of items the user prefers.

This categorical information can consist of three types: *Categories*, *Mechanics* and "*Other attributes*". Categories depict the different genres of the games on a broader scale, e.g. "*Economic*", "*Science Fiction*" or "*Negotiation*". Mechanics are a more fine-grained characteristic and describe the gameplay elements of the board game in question.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Examples are "*Dice Rolling*", "*Modular Board*" or "*Set collection*". Finally, the third group, here simply called "Other attributes" consists of "*average playing time*", "*weight*", "*minimum player count*" and "*minimum age*". "Weight" is a measure of the complexity of the game's rules. It is based on a vote by the BGG community and ranges from 1 (least complex) to 5 (most complex). For "player count" and "age" we chose the minimum values as these are more restrictive (and more expressive) than the upper bound.

To make the recommender work, we had to compute values for this categorical information for each user. For the first two attributes, "Categories" and "Mechanics", we used the concept of TF-IDF (described in section 2.2.2). For each user, we took the games the user rated above 7.0 were as a basis for the calculation and computed a TF-IDF value of each possible category and mechanic, as defined in equation 2.11. For the 'other attributes' we simply calculated the average values of all games. For the recommendation phase, this means that those values need to be calculated for each set of inputs as well.

We combined these three sets of categorical data with the ratings of the users. A visualization of this technique is shown in figure 4.4.

During hyperparameter training, we tested 512, 1024, 2048 and 4069 hidden nodes; 10, 20, 50 and 100 epochs; batch sizes of 32, 64 and 128; regularization of 0.01, 0.001 and 0.0001.

After tuning, we found that best results are given by using 1024 hidden nodes, 50 training epochs processing batch sizes of 128. The Regularization term is 0.01.

4.5 Baseline: Popularity-Based Recommender

This recommender generates a set of fixed predictions for each user during the evaluation step. The predictions are the most rated games in the dataset. The idea is, that usually, the best games are played and therefore rated by users. Another possibility would have been to order by decreasing mean rating, but that would overvaluate games that are well-rated, but not so popular.

The same metrics that are applied in the other approaches are applied here. It serves as a baseline comparison for all our RSs and gives an answer to the crucial question: Are our recommendations better than just selecting the most popular games?

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

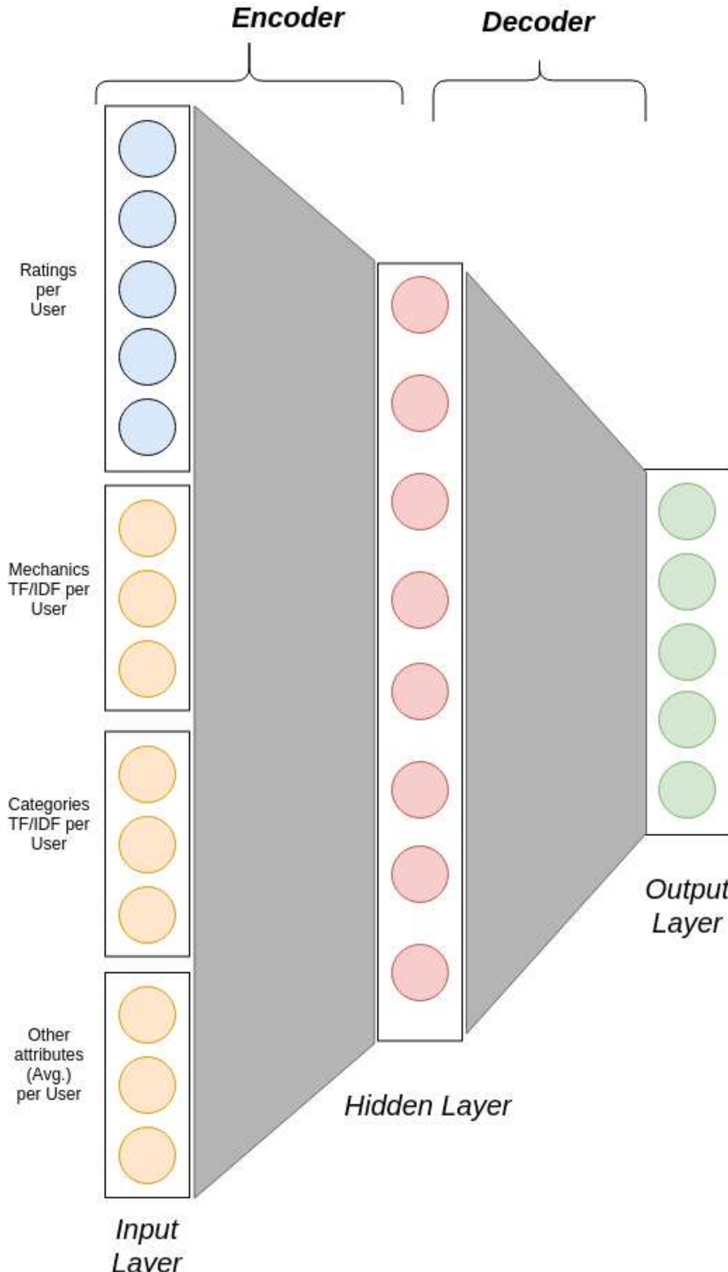


Figure 4.4: The hybrid autoencoder we tested. It uses information about preferred mechanics, categories and other attributes (orange) in addition to the ratings of each user (blue). The intermediate layers (red, purple) are a reduced representation which is trained to match the output layer, consisting of the game ratings (green).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

CHAPTER 5

Software

In this chapter, we will talk about recommendation software. We will first present what software products and libraries are available to build recommender models. We will separate Open Source libraries, Academic RS and commercial products. Then, we will discuss the software and tools we used to conduct our research results.

5.1 Existing Recommendation Software

It is not essential to build a RS from scratch if one wants to use one for their own dataset. There exist quite a few libraries and frameworks with differing degrees of adaptability. We will present the most popular and sophisticated ones in the following. The requirements are that the libraries presented here are still actively maintained and that they are explicitly devised for computing recommendations (general deep learning or machine learning frameworks themselves do not count). In a later chapter we will discuss what pieces of software were used to develop our own recommendation techniques.

5.1.1 Open Source RS

The following recommendation systems libraries are available as Open Source Projects. That means that anyone can clone the project's repository, modify it and even request to add changes to the project. Most of these projects are released under the Apache 2.0 or MIT License, which grant substantial freedoms for modifying the projects and using them, even for commercial purposes.

- **Surprise** is a SciKit library written in Python. SciKit is short for SciPy Toolkit and represents an add-on package for the SciPy Python distribution. SciPy is an open-source Python library that consists of tools for scientific computing and technical computing. Surprise contains various collaborative recommendation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Using movielens-100k data
data = Dataset.load_builtin('ml-100k')

recommender_algorithm = SVD() # Use Singular Value Decomposition
    algorithm

# Run 5-fold cross-validation and print results
cross_validate(recommender_algorithm, data, measures=['RMSE'],
    cv=10s)
```

Figure 5.1: Utilising Surprise's built-in algorithms and datasets, it is very easy to run a RS. In this example, we load a subset of the movielens dataset and then instantiate a SVD algorithm. It is then cross-validated five times and evaluated using RMSE [Hug17].

algorithms that can be used out-of-the-box with included datasets. This includes baseline algorithms, neighborhood methods and several matrix factorization-based approaches. It provides similarity measures like cosine or pearson similarity. It provides interfaces to extend the framework with own algorithm ideas. It comes with two datasets, one containing movie ratings and one being a collection of jokes with user ratings.

Furthermore, Surprise includes tools for evaluation. It provides methods to analyse recommender performance using cross-validation and compare it to other techniques. Figure 5.1 shows a simple cross-validation procedure using an algorithm and a dataset included in the library. It also includes the possibility to run a search over various parameter values to find the best configuration for a RS [Hug17].

- **QMF** has been originally built by engineers at Quora and therefore it is designed to serve the specific needs of a large-scale web company and its data. It is scalable to very large data sets including millions of users. The scalability and speed is due to the fact that it has been built in C++.

  QMF uses matrix factorization models for recommendations with implicit feedback as the input. This means that no explicit rating is given by the user to measure the degree they like or dislike an item. Implicit feedback consists of actions the user performed, for example, an item the user clicked on. QMF includes two different algorithms to compute recommendations using implicit datasets, namely "Weighted Alternating Least Squares" [HKV08] and "Bayesian Personalized Ranking" [RFGST09].

  It can be run from the command line with a wide variety of parameters. It supports computation of ranking metrics like Average Precision, Precision@k and Recall@k

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

```
from spotlight.cross_validation import user_based_train_test_split
from spotlight.datasets.synthetic import generate_sequential
from spotlight.evaluation import precision_recall_score
from spotlight.sequence.implicit import ImplicitSequenceModel

dataset = generate_sequential(num_users=300,
                              num_items=5000,
                              num_interactions=50000)

train_set, test_set = user_based_train_test_split(dataset)

train_set = train_set.to_sequence()
test_set = test_set.to_sequence()

model = ImplicitSequenceModel(n_iter=10,
                              representation='cnn',
                              loss='pointwise')
model.fit(train)

mrr = precision_recall_score(model, test)
```

Figure 5.2: This code generates a sequential prediction model in Spotlight. It uses a randomly generated sequence of user interactions as an input and splits it into training and test data. The trained model is evaluated with Precision and Recall scores.

[YB16].

- **Spotlight** is a recommendation framework based on the PyTorch machine learning library. It is written in Python and claims to allow creation of both "deep" and "shallow" RSs. The "shallow" models are matrix factorisation approaches with either implicit or explicit feedback. The "deep" RS sequential models on the other hand take sequences of user interactions as the input and predict the next item in the sequence. Figure 5.2 shows how such a sequential model can be created and validated.

  Spotlight allows for cross validation and has methods to calculate evaluation metrics like Precision@k and Recall@k. It provides various building blocks to customize the RS. Spotlight allows the use of different loss functions and recommendation models. It includes the Goodbooks dataset, the Movielens dataset in various sizes for experimentation and has methods to generate synthetic datasets according to user-defined parameters [Kul17] .

- **LightFM** provides implementations of multiple collaborative- and content-based learning-to-rank recommender algorithms. Additional to a slew of parameters the user can modify, it offers the choice of four different loss functions and two learning

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
CREATE RECOMMENDER MovieRec ON ml_ratings
USERS FROM userid
ITEMS FROM itemid
EVENTS FROM ratingval
USING ItemCosCF
```

Figure 5.3: Recommenders in RecDB are created using the “CREATE” command. The “ON” clause defines what table is used as the source of user input. The user can specify the columns used for users, items and ratings with the “USERS”, “ITEMS” and “EVENTS” keywords respectively. The “USING” clause finally selects which type of recommendation algorithm should be used for calculating recommendations.

rate schedulers. It is written in Python, but by using Cython extensions it allows for very efficient and scalable recommendations.

It includes metrics like precision@k, recall@k and reciprocal rank. LightFM provides methods for cross validation while training models. It includes the movieset dataset and also two variants of a snapshot from the “StackExchange” network. This is a website where users ask questions and answer other users’ questions. It is developed and used by luxury fashion e-commerce company Lyst and is also being applied in production by a couple other companies [Kul15].

- **LensKit** is a RS that allows for user- and item-based collaborative filtering, matrix factorisation and the slope-one algorithm [LM07]. Every recommender class is shipped within LensKit as a separate module. There is also an evaluation module that contains evaluator classes for applying metrics to results. Due to the fact that every module is easily extendible, LensKit is a handy tool for conducting research on recommender algorithms or evaluation techniques. It is also possible to use LensKit from the command line using its CLI module.

  It is written in Java and an open-source project. It is best suitable for medium-scale operations. It is built and deployed with Gradle and can be integrated as a plugin into Gradle build pipelines. It even supports Groovy DSL for configuration [ELKR11].

- **RecDB** is an Open Source Recommendation Engine and written in C. It is available as an extension for PostgreSQL 9.2, as it is designed to be used from fully within the database. The recommendation algorithms are executed directly in the database when specific queries for recommendations are executed. The queries look very similar to normal SQL queries. The “FROM” clause selects which dataset to use. with additions of the choice of recommender algorithm and choice of similarity measure used. Similarly, already configured recommenders can be removed using the “DROP” command. Refer to Figure 5.3 for an example.

  It includes algorithms like user-user collaborative filtering, item-item collaborative filtering and singular value decomposition. It allows the user to choose Cosine

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

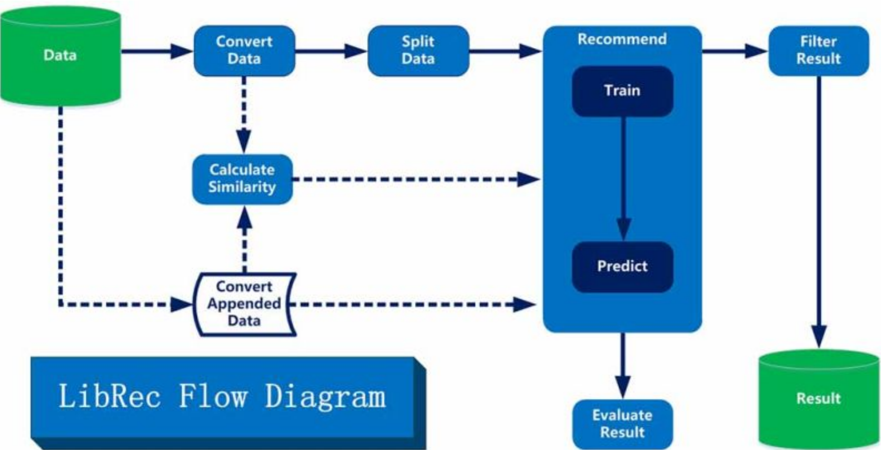


Figure 5.4: An example recommendation pipeline in LibRec, starting from the input Data, processing it to enable training of the recommender all the way to getting the recommendation result and evaluating them [GZSYS15].

Similarity measure or Pearson similarity. It does not come with any datasets but it is bundled with scripts which download variations of the Movielens dataset [SMMA17].

5.1.2 Academic RS

- **LibRec** is a library for RS that implements various collaborative approaches. Its focus lie on providing methods for rating prediction and item ranking. For that end, it includes more than 70 variants of implemented algorithms. It consists of six main components, namely data split, data conversion, similarity, algorithms, evaluators and filters. You can see the execution of the different components in a typical recommendation process in Figure 5.4.

  This means that LibRec comes bundled with tools to create an encompassing end-to-end recommendation pipeline, which includes data pre-processing and evaluation. However, the components are designed to be lowly coupled with each other, so combination with other frameworks is possible.

  It is fully written in Java and distributed over the Maven package manager. The project is well documented and even provides a command line tool to easily generate recommender classes and run algorithms from a configuration file. It provides interfaces that make it straightforward to extend the library with custom algorithms and embed them with the library's components [GZSYS15].

- **RankSys** is a another recommendations engine framework for the implementation and evaluation of recommendation techniques. It focuses on ranking tasks rather

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
rating_prediction --training-file=u1.base --test-file=u1.test
    --recommender=BiasedMatrixFactorization
```

Figure 5.5: Training a Matrix Factorisation model with MyMediaLite.

than rating prediction. What distinguishes it from other offerings is that it contains lots of variations of novelty and diversity metrics. Besides the standard matrix factorisation and collaborative filtering approaches, it includes algorithms like nearest-neighbor recommendation.

It consists of 13 different modules that include algorithms, evaluation techniques and examples. There is one separate module that contains diversity metrics and enhancement techniques, as well as one for novelty. It has been written in Java 8 using state-of-the-art capabilities of the modern language, like streams and lambda functions. [CHV15].

- **MyMediaLite** is a in-memory recommender system that has implementations of more than a dozen recommendation algorithms. It includes algorithms for both implicit and explicit feedback systems, offering ratings prediction as well as item recommendation respectively. In addition to collaborative models it also allows for content-based approaches using attribute data. It is also possible to run MyMediaLite from the command line, offering most of the functionality of the library. The two command line tools are "item_recommendation" and "rating_prediction". And example is shown in Figure 5.5.

  It allows to read and write trained models from disk and incremental real-time updates for its deployed recommenders. It is written in C#, for the .NET platform. However, using the cross-platform mono framework, it is available for Linux, Mac and Windows. It includes evaluation routines using the evaluation metrics MAE, RMSE, Precision@N, MAP, and NDCG among others. It has a long history of being used in both academic institutions and commercial research [GRFST11].

- **SMORe**, formerly proNet-core, is a framework for variant weighted network embedding techniques. Refer to Figure 5.6 for a demonstration of how SMORe generates embeddings of various input data.

  One of the common use cases for this frameworks is generating recommendations. It allows for interaction through a command line interface. Included models are BPR [RFGST09], DeepWalk [PARS14], HOP-rec [YCWT18], CSE [CWTY19], and HPE [CTLY16]. It is written in C++ [CYCT17].

- **LIBMF** is a matrix factorization tool for recommenders that includes a few optimisations. It supports on-disk training, which reduces memory load and parallel computing for multi-core CPUs using CPU instructions to accelerate vector operations. It also supports cross validation. There are APIs available to include

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

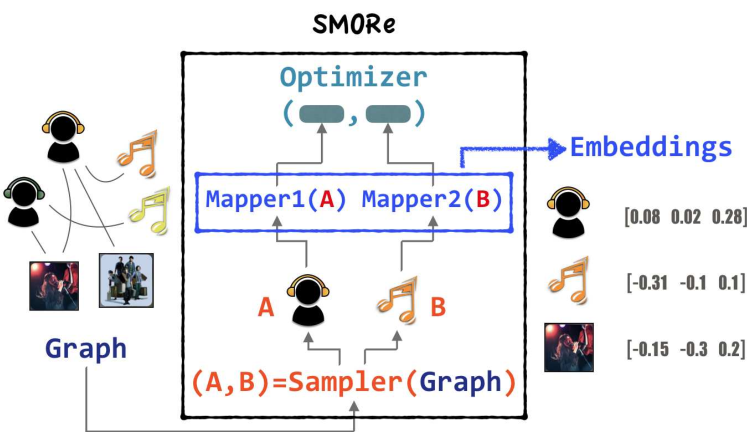


Figure 5.6: The input for SMORe recommenders is a network graph. In this example, we have music titles, artists and songs. The framework generates embeddings for each separate entity [CYCT17].

LIBMF into R and Python projects, respectively called "R LIBMF" and "Python LIBMF" [CZJL15].

5.1.3 Commercial and Software-as-a-service applications

There exist a couple of commercial recommendation offerings. Most of these are distributed as SaaS (Software-as-a-service) applications, which is a software distribution method where the software is licensed on a subscription basis. In most cases, there is a set fee for a certain amount of API calls (e.g. recommended item requests for 100 users).

- **Amazon Personalize** In June 2019, AWS added another Software-as-a-service product among there vast range of cloud-based products. Amazon Personalize makes the promise to bring the recommendation technology that powers the e-commerce site amazon.com to the masses. The stand out feature is support for real-time refinement in the recommendation feature, so that the user can get instantaneous suggestions while interacting with an application. Figure 5.7 shows the capabilities of Amazon Personalize and how it is integrated into the AWS ecosystem.

  It is essentially a machine learning API that enables developers to fetch individualized recommendations for the customers who are using their applications. The application of the service is flexible: It's up to the developer to decide what sort of recommendations these are - clicks, page views, signups, purchases or other. These can be added as search results, personalized notifications or just recommended items in a user interface. It also brings a range of pre-trained models. In Figure 5.8, we show a short example of how to bootstrap Amazon Personalize to create

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

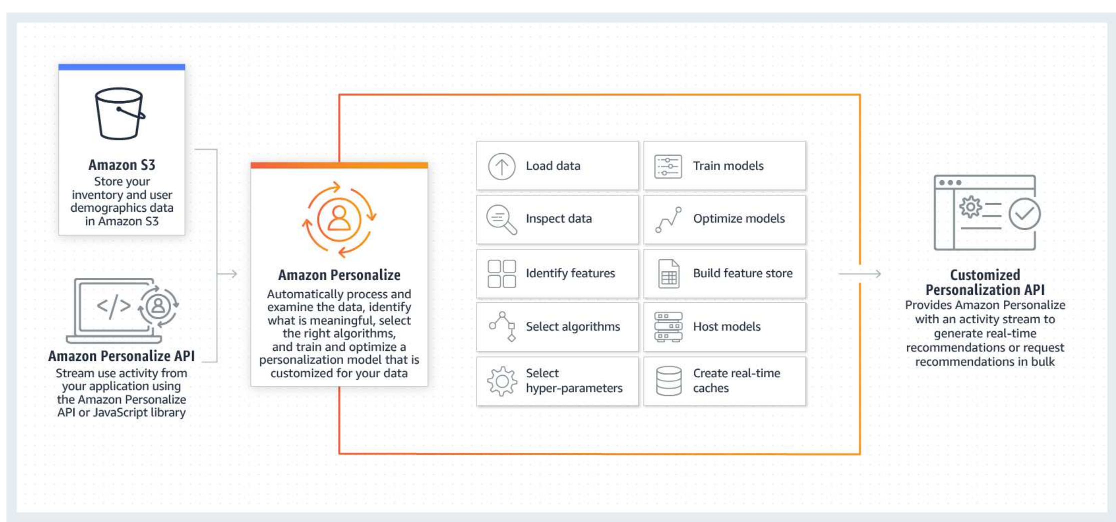


Figure 5.7: In the middle orange box, this figure shows the capabilities and feature offerings that Amazon Personalize includes. The user has the possibility to choose a recommendation model, configure used hyperparameters and run optimizations. On the left we have the input of the recommender systems. These can stream input data from an app directly to refine the recommenders in real time. Another way is to use data at rest in S3 (AWS' cloud storage offer). The output of Amazon Personalize on the right side makes it easy to integrate recommendations into user interfaces.

suggestions for a user. It only uses the Python library boto3, which allows for programmatic configuration of AWS resources.

- **Yusp** Yusp, formerly Gravity R&D, is a company formed by team "Gravity", the finalists of the Netflix Prize Competition. They were founded in 2007 and are headquartered in Budapest, Hungary. They offer tailored recommendation services for various industries like Brick and Mortar franchises, retail banking and streaming services. The type of recommendation service provided ranges from E-mail personalization to product recommendations and online advertising.

  They are openly and regularly contributing their research findings to the public, however, some patents exist on certain algorithms. In 2016, Gravity served over 4 billion recommendations per month, which accounted for about 30 million $ in revenue. They have a close cooperation with Deloitte for business solutions. Some of their customers include T-Mobile and dailymotion [TPNT07].

- **Peerius** Peerius is a company that was founded in 2007 in the US and offered similar services to Yusp. Since it has been acquired by Episerver, and was integrated into their enterprise offerings. They distribute a Content Management System (CMS), which allows customers to build commerce websites. This includes retail digital commerce, B2B commerce and financial services. The recommendation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
import boto3
personalize = boto3.client('personalize')

# 1. Create a "solution" and specify the Algorithm
response = personalize.create_solution(
   name = "SolutionName",
   datasetGroupArn = "ARN for the input dataset",
   performAutoML = True)
solution_arn = response['solutionArn']

# 2. Create a 'campaign' based of this solution.
response = personalize.create_solution_version(solutionArn =
    solution_arn)
solution_version_arn = response['solutionVersionArn']
response = personalize.create_campaign(
   name = 'campaign name',
   solutionVersionArn = 'solution_version_arn',
   minProvisionedTPS = 10)

# 3. Generate recommendations for a user
camapaign_arn = response['campaignArn']
personalizeRt = boto3.client('personalize-runtime')
response = personalizeRt.get_recommendations(
   campaignArn = camapaign_arn,
   userId = 'User ID')
```

Figure 5.8: How to use Amazon Personalize to generate item recommendation. First, you have to create a solution by specifying an algorithm to use. If none is specified, as in this example, AWS will choose one for the user. By creating a solution version, the model is trained. This is used in the second step when creating a pointer to the solution campaign. In the third step, the campaign is used to query recommendations for a user. It is important to note that these 3 steps would be usually called in different parts of the application, depending on the use case.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
RecombeeClient client = new RecombeeClient("DataBase", secret);
client.send(
  // user "user_id" viewed item "item_x"
  new AddDetailView("user_id", "item_x"))
);
RecommendationResponse recommended = client.send(
  // fetch 10 recommendations for user "user_id"
  new RecommendItemsToUser("user_id", 10)
);
```

Figure 5.9: How to use Recombee's provided library to update information on users and get recommendations. The piece of code shows how to get a new connection to a database stored at Recombee. Then, updates to user views can be made. This is possible as Recombee allows for real-time modification of its algorithms. Then, recommendations can be fetched and presented to the user.

capabilities of Peerius include personalised search, content recommendation and product recommendations.

- **Recombee** Recombee is a recommender engine which promises its clients to reach their KPIs, increase conversion rate and maximise the enterprise's turnover. One big selling point is the ability to scale flexibly with the client. They apply a variety of machine learning and artificial intelligence algorithms to generate results, however they are unclear about specific algorithms and technologies used.

  On their website, they mention using a mix of deep-learning and collaborative filtering for numeric information. The collaborative methods they use are based on Matrix Factorization, Nearest-Neighbour and association rules, however, they do not disclose to what amount they factor in each type of algorithm. For images and textual data, they also apply content-based recommendation algorithms (detecting similar images or textual descriptions). Needless to add, their models are optimized automatically, with no possibility to modify the training.

  A unique selling point are the specialised models they provide for very narrowly focused business needs. This includes models that recommend items which are very different to each other (increasing diversity), models that take into account popularity trends or periodicity models that suggest items based on repeating behaviour of users.

  They offer libraries for JavaScript, Python, Ruby, Java, PHP and Node. General integration over a REST interface is also possible. An example of using their Java library and interacting with the Recombee API is shown in Figure 5.9. Recombee allows for modification of the rules that the recommendation software uses. It provides the functionality to specify rules through filtering or boosting the results based on client-defined attributes of the items in the client's database. For easier

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

interaction, they offer a custom query language, ReQL, in addition to their library offerings.

- **Segmentify** Segmentify is another SaaS tool that offers businesses the possibility to integrate product recommendations into their website. On top of that, it offers personalized search, personalized email and push notifications. As the name implies, one big selling point is the possibility to segment customers into three different groups: Active loyals, churned customers and one time shoppers. Strategies of how to target each segment vary. Segmentify offers a real-time dashboard to gather insights about customer behaviour. Unfortunately, the recommendation process is a black box. They do not reveal how recommendations are made.

5.1.4 Summary of Existing Recommendation Software

As we have just shown, there are a lot of available libraries to build recommendation engines out-of-the-box.

The advantages are that these are easily adaptable and accessible. Lots of frameworks are written in very popular languages like Python or Java and are therefore easily extendable. They tend to be very fast to set up, and are used often for prototyping and experimenting with new concepts.

However, they are rarely optimized (often due to the technologies used) and the performance drops considerably with huge data sets. Most of these frameworks are also not optimized to work on multi core processors or on distributed infrastructure and are rarely used in a production environment for these reasons.

For the commercial and SaaS offerings, the arguments change slightly. The advantage of these platforms is that a lot of the heavy lifting disappears when using these services. There is no need to hire a specialised machine learning engineer when the model is chosen and trained for you. You even do not need to hire many operations-focused engineers as the recommendation system runs in the cloud, using third parties' resources. Tasks like hyperparameter tuning are automated.

However, there are also a few disadvantages to use these platforms. First of all, a lot of the models use aim to provide a one-size-fits-all approach. Some services like Amazon Personalize can choose a RS for you. However, there is no guarantee that the solutions provided are the best, or even remotely good. These services usually offer metrics, but to interpret and adjust them you need specialised personnel. Since the SaaS solutions are not open-source, you cannot inspect the source code and deep dive on the algorithms implemented.

The last disadvantage is that these services are expensive. In most cases, there is no flat-rate available, you pay for every time recommendations are computed for a user. It can happen that this type of recommendation software was chosen for it's ease of use and rapid deployment, but as an application's lifecycle unfolds and it scales, it can become very expensive. Arriving at the point where you have thousands of users or more, chances

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

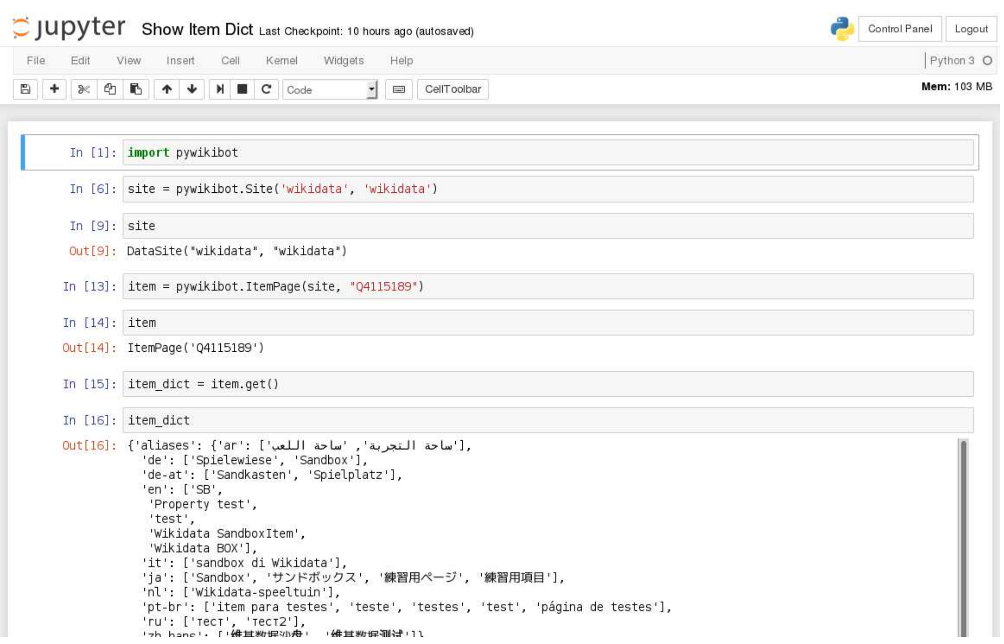


Figure 5.10: An example of a Jupyter Notebook instance running in a web browser while working with code in Python.

are, that your application is already so tightly integrated with the SaaS solution that it is hard to migrate (vendor lock in).

We have not discovered one framework that provides an easily adaptable recommendation solution that operates on Autoencoders or neural nets. The reason might be that this is a fairly new approach in the community. It is one of the approaches that we heavily focus on and, as we see later, shows to give excellent results.

For these reasons, all the techniques we applied were implemented by us from scratch in Python, as we will discuss in the next section.

5.2 Applied Software and Technologies

In this section, we will highlight the technologies that we used to reach our conclusions. We will discuss the programming languages used for our project, the editors and development environment utilised and which libraries we made use of.

5.2.1 Editor - Jupyter Notebook

Jupyter Notebook is one of the products that emerged from the nonprofit organisation known as “Project Jupyter”. The group has been formed in 2014 by Colombian physicist and software developer Fernando Pérez. Jupyter Notebook is cloned from the “IPython”

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```java
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

Figure 5.11: Printing “Hello, World!” to the console output using Java 8.

command shell, which is a command shell for interactive computing. It lends itself very well for scientific computing. It is open-source, free to use and released under a very generous software license.

Code snippets, their execution and supporting formulas, explanations and diagrams are all integrated into the same document. This type of development environment is known as a “notebook interface”. It has first been popularized in 1988 by Wolfram Mathematica 1.0 on the Macintosh. The name “Jupyter” refers to the supported programming languages - Julia, Python and R. These languages are the most popular amongst practitioners of scientific computing, statistics or data science. Some of the features include type introspection, integration of rich media like interactive data visualisation, tab completion and a log of past executions. Figure 5.10 demonstrates an example of the application. Many extensions exist for Jupyter Notebooks, for example a plug-in that allows to generate PDF documents out of a notebook file.

One of the most important reasons for the widespread adoption of Jupyter Notebooks is the separation of the interface and the execution of the programming logic. The execution is done in a so-called Jupyter Kernel. Kernels can be connected to multiple instances of notebooks. During our research, all development has been done using a Jupyter Kernel on a local machine - the same one where the Notebook instance resided. However, for large-scale tasks, it is essential to use the increased capabilities of distributed computing. It is possible to launch Jupyter kernels in a distributed cluster. The big cloud providers provide already their own versions of Jupyter Notebooks - examples are Amazon’s SageMaker Notebooks, Google’s Colaboratory and Microsoft’s Azure Notebook. The notebook interface has become so popular, that many projects exist now for a lot of different programming languages, like JavaScript.

5.2.2 Programming Language - Python 3.7

Python has become one of the most popular programming language. According to the TIOBE index, which measures the popularity of programming languages, it currently resides in the “Top 3” of all languages (Status: November 2019). Python has been first released by Dutch software engineer Guido von Rossum in 1991, who developed it first as a hobby software development project to keep him busy during Christmas holidays. He named it after British comedy group Monty Python.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

```
print("Hello, World!")
```

Figure 5.12: In contrast, printing “Hello, World!” using Python 3.7 to the console only takes one line.

Some of the characteristics that make Python unique from other programming languages are also the reason for its widespread success. First and foremost, it is a general-purpose language that can be used in a wide variety of application domains. Its usage ranges from scientific computing to running lightweight servers to enterprise software development. It is an interpreted language, which means that the instructions in the code are executed directly. There is no intermediate compilation step to produce machine-readable byte code, like in Java for example. This is made possible by the interpreter, which directly translates every line into subroutines readable by the execution environment.

Finally, Python is a high-level programming languages. It means it abstracts a lot of responsibilities that are inherent to computing, like memory management, away from the user. This level of thinking is visible when considering the language syntax of Python. It is very clean, minimal and expressive. For an example comparison, we show two variants of printing the line “Hello, World!”. The Java implementation is shown in Figure 5.11. The equivalent in Python only needs one line of code and is displayed in Figure 5.12.

The ability to express programming logic in a very concise manner without much boilerplate or overhead makes Python very easily readable. It includes a very sophisticated standard library. It has functionality for building user interfaces, executing unit tests, building network interfaces or methods for statistics and maths. This is probably the main reason why it garnered so much popularity among the Data Science community. Expressing the idea of an algorithm in Python is not much more complex than formulating it using pseudo code - with the added benefit, that, in Python, the algorithm is ready to be executed and tested. One such example is shown in Figure 5.13, where we show the code for a function that computes a factorial. One remarkable oddity of the programming language that is visible here, is how Python uses only whitespace for indentation. Usually, in languages like Java or C, curly brackets or keywords have been used to indicate the begin and end of assignment blocks. In Python, assignments that are indented the same are on the same level of code execution. Python uses duck typing, meaning that the nature of an object is determined by existing methods or functionality, rather than an explicit type. This is possible as type constraints are only checked at run time, not compile time. Python is still strongly typed however, only allowing for well-defined operations.

Additional functionality can be included by using the *import* statement. Extensions to the Python language are distributed via modules. In the following, we will discuss additional modules for the Python 3.7 programming language that we used for our research.

Python features a garbage collection for memory management and is dynamically typed,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
def factorial(f):
   if f < 0:
      return 0
   else:
      result = 1
      i = 2
      while i <= f:
         result = result * i
         i += 1
      return result
```

Figure 5.13: A function that calculates the factorial of an input *f*. It uses *result* as a temporary variable to store intermediary computation results and returns it when the iteration is done.

allowing for liberal modifications of objects. It does allow writing application code by using many different paradigms, for example procedural code, object oriented programming or functional programming. Python 2.0 was released around 10 years after the original Python. Python 3 was first released in 2008 and featured lots of major structural revisions. Applications written in Python 2 were not compatible, however, support continued for Python 2 due to the widespread adoption. In 2020, Python 2.7 will be officially deprecated and will cease to be supported. There exist reference implementations of Python, most notably CPython, which can compile Python code into intermediary byte code.

5.2.3 Libraries and Modules

The established standard to installing extensions for the Python programming environment is by using the package manager "pip" in conjunction with the Python Package Index, or short, "PyPI", and this is what we used in our project to manage third-party packages as well.

PyPI is the official software repository for third-party extensions for Python. Currently, there are over 200.000 packages available, accumulating more than 1.5 million releases and over 380.000 users (Status: November 2019). When publishing a package to PyPI, it is necessary to provide some metadata in addition to the raw package files. PyPI is capable to handle version histories of packages as well as different builds for different Python versions.

Pip is the standard package management system for Python. It is a tool to install and update Python packages on a machine. The acronym 'Pip' is a recursive acronym - as it was in fashion among computing enthusiasts and hackers since the 1960s - that means "Pip installs packages". Pip comes preinstalled with most Python distributions and is used over the command line. For bigger projects, it is possible to create a text file containing all dependencies and pass it to pip, which installs the correct libraries in the specified versions. When working on several projects, it is recommended to create

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

```
###### requirements.txt ##############
#### Dependencies without version ######
numpy
beautifulsoup4
#
#### Dependencies with version #########
pandas == 1.5.1 # Must be version 1.5.1 ##
csv >= 2.2.1   # At least version 2.2.1 ##
####### end requirements.txt ##########


pip install -r requirements.txt
```

Figure 5.14: This figure shows a text file with the name requirements.txt. It contains all packages that must be installed for the project in which this file is contained to run correctly. It is possible to specify a version number, but not obligatory. When specifying a version number, one can be specific or give minimum or maximum version specifications. In the last line, we feed this file to the "pip install" command.

a virtual environment for each, so that there is no conflict among interfering version numbers. Figure 5.14 shows how to install packages from a given requirements file via the command line. It is not necessary to use PyPI as a software repository, but it is set as the default setting. It is possible to set multiple sources, which is handy for companies maintaining their own private repositories.

Boardgamegeek

"Boardgamegeek" is a library for Python 3 that makes it easy to interact with the official XML API from BoardGameGeek.com. The XML API allows to send HTTP requests with the use of and URI of the format boardgamegeek.com/xmlapi2/*type*?*parameters*. For the *type*, it is possible to specify games, users or communities. The given parameters have to include an ID or name of the item to retrieve but can contain additional boolean flags to add or dismiss certain criteria, for example to include comments by a user or not. When executing a correctly formatted request, the resulting response is an XML file containing information of the specified object. If it was a game, it will contain data on ratings, game details etc., which are navigable by their respective XML tags.

The Boardgamegeek Python library provides a layer of abstraction over this. The library is based on an older library called "libBGG". The advantage of the use of this library over the API is that it allows a structured approach to execute requests. All the entities that are retrievable over the XML API are modeled as Python classes, which allow for more intuitive handling of data than parsing an XML file. The available classes are Users, Games, User collections, Player Guilds, Plays and Hot items (currently popular items). For our purposes, we use data on Users and Games to build our recommendation engines.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

Users give us the individual ratings on collections owned by users while Games provide us with information on attributes like mechanics or game types. As we effectively mirror the BoardGameGeek database on user ratings and need data on all present games, it is important to space out requests being sent to the BoardGameGeek server. When sending a lot of requests in a very short amount of time, the risk exists that your requests are being classified as a DOS (Denial of service) attack and you are being blacklisted as a result.

Keras

Keras is an open-source library written in Python. Its aim is to provide very high level abstractions for programmers to build neural networks. As such, it doesn't implement the actual neural networks, but runs as a layer on top of a neural network backend such as TensorFlow, CNTK or Theano. The goal is to facilitate rapid prototyping of ideas for neural networks and get results as quickly as possible. Its components are designed to be modular and extensible. Computation can be run on either GPU or CPU. Since 2017, both Google and Microsoft support Keras in the core libraries of Tensor Flow and CNTK respectively. It has become one of the most commonly used tools among machine learning engineers with over 200.000 users.

Keras is roughly split up into three different modules, *Models*, *Layers* and *Preprocessing*. A model is the starting point of a neural network, it denotes a way how to organize the neural layers. There are two model classes available in the Keras *Models* module: The *Sequential* model, which allow for stacking neural layers one after the other, as well as a generalised *Model* class that is fully configurable with a functional API and allows for more complex network architectures. The *Layers* module provide the computational components for a model. Keras provides a variety of layers, including convolutional layers, recurrent layers, embedding layers, normalization layers and many more. Finally, the *Preprocessing* module includes tools for image, text and sequence preprocessing. Keras provides a variety of optimizer and activation functions for neural network training as well as metrics for evaluation. We used Keras in combination with a TensorFlow backend for the development of our Autoencoders.

Pandas

Pandas, originally developed by Wes McKinney, is a software library for Python that facilitates manipulation and processing of data for data analysis tremendously. It is open source and released under the very permissive BSD software license.

The core object in Pandas is the dataframe. A dataframe is a multi-dimensional store of data values that can be accessed by indices. In most cases, a dataframe is two dimensional, containing rows and columns. Columns and rows can be labeled. Slicing and creating subsets is possible by either using labels or indices. It is possible to store time-series data in dataframes as well and to execute domain specific operations such as moving window statistics. Pandas allows for many operations that are commonly found in database

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

query languages, such as aggregation, grouping and filtering. It is possible to import and export from two-dimensional textual representation file formats like *.csv, *.tsv (comma- and tab-separated values) as well as Excel files. It also includes functionality for data preparation, like handling of empty and null values.

We use Pandas in almost all aspects of our work. All the input data is parsed into dataframes, which is then processed by using pandas capabilities to operate on all the data at once. For example, we apply the normalisation of ratings in this manner. Pandas is also used for recommendation computation itself, with the exception of the autoencoders, which are handled by Keras. We use dataframes to store information on users and ratings. For our content-based recommenders, we store categorical information in them.

SciPy, NumPy and Matplotlib

SciPy is an open-source library for Python aimed at scientific computing. It extends on the array object contained in NumPy, which allows for calculation with multi-dimensional arrays and matrices. Matplotlib is a data visualisation library for Python that makes plotting and graphical representation of data easy. These three tools work very well together and therefore are often used together in practice.

In our project, we use the stats module of the SciPy library to calculate our evaluation metrics, namely: Precision, Recall, NDCG, Novelty and Diversity. The stats module, which makes it easy to calculate the separate components of the metrics formulas, uses NumPy in the background. We used Matplotlib to create the figures representing our data sets in this paper.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 6

Results

6.1 Evaluation Methodology

The performance of RS has been evaluated in research literature since 1994 with the GroupLens recommender [RIS+94]. Evaluation can be done using **offline analysis** or **live user experiments** (**"online"** evaluation) [HKTR04].

In offline evaluation, the RS is used to predict several withheld values from the dataset. The data is split in training set and test set. The advantages of offline evaluations are that they are relatively simple, quick and efficient. The only requirement is to have the dataset ready. A major weakness, however, is that they do not exactly tell if users themselves will prefer one RS over the other, as there is no feedback on how the recommended sets are perceived. All that is available is a set of ratings.

Online approaches, however, needs the participation of a community of users who interact with the RS. They can be either performed in a controlled environment or with "real" users in a production system. Depending on the form, the evaluation of online approaches vary. Performance can be determined with surveys about user satisfaction or keeping track of various user metrics (e.g. participation rate).

While online approaches give a better feedback on the perceived performance, they involve more preparation to execute. In a controlled user study, participants and useful questionnaires have to be prepared. Well-defined hypotheses that need to be tested should be prepared. In these cases, a certain bias still pertains. To avoid this, a RS can also be evaluated in a field study. This involve the integration into a live production system where users already exist. In these cases, conclusions can be drawn from recorded user interactions or with explicit feedback. While these reveal more of the real users in their own context, they might not be as focused as controlled experiments.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

We perform our evaluation with offline metrics. They can be differentiated into three metric categories measuring different qualitative aspects: Predictive accuracy, classification accuracy and rank accuracy.

Predictive accuracy metrics measure how close the predicted rating of an item by a user resembles the actual rating given. An example would be to calculate the mean absolute error of the rating difference. This is useful for systems where an exact rating is given to users and relevant to them. However, if users are only interested in the top results, it is unnecessary to calculate this metric for all rating predictions.

Classification accuracy metrics quantify how often the predictions of a RS are, overall, good. This is valuable for RSs where the general goal is to recommend good items, but no interest is given to an assumed rating or strict ranking. The users taste should be binary, they either like or dislike an item. Recall, Precision and related measures like Mean Average Precision (mAP) are classification accuracy metrics we present in the following. They are often implemented with a cutoff point, simulating how a user generally only considers the first few items.

Finally, **rank accuracy metrics** are used to evaluate the order of a set of recommended item. These are useful in domains where the order of recommendations matter to the user. It is not useful in domains where the ordering does not matter too much - a rank accuracy metric will give different results if the best result is ranked fifth instead of first. An example metric is Normalized Discounted Cumulative Gain (nDCG).

6.2 Evaluation Metrics

To reliably judge the performance of our models, we are using Precision at k, nDCG at k and mAP [SMR08] as our main guiding metrics. Precision at k and mAP are related to Precision and Recall, which are standard metrics in RS literature for tracking performance. Furthermore, we use Diversity and Novelty to describe certain characteristics of recommended sets.

To ensure comparability, each RS is evaluated on the same metrics and therefore provides identical input/output formats.

6.2.1 Precision and Recall

We first discuss Precision and Recall as they are heavily drawn upon by the other metrics. Precision and Recall have been the most popular evaluation metrics for information retrieval systems since their introduction in 1966 [CMK66].

In our use case, we want to determine how many games out of a recommended set of games are relevant to a user. Relevant games are defined as games that were rated with a rating of 7.0 out of 10.0 or higher by said user (i.e. games that the user judged as 'good').

TU Bibliothek, WIEN Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Precision presents the fraction of the relevant games in the recommended set to the total number of games in the recommended set, as shown in 6.1. In our evaluation, we fix the set size to k=100, so the denominator in the Precision formula is always 100.

$$\text{Precision} = \frac{|\{\text{relevant games}\} \cap \{\text{retrieved games}\}|}{|\{\text{retrieved games}\}|} \tag{6.1}$$

Recall is the fraction of relevant games in the recommended set compared to the total count of relevant games, as shown in 6.2.

$$\text{Recall} = \frac{|\{\text{relevant games}\} \cap \{\text{retrieved games}\}|}{|\{\text{relevant games}\}|} \tag{6.2}$$

In most cases, it is necessary to show Precision and Recall together, as they are inversely related to each other [Jiz00]. This conclusion is quite intuitive: When the size of the recommended set increases, it is likelier that relevant games appear - Recall increases. However, at the same time, the denominator in the Precision formula increases - Precision drops. The F1-score is one approach to combine the two metrics, shown in 6.3. By comparing the F1 measures of different simulation runs, it is possible to select an overall best configuration.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{6.3}$$

6.2.2 Precision@k & Recall@k

While Precision usually takes into account all retrieved items, Precision@k (Precision at k) only concerns the very first k retrieved items into the calculation. k can be set arbitrarily, e.g. at 5, 10 or 100. Precision at k considers the fact that users will only look at a fixed number of top results when using a recommendation system. For example, when people use a search engine, they only look at the first results page but barely more. The same goes for Recall@k.

6.2.3 Normalized Discounted Cumulative Gain (nDCG)

nDCG takes the notion of Precision at k a bit further. The metric uses a graded relevance scale to attribute the usefulness of recommended results based on their rank. This gain is based on the position in the result list. Higher results have a higher usefulness, visible in the numerator in the formula, while the gain is discounted at lower ranks, apparent in the denominator in equation 6.5. Consequently, it is also executed over a set number of top results k [SMR08]. This metric is executed over a set of users U. $\frac{1}{IDCG_{kj}}$ is a normalization factor that transforms the nDCG of a perfect user for the user j at k to 1. $rel(j, m)$ is the relevance of the item m in the user j. In our case, this would be the rating of the game (given by user j for the item m). $optrel(j, m)$ is the ideal relevance

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

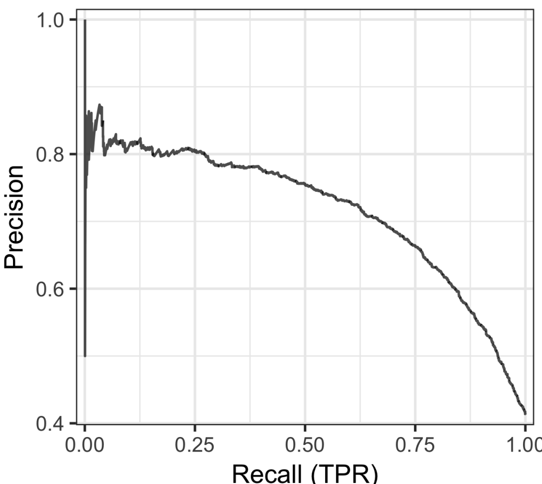


Figure 6.1: An example of a Precision-Recall curve. [Ber16]

factor that is possible for the item at position m. In our case, the ideal values are taken from the existing user's rating.

$$NDCG(U,k) = \frac{1}{|U|}\sum_{j=1}^{|U|}\frac{1}{IDCG_{kj}}\sum_{m=1}^{k}\frac{2^{rel(j,m)}-1}{log_2(m+1)} \tag{6.4}$$

$$IDCG_{kj} = \sum_{m=1}^{k}\frac{2^{optrel(j,m)}-1}{log2(m+1)} \tag{6.5}$$

6.2.4 Mean Average Precision

mAP is calculated over a set of users. It is the mean of the Average Precision (AP) for each of these users. In our case, the set of users consists of “recommendation” users for one user.

The AP for a specific user is calculated as the average precision score from every recall value from 0 to 1. Usually, Precision and recall are inversely correlated. It is possible to plot the precision at each recall value, as seen in figure 6.1. The AP can therefore also be understood as the area under the precision-recall curve.

The AP for one user is shown in equation 6.6, where $r \in R$ symbolizes the set of ratings for each recall level. mAP for each user $u \in U_R$ is shown in equation 6.7.

$$AP = \frac{1}{|R|}\sum_{r\in R} precision(r) \tag{6.6}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

$$mAP = \frac{1}{|U_R|} \sum_{u \in U_R} AP(u) \tag{6.7}$$

6.2.5 Diversity and Novelty

Contrary to Precision and Recall, which are based on the ratings of users, we also use metrics that are based on the characteristics of the games itself. The idea is that the user of the RS perceives the recommendations as interesting and fresh - if the recommendations are too similar to their already preferred games, users might not find the recommendations too exciting. To achieve this, we use Novelty and Diversity [CVW11], which we tailored to fit the characteristics of our items.

Novelty of a recommended item refers to how different it is to the items that the user already knows and likes. Formulated in another way: How different is the output of the RS to the input used for computation?

Diversity, on the other hand, only regards the recommended set itself. How different are the items in it to each other? Recommendations are more interesting to the user if they are more dissimilar to each other, offering the user a broader spectrum of items to investigate.

We calculate Diversity and Novelty by computing the pairwise distances of the games' mechanics and categories to each other. These attributes are stored as boolean arrays, which makes it practical to use Hamming Distance to calculate similarity. Hamming Distance between two vectors of identical sizes is defined as the count of positions at which both vectors differ. The higher the hamming distance, the more dissimilar the items are. In our formulas of Diversity and Novelty, we divide the Hamming Distance (absolute count of varying attributes) by total number of present attributes and then take the mean of all games. This is illustrated in the formula in 6.8. The essential difference between Diversity and Novelty is that the former represents the total of pairwise differences between items of the same set (6.9), whereas the latter represents the pairwise differences in two distinct sets (6.10). X and Y are binary attribute matrices. X is the list of recommendations. Y is the list of the user's liked items. The difference $X_i - Y_j$ denotes essentially the Hamming Distance between column j of matrix Y and column i of X.

$$Diversity/Novelty = \frac{\frac{\#DiffMechanics}{\#TotalMechanics} + \frac{\#DiffCategories}{\#TotalCategories}}{2} \tag{6.8}$$

$$Diff_{Diversity} = \sum_{i=1}^{k} \sum_{j=1}^{k} |X_i - X_j| \tag{6.9}$$

$$Diff_{Novelty} = \sum_{i=1}^{k} \sum_{j=1}^{l} |X_i - Y_j| \tag{6.10}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

6.3 Training and Test Data Selection

We perform the training of our RS with the total available data. To find the best parameters for our algorithm, however, we perform hyperparameter tuning with a representative subset of the available data.

In machine learning, a "hyperparameter" denotes a parameter that is set to an arbitrary value, before the system is trained. All other parameters are then derived during the training process. Hyperparameters can not be inferred during the training phase. When setting the hyperparameters of a model, we can speak of its "configuration". The amount of hyperparameters is different from model to model. Some simple models do not require any, but usually machine learning models demand a few hyperparameters to be able to run.

"Hyperparameter tuning" or "Hyperparameter optimization" describes an approach to discover the best fitting set of values for the hyperparameters. As it is most often not straightforward to understand the effect that one value for a parameter as opposed to another, it is necessary to train a few models with different hyperparameter configuration, evaluate them and then compare the performance. Since most hyperparameters are integer or floating point based, there are many possible values with which they can be configured with. It would be impossible to try every value for them; and even less possible to try every single combination of parameter values if the model has multiple values.

Therefore, the most common approach is to define a range of representative values for each hyperparameter. As there are a few values for each parameter, this can be represented as a "grid". During training, the values for each hyperparameter are picked by random from this grid. Usually, a representative number of training and evaluation runs is defined per model. This procedure is commonly known as "hyperparameter tuning" or "hyperparameter optimization".

To efficiently compute many different permutations of hyperparameters, we only use a subset of the total available data. This is possible, as the performance of the RSs techniques is assumed to only improve marginally with more data. This has been shown in academic studies [ZVRF12] as well as production systems [Ama12]. The relation of more data vs. better performance approaches a logarithmic curve, as shown in 6.2. We can run lots of different hyperparameter combinations with a subset of training data. Then, to compare the algorithms (with the discovered best parameters) to each other, we use the total data.

Therefore, we perform hyperparameter training with a subset of the full ratings set. The subset consists of 500.000 user ratings. These ratings subset are ratings that are created by 1171 different users (426 ratings per user in average). They are ratings on 27581 unique games, which represent about 40% of all the games in the full set. This is the data that we use to sufficiently determine the best configuration of the parameters of the recommenders.

For generating metrics evaluation, we need the whole dataset. The full set consists of

TU Bibliothek Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar. WIEN Your knowledge hub The approved original version of this thesis is available in print at TU Wien Bibliothek.

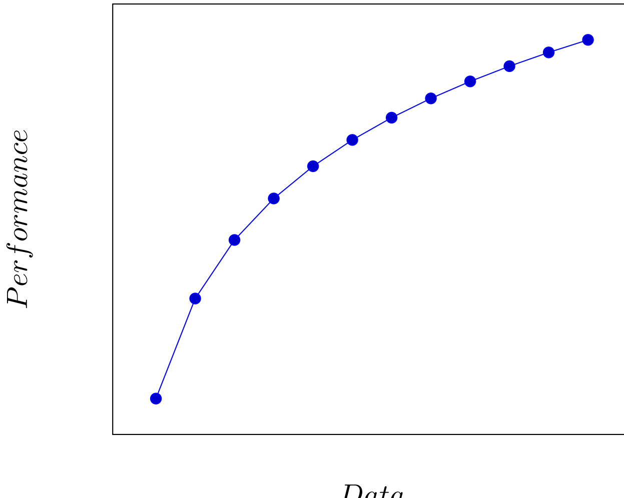


Figure 6.2: Relation of data vs. performance in machine learning applications [ZVRF12]

collected attribute information on 80.474 games in total. There exist over 13 million ratings in total, which are given by around 250.000 users of BGG.

In order to produce training and test set, we partition the existing data. This happens by randomly splitting the ratings of each user. From each user, we take 80% of the data for training and 20% of data for testing. The RS is therefore trained with 80% of the totally available data. Consequently, this means that in the prediction step of the evaluation, 20% of the user's ratings are taken as an input for the trained RS. The other 80% of the users' ratings are then compared to the actual output of the system according to our metrics. We only consider users for training that have a minimum of 200 ratings, in order to draw useful conclusions.

This is how the number of millions of actual existing users that exist on the website BGG shrinks to the comparably minuscule figure of about 250.000 in the full dataset. The vast majority of registered users do not rate any games on the website and are therefore not needed to be included in the evaluation. Similarly, there exist hundreds of thousands of entries for games on BGG. However, a lot of these games might never get any commercial appeal or are never printed in high numbers.

Quite a few of these games are also more "passion projects" by enthusiast board game developers and therefore never amass a high number of ratings. Having a RS recommending items that cannot be bought or easily obtained does not make much sense. These kind of games are automatically omitted from the data set by introducing the lower bound limit of 200 ratings.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

6.4 Discussion of Results

In the following, we present the results of the evaluation. We use the mentioned metrics and discuss the performance of each technique in regard to all others. The results of all metrics for each RS are presented in Table 6.1. The best results for each evaluation metric are highlighted and conclusions why certain models performed better than others will be mentioned.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Recommender Type	***Collaborative Filtering***			***Autoencoder (AE)***			***Content***		***Hybrid***	***Base-line***
Rec. Name → Evaluation Type ↓	Memory (User)	Memory (Item)	Matrix Fact.	Classic	Den. AE	Var. AE	kNN	IDF	Hybrid	Popu-larity
Precision at 5	0.016	0.398	0.689	0.691	**0.697**	0.692	0.069	0.115	0.696	0.548
Precision at 10	0.014	0.379	0.628	0.640	**0.651**	0.648	0.068	0.106	0.639	0.544
Precision at 20	0.012	0.303	0.551	0.574	0.582	**0.594**	0.065	0.100	0.567	0.506
Precision at 100	0.011	0.130	0.364	0.404	0.408	**0.418**	0.040	0.072	0.403	0.361
Recall at 5	0.000	0.011	**0.019**	**0.019**	**0.019**	**0.019**	0.002	0.003	**0.019**	0.015
Recall at 10	0.001	0.020	0.034	0.035	**0.036**	**0.036**	0.003	0.005	0.035	0.030
Recall at 20	0.001	0.032	0.059	0.062	0.063	**0.064**	0.006	0.010	0.061	0.055
Recall at 100	0.006	0.066	0.184	0.206	0.209	**0.214**	0.019	0.034	0.206	0.188
NDCG at 5	0.002	0.129	**0.339**	**0.339**	**0.339**	0.317	0.024	0.046	**0.339**	0.244
NDCG at 10	0.002	0.138	0.337	**0.340**	**0.340**	0.316	0.024	0.046	0.339	0.259
NDCG at 20	0.002	0.133	0.335	**0.337**	0.332	0.319	0.026	0.049	0.335	0.264
NDCG at 100	0.003	0.108	0.331	**0.344**	0.343	0.337	0.028	0.057	**0.344**	0.284
MAP	0.023	0.017	0.017	0.022	0.022	**0.073**	0.000	0.005	0.021	0.006
Diversity	0.067	0.068	0.087	0.085	0.086	0.084	0.029	**0.110**	0.085	0.083
Novelty	0.076	0.075	0.084	0.083	0.084	0.082	0.063	**0.109**	0.083	0.081

Table 6.1: Evaluation results. Columns indicate the various recommenders. Rows indicate different metrics. Highest results per metric (row) are highlighted.

6.4.1 Baseline: Popularity Recommendations

We use popularity recommendations as a baseline. This RS returns the overall most popular games, regardless the input. These games are sorted by the number of ratings. As we see, the evaluation metrics perform rather well for such a simple approach. This is easily explainable: The most popular games are generally well rated, and lots of users have these games in their collection.

Now, we go through the evaluations of all just discussed techniques. In order to be a recommender of good quality, the results for Precision@k, Recall@k, NDCG@k and MAP should be higher than the presented picks by popularity. Increases in Diversity and Novelty are nice-to-have.

Our baseline beats the content and memory based RS in terms of Precision@k, Recall@k and NDCG@k. The degree of outperformance varies and we make a more detailed comparison with this baseline for each recommender in the following.

6.4.2 Memory Based Collaborative Filtering

User - User based

All of the metric results for User - User based collaborative filtering seem to be at the bottom of the range of performance. Only in terms of Diversity and Novelty it gives results that are comparable to other approaches, which is what this type of recommender is also good for (predicting obscure, unknown games).

Item - Item based

This recommender gives us way better results, which are quite remarkable for the simplicity of its principles. Precision@k vastly improved and is 0.398 for k=5 and still 0.303 for k=20. For k=100 it deteriorates to 0.130. Recall@k improved compared to the User based approach and is the best for k=100 at 0.066. Compared to all our approaches, these results are still mediocre as we see in the following. The NDCG@k results are the best after all the Autoencoder and Matrix Factorisation approaches. They also did not beat the popularity baseline for NDCG@k. Overall, the MAP is the lowest among collaborative approaches due to the bad results for recall. Diversity and Novelty are higher than in the previous approach.

6.4.3 Model Based Collaborative Filtering - Matrix Factorisation

Using factorized matrices seems to yield an overall improvement in our metrics. Precision@k and Recall@k show results that are among the best for all our approaches, only beaten slightly by the Autoencoders. The advantage compared to the previous user- or item-based approaches is, that it is always possible to predict ratings for all items in the dataset, even if only few users rated it (the user approach only takes ratings contained in a certain user neighbourhood; the item-based approach can only calculate coorelation of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

existing ratings). This is due to the use of latent user and item vectors. The autoencoders have a similar approach, but their hidden layers seem to be an even better fit for modeling latent relationships in the ratings, explaining their slight evaluation improvements.

Adding to that, NDCG is comparably better than in the previous approaches. The Autoencoders have comparable NDCG@k and it is better than the popularity baseline. MAP seems also to be mediocre indicating a big Recall/Precision tradeoff.

The factorised approach also seems to improve diversity and novelty of recommended sets, as they yield the second best result among our recommenders, together with the autoencoders.

6.4.4 Basic Autoencoder

The Autoencoder techniques exhibit the best results overall. They are comparable to those of Matrix Factorisation, but a notch better. Precision@5 has the highest value with 0.691, dropping only slightly with Precision@20 at 0.574. Precision@100 is still 0.40, meaning that, in average, across all users, 40 of the recommended 100 games are indeed games the users rated well (without counting the good games used as input).

Recall@k is comparable to Matrix Factorisation, but even here, there's an improvement. Recall@100 is higher with 0.206, which means that, in average, the Basic Autoencoder manages to find a fifth of all the well-rated games of a user when computing a recommendation set of 100 games based on a fraction of the user's other ratings.

NDCG@k performed similar as in Matrix Factorisation and has the best values together with the other Autoencoders. NDCG@5 is 0.340 and is about the same for all k. Diversity and Novelty are also above average, as hinted in the previous section.

6.4.5 Denoising Autoencoder

The slightly noisy input that is characteristic of the Denoising Autoencoder doesn't change evaluation drastically. Precision@k and Recall@k results improve only by a very small margin. NDCG@k, Diversity and Novelty stay relatively the same as in the Basic AE.

6.4.6 Variational Autoencoder

The changes from Denoising AE to Variational AE are similar to the step up from Basic to Denoising AE. This is especially observable in the MAP, giving us the best result overall with 0.073.

NDCG@k is, again, among the highest overall. It is slightly worse than in the other Autoencoders. It is worth mentioning that it seems to slightly increase with increasing k.

The improvements for Precision@k and Recall@k for k=20 and k=100 are still small, but slightly bigger than previously, making them the best results among all our recommenders. Interestingly, Diversity and Novelty increased slightly too.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

6.4.7 k-Nearest-neighbor

Coming to the content based recommenders, it was expected that Precision and Recall would fare worse than in the ratings-based approaches. In the following, we debate the usefulness of content-based approaches and possible advantages.

The kNN approach yields only mediocre results for the metrics Precision, Recall, MAP and NDCG. Additionally, Diversity and Novelty are remarkably low. Considering that the similarity is computed based on the attributes, this is not surprising.

These RS could be still interesting to be used as part of a recommendation process. They offer thematically similar, but unknown recommendations that not a lot of users might be aware of.

6.4.8 IDF-based recommender

This type of RS is based on utilising the inverse document frequency measure. It outperforms kNN overall based on our metrics. Precision@k, Recall@k and NDCG@k performance is 50% to 100% better here, MAP improved a bit.

On the other hand, Diversity and Novelty are quite notable here, yielding the best results across all recommenders. This is explained by the definition of IDF itself: Rare categories get a bigger importance, increasing overall variety.

6.4.9 Hybrid Autoencoder

The results of the hybrid approach turned out to be nearly identical to the other autoencoders we investigated. It mirrors Precision@k, Recall@k, NDCG@k and MAP values of the standard autoencoder. This shows that ratings are a more expressive indicator of good recommendations (based on our metrics). Just including similar attributes does not necessarily have added value when ratings are already involved. The results of the RS are, overall, still one of the best, but the feature extraction of the attributes involved seems like unnecessary overhead.

6.5 Summary of Results

It is enough to take a quick glance at the evaluation table (Figure 6.1) to see that, overall, the Autoencoder approaches return the best evaluation results. It is fairly evident that in terms of Precision, Recall, MAP and NDCG, the Autoencoder approaches get the best results. Depending on the metric, the results for Classic Autoencoder, Denoising AE and Variational AE variate a bit. A substantial difference is evident in the MAP metric, where the Variational AE has a big advantage compared to the other recommenders, and even the other Autoencoders. We conclude that the Variational AE is the most promising kind of AE, confirming recent research [LKHJ18].

However, we also notice that other approaches work well considering the context. Considering what the goals of the recommendations are, other approaches seem to be more

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

useful for recommending board games. For recommending diverse and novel sets, our diversity and novelty metrics indicate that it would be commendable to use the IDF recommender.

Additionally, we remarked that a hybrid approach did not give us an advantage in our case. However, even if the hybrid recommender did not outperform the other models in terms of metrics, the results are not terrible. This might be due to the influence of the ratings-based Autoencoder “carrying over” the good results.

As for memory-based recommendation approaches, we can disregard them entirely as they did not measure up to the baseline of popularity-based recommendations. Contrary to that, the results of the matrix factorisation approach seem to be solid. The question is, if this model might be more computationally efficient in application. Then a industrial application might be reasonable. This would highlight other possible research topics.

In the next chapter, we present a summary of this thesis and present possible future research questions.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

CHAPTER 7

Summary & Future work

7.1 Summary of this thesis

In this thesis, we have discussed possible methods to build recommendation systems for board games.

We started by presenting our approach and stating our motivation. We then discussed how far recommendation systems have come and showed the most popular and most well known systems. We also gave examples for appropriate libraries that can be used.

We moved on to discuss the history of board games and showed that due to their rapid growth and huge selection it would be a good application of recommendation systems. We also demonstrated how we were able to build a huge data set that served us as a starting point. In the next section, we discussed the RSs we chose for recommending board games and discussed specifics of their implementation.

Finally, we were able to conduct an evaluation of their performance. For this end, we presented certain metrics that would let us do a meaningful comparison. We ended with a discussion of the evaluation results and drew conclusions.

7.2 Answered Research Questions

With our work, we were able to find answers to our research questions.

- **Q1**: Which recommendation methods are appropriate for recommending board games?

  A1: The collaborative methods outweighed the content-based methods by far. Out of these, we have found the Autoencoder approach to work the best. Out of the

Autoencoders, the Variational Autoencoder gave us a bit of an advantage over the others.

The content based recommenders did not deliver great results at all. Only the IDF recommender surpassed the results of a naive popularity based recommender.

- **Q2**: Can attributes be used for recommendations?

  A2: Yes, and we showed the possibility with our IDF and kNN recommender. The IDF recommender's result were the best among attribute-based methods. It showed that weighting attributes of games to their relative occurrence is successful in modeling the taste of a user.

  While we showed that they certainly *can* be used to this end, we want to emphasize that according to our evaluation results they might not be the most *effective* way to do so.

- **Q3**: Which metrics can be applied for measuring the quality of board game recommendations?

  A3: The standard metrics that are commonly used in RS research gave us good and discernible results. These were: Precision@k, Recall@k, Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG). We also introduced Diversity and Novelty as additional metrics to describe the quality of recommendations.

7.3 Opportunities for future research - open issues

It is possible to build on the results of the thesis and extend it further. Possible research question for future endeavors could be:

- **Hybrid recommendation approaches**

  We presented and evaluated one method to combine different RS together. Needless to say, there are many more possibilities to accomplish this. An additional research opportunity would therefore be to expand on this, merging the techniques of two or more approaches and see how the results variate.

  In our hybrid approach, we extracted content based features for every user and added the respective ratings. Another approach to hybrid recommendation techniques would be to use weighted averages. The output of each RS is considered to have a different weight attached to the recommendation results. The best ratio could be found by testing varying degrees of distribution in multiple runs, for example 90-10, then 80-20, and so on - similar to Hyper-parameter tuning.

- **In-depth survey or user study** Instead of using existing ratings for evaluation, a meticulous case study could be done with real people. Their backgrounds, habits and personal preferences can be registered beforehand. This approach would allow

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the users to fill out a very detailed survey about the recommendations they receive. It could highlight a lot of subjective impressions that other evaluation methods miss. This would be most feasible if building a product and delivering recommendations over a user-friendly user interface. We highlight this research possibility below as well.

- **Evaluation of other metrics**

  In this paper, we have only taken into account the goodness of the predictions in our evaluation. All our metrics dealt with different ways of how the generated ratings and the recommended games could be perceived by the user. Indeed, there are some other metrics that can also be used to measure this, including variations and modifications of metrics we did use in our research.

  However, there are other factors that influence the user experience when utilising a recommendation engine. One example would be the interface used, if the recommendation engine is packaged into a presentable product for the user.

  Another interesting key factor for the success of a recommendation model would be its computational characteristics. This could be the speed with which the model generates recommendations, how much memory or disk space it utilizes or how many resources it needs to function properly. One system might be preferable to another just due to the fact it is more computationally efficient, even if the more inefficient one produces marginally better recommendations.

- **Additional recommenders**

  It is always a possibility to investigate other forms of RSs. On the one hand, there exist other variations of the recommendation techniques we presented in this paper. In the most cases, it's just minor technical differences that distinguish them, for example different configurations or other optimizer algorithms.

  On the other hand, there exist entirely different recommenders that one might want to investigate. We focused in this research paper on the models that are the most commonly used, and the most promising to our knowledge. However, other models might be interesting to investigate as well.

- **Build a product**

  Using our RSs, it would be possible to build a product for users. For example, a web application where users can get game recommendations. This would introduce range of research points: What information should be displayed, what should the User Interface look like and how do users interact with it. Lessons can be learned from existing products, but they should also be reviewed and scrutinized.

  A user study should be conducted in this case to evaluate the resulting product. The final product can be compared to the ease of use of existing solutions, for example the interface provided by Amazon or Netflix. Even if they concern different types of media, this study would only concern the user-friendliness of the interaction.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

However, this topic could span a whole new research paper and would be belong more to the topic of human computer interaction.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Figures

1.1 Released games per year 1900 - 2010. Left: Linear Scale, Right: Logarithmic Scale [Vat17] 2

2.1 We want to predict movie ratings for user u_n (yellow). We see that u_n shares similar ratings on items i_2 and i_4 with u_1 and u_3 (highlighted blue). Consequently, these are the users that are taken for predicting the ratings of items i_1 and i_3 - items that u_n has not rated yet (highlighted red). User u_2's ratings are not very similar to u_n, therefore, u_2 does not appear in the neighborhood of u_n. 10

2.2 R is original user-item ratings matrix. It is sparse, since usually every user only rates a handful of all existing items. The reduced dimensionality k is 3 in this example. P and Q represent the reduced matrices of users and items. These are derived by applying an algorithm like SVD. When P and Q are multiplied, we get a matrix R'. It includes derived ratings for all items by all users. 11

2.3 When we only use the error term during learning, the problem emerges that the model is tailored too much after the training data and can generalize badly (right image). Introducing a regularization term that depends on the parameter can improve generalization (middle image). The impact of this term can be controlled by a factor and should not be exaggerated, otherwise the model might not be accustomed at all (left image) [Ng08]. 12

2.4 When performing SGD, we initialize random parameters for P and Q and calculate the cost (1). The negative gradient is added to the parameters, which decreases the cost (2). After a certain amount of iterations, a minimum value is reached (3). 13

2.5 High level architecture of a CBRS: Features are extracted from items via a content analyzer. User are represented in user profile. Based on the feedback of an active user, the profile learner adjusts this user's profile. The filtering component matches users to fitting items and composes a list of recommendations [RRS15]. 14

2.6 Due to the co-occurrences of the terms *glass* and *wine* with the term *beer* in documents c1, c3, c4 and c8; the term-context matrix shows them as being related to each other [RRS15]. 17

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

2.7 Example of an autoencoder. The encoder part converts input data (blue circles) to a more compact representation (red circles). The decoder processes it back into the original representation (green circles) 18
2.8 A depiction of a Collaborative Denoising Autoencoder. The red links are the user specific weights V_u. The other links are shared and user independent [WDZE16]. 20
2.9 Like a normal Autoencoder, the Variational Autoencoder possesses an arbitrary number of hidden layers. However, at the core, there is a layer that transforms the inputs to parameters of a normal distribution, then transforms the sampled distribution back through one or multiple dense neural layers. 20

3.1 Examples of historic board games. From left to right: *Senet* with game pieces found in the tomb of Tutankhamun (1332 B.C.E.), from [Wik18b]; *The Game Of The Goose* (1880s), from [Wik18a]; Gaming pieces of the *Kriegsspiel* by Lieutenant Georg Leopold, Baron of Reiswitz (1812) from [Hil00]. Modular game pieces like the ones depicted are a mechanic featured in lots of contemporary games. 24
3.2 New releases of Board Games per year since 1930 with blue lines indicating certain industry-shaping events. 25
3.3 Top: Histogram of the most common mechanics in board games. Bottom: Histogram of the most common categories. 27
3.4 Chess enthusiasts watch Garry Kasparov (lower right) play his final game against the Deep Blue Computer, New York, 11 May 1997, from [Sta18] . 28
3.5 A screenshot of the BGG web site with an example game selected. Users can easily see average rating, rankings, categories, complexity and more details on this game (site accessed October 2018) [DSb]. 29
3.6 Number of total ratings per user in the dataset. More than 85% of users rated more than 50 games, giving a healthy data set for recommendation purposes. 30
3.7 Histogram of game ratings per user. It is more likely that users give ratings ranging from 6 to 9. 30
3.8 Board game collection process view. The API does not allow to iterate directly over users IDs, but game IDs. This is why we start with iterating over the games: The top process “Collect Board Games” loops over all existing board game IDs and extracts descriptive attributes. At each game, we also loop over all associated comments. This is how we collect user information and is shown in the blue box “Collect Users - Step wise Aggregation” on the lower right. The user ID is associated with the comments, which is why we can iterate now over the ratings and comments from each user by sending requests with user IDs. The data is saved in a separate database. 31
3.9 Average ratings of games in the years 1960 - 2018 with standard error bars. 32

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Wien Bibliothek Your knowledge hub

3.10 The data model of the collected board games and users. Users are associated to their respective ratings. A game contains all critical information and has many-to-many relations to other tables containing information to categories or mechanics in order to save storage space. 33

4.1 The sigmoid and ReLu functions, often used as activation functions in neural networks. The sigmoid function maps inputs to a range of [0,1]; the ReLu function maps inputs to a range of [-1, 1] 40

4.2 The encoder part of the variational autoencoder. It features a hidden layer for encoding the mean and another for the variance. 41

4.3 Correlation of mechanics in board games. The same mechanics are listed horizontally and vertically. A green field indicates that mechanics are more likely to appear together, a red field implies they are more likely to not occur together. 43

4.4 The hybrid autoencoder we tested. It uses information about preferred mechanics, categories and other attributes (orange) in addition to the ratings of each user (blue). The intermediate layers (red, purple) are a reduced representation which is trained to match the output layer, consisting of the game ratings (green). 46

5.1 Utilising Surprise's built-in algorithms and datasets, it is very easy to run a RS. In this example, we load a subset of the movielens dataset and then instantiate a SVD algorithm. It is then cross-validated five times and evaluated using RMSE [Hug17]. 48

5.2 This code generates a sequential prediction model in Spotlight. It uses a randomly generated sequence of user interactions as an input and splits it into training and test data. The trained model is evaluated with Precision and Recall scores. 49

5.3 Recommenders in RecDB are created using the "CREATE" command. The "ON" clause defines what table is used as the source of user input. The user can specify the columns used for users, items and ratings with the "USERS", "ITEMS" and "EVENTS" keywords respectively. The "USING" clause finally selects which type of recommendation algorithm should be used for calculating recommendations. 50

5.4 An example recommendation pipeline in LibRec, starting from the input Data, processing it to enable training of the recommender all the way to getting the recommendation result and evaluating them [GZSYS15]. 51

5.5 Training a Matrix Factorisation model with MyMediaLite. 52

5.6 The input for SMORe recommenders is a network graph. In this example, we have music titles, artists and songs. The framework generates embeddings for each separate entity [CYCT17]. 53

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek WIEN Your knowledge hub

5.7 In the middle orange box, this figure shows the capabilities and feature offerings that Amazon Personalize includes. The user has the possibility to choose a recommendation model, configure used hyperparameters and run optimizations. On the left we have the input of the recommender systems. These can stream input data from an app directly to refine the recommenders in real time. Another way is to use data at rest in S3 (AWS' cloud storage offer). The output of Amazon Personalize on the right side makes it easy to integrate recommendations into user interfaces. 54
5.8 How to use Amazon Personalize to generate item recommendation. First, you have to create a solution by specifying an algorithm to use. If none is specified, as in this example, AWS will choose one for the user. By creating a solution version, the model is trained. This is used in the second step when creating a pointer to the solution campaign. In the third step, the campaign is used to query recommendations for a user. It is important to note that these 3 steps would be usually called in different parts of the application, depending on the use case. 55
5.9 How to use Recombee's provided library to update information on users and get recommendations. The piece of code shows how to get a new connection to a database stored at Recombee. Then, updates to user views can be made. This is possible as Recombee allows for real-time modification of its algorithms. Then, recommendations can be fetched and presented to the user. 56
5.10 An example of a Jupyter Notebook instance running in a web browser while working with code in Python. 58
5.11 Printing "Hello, World!" to the console output using Java 8. 59
5.12 In contrast, printing "Hello, World!" using Python 3.7 to the console only takes one line. 60
5.13 A function that calculates the factorial of an input *f*. It uses *result* as a temporary variable to store intermediary computation results and returns it when the iteration is done. 61
5.14 This figure shows a text file with the name requirements.txt. It contains all packages that must be installed for the project in which this file is contained to run correctly. It is possible to specify a version number, but not obligatory. When specifying a version number, one can be specific or give minimum or maximum version specifications. In the last line, we feed this file to the "pip install" command. 62

6.1 An example of a Precision-Recall curve. [Ber16] 68
6.2 Relation of data vs. performance in machine learning applications [ZVRF12] 71

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

List of Tables

6.1 Evaluation results. Columns indicate the various recommenders. Rows indicate different metrics. Highest results per metric (row) are highlighted. 73

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[Ama12] Xavier Amatriain. Netflix recommendations: Beyond the 5 stars. Online, 2012.

[BCN16] L. Bottou, F. E. Curtis, and J. Nocedal. Optimization Methods for Large-Scale Machine Learning. *ArXiv e-prints*, June 2016.

[BCT11] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender systems handbook*, pages 299–331. Springer, 2011.

[Bec64] Joseph Becker. The medlars project. *ALA bulletin*, 58(3):227–230, 1964.

[Ber16] Andy Berger. Precision-recall curves. Online, 2016.

[Biz09] Christian Bizer. The emerging web of linked data. *IEEE intelligent systems*, (5):87–92, 2009.

[Boa18] LLC ("BGG") BoardGameGeek. Xml api terms of use, 2018. [Online; accessed 26-October-2018].

[Boo15] Paul Booth. *Game play: paratextuality in contemporary board games*. Bloomsbury Publishing USA, 2015.

[BP97] Daniel Billsus and Michael Pazzani. Learning probabilistic user models. In *UM97 Workshop on Machine Learning for User Modeling*. Citeseer, 1997.

[BRG07] Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. Clustering short texts using wikipedia. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 787–788. ACM, 2007.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[C+15] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[Car65] Jane Carson. *Colonial Virginians at play*. University of Virginia Press, 1965.

[CAS16] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

[CDVdV16] Walter Crist, Anne-Elizabeth Dunn-Vaturi, and Alex de Voogt. *Ancient Egyptians at Play: Board Games Across Borders*. Bloomsbury Publishing, 2016.

[Che16] Jim X Chen. The evolution of computing: Alphago. *Computing in Science & Engineering*, 18(4):4–7, 2016.

[Chi96] Shirish Chinchalkar. An upper bound for the number of reachable positions. *ICGA Journal*, 19(3):181–183, 1996.

[CHV15] Pablo Castells, Neil J. Hurley, and Saul Vargas. *Novelty and Diversity in Recommender Systems*, pages 881–918. Springer US, Boston, MA, 2015.

[CMK66] Cyril W Cleverdon, Jack Mills, and Michael Keen. Factors determining the performance of indexing systems. 1966.

[CTLY16] Chih-Ming Chen, Ming-Feng Tsai, Yu-Ching Lin, and Yi-Hsuan Yang. Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 79–82, New York, NY, USA, 2016. ACM.

[CVW11] Pablo Castells, Saúl Vargas, and Jun Wang. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. 2011.

[CWTY19] Chih-Ming Chen, Chuan-Ju Wang, Ming-Feng Tsai, and Yi-Hsuan Yang. Collaborative similarity embedding for recommender systems. *CoRR*, abs/1902.06188, 2019.

[CYCT17] Chih-Ming Chen, Yi-Hsuan Yang, Yian Chen, and Ming-Feng Tsai. Vertex-context sampling for weighted network embedding. *arXiv preprint arXiv:1711.00227*, 2017.

[CZJL15] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):2, 2015.

[DDF+90] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[DSa] Scott Alden Derk Solko. Boardgamegeek main site.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[DSb] Scott Alden Derk Solko. Boardgamegeek sub page about "catan".

[Duf14] Owen Duffy. Board games' golden age: sociable, brilliant and driven by the internet. *The Guardian*, 2014.

[EGM08] Ofer Egozi, Evgeniy Gabrilovich, and Shaul Markovitch. Concept-based feature generation and selection for information retrieval. In *AAAI*, volume 8, pages 1132–1137, 2008.

[ELKR11] Michael D Ekstrand, Michael Ludwig, Joseph A Konstan, and John T Riedl. Rethinking the recommender research ecosystem: Reproducibility, openness, and LensKit. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 133–140. ACM, 2011.

[F+07] Irving L Finkel et al. Ancient board games in perspective. In *British Museum Colloquium*, 2007.

[GBCB16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[Ger06] Robert S Gerber. Mixing it up on the web: Legal issues arising from internet" mashups". *Intellectual Property & Technology Law Journal*, 18(8):11, 2006.

[GNOT92] 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.

[GP17] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[GRFST11] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. MyMediaLite: A free recommender system library. In *5th ACM International Conference on Recommender Systems (RecSys 2011)*, 2011.

[Gri17] Milton Griepp. Hobby games market over $1.4 billion in 2016, 2017. [Online; accessed 26-April-2018].

[Guo14] Philip Guo. Python is now the most popular introductory teaching language at top us universities. *BLOG@ CACM, July*, page 47, 2014.

[GZSYS15] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. In *UMAP Workshops*, volume 4, 2015.

[Hil00] Philipp Hilgers. Eine anleitung zur anleitung. das taktische kriegsspiel 1812-1824. pages 59–77, 01 2000.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM TRANSACTIONS ON INFORMATION SYSTEMS*, 22:5–53, 2004.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[HSRF95] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[Hug17] Nicolas Hug. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, May 2007.

[Jiz00] Richard Jizba. Measuring search effectiveness. *Creighton University Health Sciences Library and Learning Resources Center*, 2000.

[JOP+] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed <today>].

[JZFF10] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*. Cambridge University Press, 2010.

[Kar90] Jussi Karlgren. An algebra for recommendations: using reader data as a basis for measuring document proximity, 1990.

[KB14a] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KB14b] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[Kor09] Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81:1–10, 2009.

[Kul15] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. In Toine Bogers and Marijn Koolen, editors, *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org, 2015.

[Kul17] Maciej Kula. Spotlight. `https://github.com/maciejkula/spotlight`, 2017.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LH14] Dokyun Lee and Kartik Hosanagar. Impact of recommender systems on sales volume and diversity. 2014.

[LKHJ18] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. *arXiv preprint arXiv:1802.05814*, 2018.

[LM07] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. *CoRR*, abs/cs/0702144, 2007.

[LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[McG] Jesse McGatha. Bgg xml api2.

[MCK06] Frank McCarey, Mel O Cinneide, and Nicholas Kushmerick. Recommending library methods: An evaluation of the vector space model (vsm) and latent semantic indexing (lsi). In *International Conference on Software Reuse*, pages 217–230. Springer, 2006.

[McK10] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[Mil95] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Mit15] Ryan Mitchell. *Web scraping with Python: collecting data from the modern web.* " O'Reilly Media, Inc.", 2015.

[MN+98] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[Moo50] Calvin N Mooers. *The theory of digital handling of non-numerical information and its implications to machine economics*. Number 48. Zator Co., 1950.

[Mur52] Harold James Ruthven Murray. *A history of board-games other than chess.* Clarendon press, 1952.

[Ng08] Andrew Ng. Stanford cs229 - machine learning - ng. 2008.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[NP12] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250, 2012.

[Oli06] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[OTTK12] Ante Odic, Marko Tkalcic, Jurij F Tasic, and Andrej Košir. Relevant context in a movie recommender system: Users' opinion vs. statistical detection. *ACM RecSys*, 12, 2012.

[Par99] David Sidney Parlett. *The Oxford history of board games*. Oxford University Press, USA, 1999.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 701–710, New York, NY, USA, 2014. ACM.

[Pet12] Jon Peterson. *Playing at the world: A history of simulating wars, people and fantastic adventures, from chess to role-playing games*. Unreason Press San Diego, 2012.

[PG07] F. Perez and B. E. Granger. Ipython: A system for interactive scientific computing. *Computing in Science Engineering*, 9(3):21–29, May 2007.

[PPB15] Nikolaos Pappas and Andrei Popescu-Belis. Combining content with user preferences for non-fiction multimedia recommendation: a study on ted lectures. *Multimedia Tools and Applications*, 74(4):1175–1197, 2015.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[R+16] Francisco J.R. Ruiz et al. Boardgamefinder.net. `https://www.boardgamefinder.net/#ourpaper`, 2016.

[RFGST09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[RIS+94] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[RRMB16] Maja Rudolph, Francisco Ruiz, Stephan Mandt, and David Blei. Exponential family embeddings. In *Advances in Neural Information Processing Systems*, pages 478–486, 2016.

[RRS15] F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook*. Springer US, 2015.

[RV97] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[Sah05] Magnus Sahlgren. An introduction to random indexing. 2005.

[Sch09] Barry Schwartz. The paradox of choice, 2009.

[SCJ11] Andrzej Szwabe, Michal Ciesielczyk, and Tadeusz Janasiewicz. Semantically enhanced collaborative filtering based on rsvd. In *International Conference on Computational Collective Intelligence*, pages 10–19. Springer, 2011.

[SG09] Guy Shani and Asela Gunawardana. Evaluating recommender systems. Technical report, November 2009.

[Sha88] Claude E Shannon. Programming a computer for playing chess. In *Computer chess compendium*, pages 2–13. Springer, 1988.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[SLBdG09] Giovanni Semeraro, Pasquale Lops, Pierpaolo Basile, and Marco de Gemmis. Knowledge infusion into content-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 301–304. ACM, 2009.

[SM71] Gerard Salton and M McGill. The smart retrieval system—experiments in automatic document retrieval, 1971.

[SM95] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[SMMA17] Mohamed Sarwat, Raha Moraffah, Mohamed F Mokbel, and James L Avery. Database system support for personalized recommendation applications. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 1320–1331. IEEE, 2017.

[SMR08] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Evaluation in Information Retrieval*, volume 39, chapter 8, pages 151–175. Cambridge University Press, 2008.

[SMSX15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

[Sta18] Stan Honda,AFP, Getty Images. Man versus machine: chess enthusiasts watch garry kasparov in his final match against deep blue, new york, 11 may 1997., 2018. [Online; accessed October 26, 2018].

[TP10] Peter D Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of artificial intelligence research*, 37:141–188, 2010.

[TPNT07] Gabor Takacs, Istvan Pilaszy, Bottyan Nemeth, and Domonkos Tikk. On the gravity recommendation system. In *Proceedings of KDD cup and workshop*, volume 2007, 2007.

[Tur53] Alan M Turing. Digital computers applied to games. *Faster than thought*, 1953.

[Vat17] Dinesh Vatvani. An analysis of board games: Part i - introduction and general trends, 2017. [Online; accessed 26-April-2018].

[Vem05] Santosh S Vempala. *The random projection method*, volume 65. American Mathematical Soc., 2005.

[VLBM08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[VRDJ95] Guido Van Rossum and Fred L Drake Jr. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[W+63] Alvin M Weinberg et al. Science, government, and information: The responsibilities of the technical community and the government in the transfer of information. 1963.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU WIEN Bibliothek Your knowledge hub

[WDZE16] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 153–162. ACM, 2016.

[Wik18a] Wikipedia, the free encyclopedia. Goosy goosy gander, 2018. [Online; accessed October 26, 2018].

[Wik18b] Wikipedia, the free encyclopedia. Senet game pieces (tutankhamun), 2018. [Online; accessed October 26, 2018].

[Win14] Nick Wingfield. High-tech push has board games rolling again. *The New York Times*, 2014.

[Woo12] S. Woods. *Eurogames: The Design, Culture and Play of Modern European Board Games*. McFarland & Company, 2012.

[YB16] Denis Yarats and Alberto Bietti. Qmf. `https://github.com/quora/qmf`, 2016.

[YCWT18] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. Hop-rec: High-order proximity for implicit recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, pages 140–144, New York, NY, USA, 2018. ACM.

[Zei12] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[ZVRF12] Xiangxin Zhu, Carl Vondrick, Deva Ramanan, and Charless C Fowlkes. Do we need more training data or better models for object detection?. In *BMVC*, volume 3, page 5. Citeseer, 2012.

[ZYS17] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar.
The approved original version of this thesis is available in print at TU Wien Bibliothek.