Session-based Recommender Systems

Bachelor's Thesis submitted

to

Prof. Dr. XYZ

Humboldt-Universität zu Berlin
School of Business and Economics
Chair of XYZ

by

Grzegorzthehero

in partial fulfillment of the requirements
for the degree of
Bachelor of Science
Berlin, September 21, 2020

Contents

List of Tables

List of Figures

Abstract

Recommender systems aim at assisting users in a huge space of information available online, e.g. media streaming applications or e-commerce. They are usually based on the information from the users long-term interaction with a domain. Nevertheless, this information is not always available due to various reasons. Hence, in the recent years the interest in session-based recommender systems has risen significantly. In this work, we take on a task of thoroughly describing GRU4Rec, KNN-based and factorization-based approaches for session-based recommendation. Later we compare the algorithms' performance - with appropriate evaluation metrics - based on results obtained during the study. The algorithms are trained on three different datasets from music and movie domain: Nowplaying, Yahoo, and Movielens-1M. The results of our experiment suggest a consistent and very good performance of techniques based on deep neural networks. Furthermore, we find that the recently introduced loss functions for GRU4Rec algorithm - Top1-max and BPR-max - perform better than Top1 and BPR losses. The results suggest that the relative performance of algorithms often varies across different datasets what leaves a room for further research in this topic.

1 Introduction

Very often customers are welcomed with a plethora of products available online. Therefore, it is essential to adopt personalization strategy in order to enhance user experience (S. Zhang et al., 2019). One of the prominent examples of value adding services to customers that can be provided on websites using information technology are online recommender systems. They are able to significantly diminish customers' search costs by helping the users to navigate in the overabundance of information and encourage them to purchase more products. Furthermore, by intelligently providing the customers with personalized recommendations it is also possible to encourage some clients to purchase products that they did not intent do buy in the first place (Garfinkel et al., 2006). Hence, the recommender systems appear in many areas of web activities ranging from e-Commerce systems (Huang and Huang, 2009) to entertainment (Melville et al., 2002) and social media (J. Chen et al., 2011).

1.1 Introduction to Recommender Systems

Meeting the user preferences seems to be an inevitable part of delivering positive user experience nowadays. Originally recommender systems were defined as systems that could direct recommendations to proper recipients based on once provided inputs (Resnick and Varian, 1997). More recently, a more general and broader definition has been present in the field, where recommender systems are described as ones that work in a personalized way by guiding users to useful or interesting objects and insights in a complex and large space of potential options (Hernández del Olmo and Gaudioso, 2008).

There exist various approaches to recommendation tasks among which three are considered as the most important: **collaborative filtering** (CF), **content-based filtering** and **hybrid recommendation system** (Singhal et al., 2017).

Collaborative filtering is based on constancy of taste over time. Through collecting and analyzing data on user's activities, behaviour and preferences, it can anticipate which objects will be liked based on similarity shared with other users. It can accurately recommend items like movies and music tracks without even learning its content. One of its key advantages is its simplicity where the question on how to calculate the utility from the item is converted into the task of missing values' extrapolation in the rating matrix (Burke et al., 2011).

On the contrary, the content based filtering techniques are based on the analysis of user's favoured item's content. Building a user profile serves the purpose of determining a preferred item that is also described using keywords. The main idea of this approach is that similar items are probably liked or disliked in the same way. In the last one, hybrid recommendation method, both of the above described techniques are initially implemented separately and then combined.

Despite the fact that previously mentioned approaches are prevalent, they fail to deliver a good prediction in the situations listed below:

– They ignore to take into account the timestamp data of each interaction resulting in inaccurately modelled preferences (Ludewig et al., 2019).

– They rely on data from a longer period of time of user interactions with a domain. However, in a big amount of applications this information is not available due to the fact that sometimes a user appears for the first time on a website, is not logged in or due to privacy concerns (Ludewig et al., 2019).

– Online customers tend to have sometimes very different and individual short-term shopping preferences that do not match general patterns (Jannach et al., 2015).

For those reasons, sequential recommendation, also known as session-based has gained popularity in practical applications and academic contributions.

1.2 Sequential Recommender Systems

The sequential recommenders capture the interaction between a user and an item in a form of dynamic sequence. By taking into account sequential dependencies they are able to capture patterns in both the recent and current user preferences. Thus the job of conventional sequential recommendation systems is to employ one of the appropriate machine learning techniques in order to model sequences (X. Chen et al., 2018). For those kind of recommenders it is of fundamental importance to simultaneously take into account users' short-term preference in order to model it and provide more accurate recommendation. In these kind of situations, approaches that are only based on users' interactions during an ongoing session and are able to adapt to their actions are very convenient and superior (Quadrana et al., 2018).

Due to the raising attention given to sequential recommender systems by academic field and business, numerous algorithms have been developed to solve the session-based recommendation problem. In this study, willing to present and compare various algorithms' applications, we will first summarize the current research state in this field. Then based on reviewed materials we will introduce relevant research questions. Willing to answer them we will conduct an empirical study on three different datasets, which will aim at comparing the performance of applied algorithms. In the end we will briefly summarize the results, which are going to be followed by a conclusion part.

2 Background and Literature Review

2.1 Background

Since its christening in 1995, the field has grown enormously in regard of the number of possible problems that can be addressed and techniques that can be employed (Burke et al., 2011). The expansion of interest in the past two decades was partly catalyzed by the Netflix Prize contest which took place in 2006 (Bennett et al., 2007) in which Netflix published a large and anonymous movie ratings dataset in order to challenge machine learning, computer science and data mining communities to develop a better recommender system that could surpass the Cinematch[1] by more than ten percent in accuracy (“Netflix Prize: Review Rules”, n.d.). Since then the number of research papers and Kaggle competitions skyrocketed, especially in the recent few years, as better performing techniques are being developed (Figure 2.1).

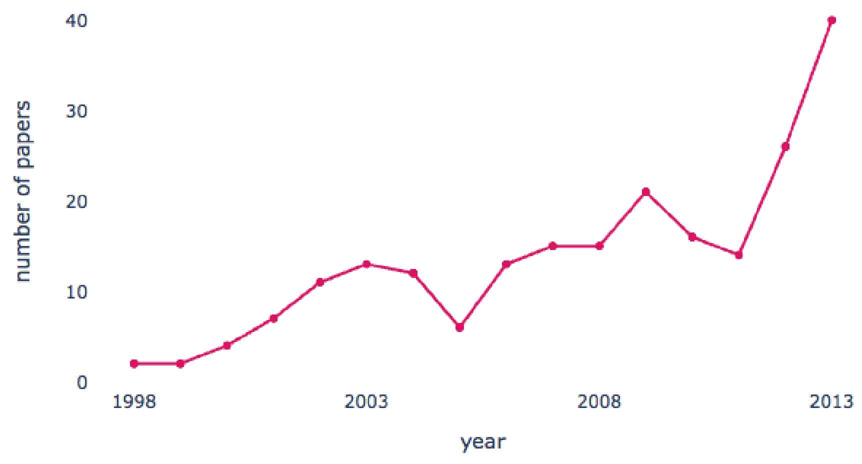


Figure 2.1: Number of research papers on recommender systems in years 1998-2013, adapted from (Beel et al., 2016)

The following studies and articles apply different algorithms and approaches to solve the recommendation task in an optimal way. The below reviewed approaches mostly belong to one of three method groups: K-nearest Neighbors (KNN-based), Recurrent Neural Networks-based (RNN-based) and Factorization-based. Each of the following group is discussed in more detail below.

2.2 KNN-based Methods

A wide application of the KNN approach in comparison to other methods can be ascribed to its robustness, interpretability and efficiency (Bonnin and Jannach, 2014; Ludewig and Jannach, 2018).

Being a widely used machine learning classification technique, KNN can be upgraded and adapted in order to reflect the specific requirements of a recommendation task and to improve the predictive accuracy. One of the examples of such procedure was a new variant of KNN implemented by Subramaniyaswamy and Logesh (2017) to focus on generating recommendation for new users.

[1]previously used Netflix’s recommendation algorithm

In their work they presented an Adaptive KNN (AKNN)-based recommendation system that gave a better results regarding F-Measure, precision, recall and accuracy compared to other associative classification algorithms such as FOIL[2], CBR[3], CPAR[4] and CMAR[5]. The algorithm performs clustering based on the similarity of items and users, known movie ratings, and then it yields a predicted rating as an output. Afterwards, in order to generate the top-n movie recommendation list, AKNN Algorithm is extended to AKNN based CF algorithm which includes the output of the previous algorithm and active target user as input. After the top-n recommendations are initialized as null, the set of expert users, featuring the higher similarity is chosen for the active target user. Based on the distance measure, the items with the nearest neighbors are computed for the user and those with the maximum closeness are then added to the close items set. After aggregation, sorted list of top-n recommendation is generated (Subramaniyaswamy and Logesh, 2017).

It has been only recently that a KNN-based approach was introduced for a session-based recommender engine. Since the advent and rapid increase of use cases of deep learning techniques in recommendation systems field, e.g. RNN-based, KNN-based baseline methods' performance has been often studied in comparison to deep learning models and scored worse in different metrics, for instance in research conducted by Hidasi et al. (2015). Nevertheless, Jannach and Ludewig (2017) argued in their paper that the effectiveness assessment and comparison ability of such novel approaches, e.g. Gated Recurrent Units (GRU) was difficult because no standard evaluation metrics had been applied. Hence, the performance of baselines used for comparison was not so apparent. In order to determine, which of the both recommenders performs better, they were tested on different datasets from e-commerce websites as well as from datasets consisting of music playlists from internet platforms. Their results, proved that a Contextual KNN (CKNN) with cosine similarity measure could compete with RNN-based approaches in a field of session-based recommendations, within four performance protocols: HR@10[6], MRR@10[7], HR@20[8] and MRR@20[9]. Moreover, the best performance was obtained by an ensemble created out of both of the approaches. It indicated that RNN-based models are proficient in capturing sequential patterns that cannot be identified by purely KNN-based approaches (Jannach and Ludewig, 2017).

[2]First-order Inductive Learner

[3]Case-based Reasoning

[4]Classification based on Predictive Association Rules

[5]Classification Based on Multiple Class-ssociation Rules

[6]Hit Rate with the recommendation list of length = 10

[7]Mean Reciprocal Rank with the recommendation list of length = 10

[8]Hit Rate with the recommendation list of length = 20

[9]Mean Reciprocal Rank with the recommendation list of length = 20

Addressing the results of the study reviewed in the last paragraph, Guo et al. (2019) pointed out that, the CKNN approach performed pretty well in session-based recommendation. However it was not able to guarantee the proportion of relevant sessions for distinct items clicked in current session and distinguish the effect of items characterised by different popularity. For the purpose of addressing these limitations they proposed a novel approach called CKNN with Diffusion Based Similarity Method (DSM). Because aiming attention at the most recent event had proven its effectiveness in the fields of news recommendation and e-commerce (Covington et al., 2016), Guo et al. (2019) decided to study the performance of Original algorithm. After conducing the experiment on three different datasets they were able to prove the superiority of their CKNN-DSM(0.5,0.5)-EPCSR[10] algorithm which assumed the same influence of differently popular items and was based on random selection strategy. The proposed algorithm gave the performance improvement on all three metrics: MRR@20, HR@20 and Coverage@20[11] (Guo et al., 2019).

2.3 Session-based Recommendations with RNN

Due to its tremendous achievements in the field of natural language processing and effective processing of sequential data, a deep learning technique - RNNs - have gained increasing popularity in sequential recommender systems (S. Zhang et al., 2019). The superiority of their performance caused an exponential increase in the number of research contributions to recommendation methods based on deep learning (Batmaz et al., 2018). For instance, recent deep learning-based techniques are capable of ingesting all text data end-to-end without performing often costly preprocessing (Zheng et al., 2017). Moreover, without these new advancements it would be impossible to represent text, interactions and images in a unified joint framework (Y. Zhang et al., 2017).

Similarly to learning word sequences, Recurrent Neural Networks (RNN) can solve different problems by considering items as words and its collection as complete vocabulary, a sample sequence being the history of each user. Gated RNN can be considered as a common and state-of-the-art approach with its two comparable architectures: Gated Recurrent Units (GRU) (Cho et al., 2014) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

Quite a handful of e-commerce sites are equipped with recommendation engines. Approximately 35 percent of what was sold in 2013 on Amazon can be ascribed to items proposed by recommender engines (Meyer and Noble, 2013). Hence, there was and still is a need to research on novel

[10]based on random selection and **E**qual **P**robability **C**andidate **S**election (EPCS)

[11]Coverage with the recommendation list of length = 20

recommendation techniques with e-commerce session data. In order to study the performance of recurrent neural networks on a recommendation system Hidasi et al. (2015) applied a version of customized GRU to e-commerce session data. After customizing GRU model and using session-parallel mini batches, the researchers pointed out that relevance-based item rankings are the core of recommender systems, and therefore decided to use pairwise ranking losses: Bayesian Personalized Ranking (BRP) and Top1 that performed well. Apart from e-commerce session data, they additionally evaluated their model on data collected from OTT - a video service platform. The optimized GRU model performed considerably better than IKNN on both datasets and both evaluation metrics Recall@20 and MRR@20 . Since then the introduced approach is often called GRU4Rec (Hidasi et al., 2015).

In their following work Hidasi and Karatzoglou (2018) proposed novel ranking loss functions which are better tailored to recommendation with RNNs. They observed that under certain circumstances, for instance increasing number of samples, the ranking losses introduced in their previous paper (Hidasi et al., 2015) suffered from vanishing gradient problem and thus the learning stopped. To address this issue they introduced a new group of listwise loss functions which originate form pairwise losses. They named the loss functions BPR-max and Top1-max functions. It conveyed the idea that the target score is compared with the sample score that is the most relevant, what was equivalent to having the maximal score amid the samples. The researchers were able to obtain a considerable improvement in their performance metrics, Recall@20 and MRR@20, over classical collaborative filtering method, Item KNN (IKNN), as well as previously applied original GRU4Rec algorithm. Moreover, they were able to show that with their new loss functions the training times did not increase significantly (Hidasi and Karatzoglou, 2018).

There has been plenty of studies concerning various types of recommender systems which have been applied to different and often complex tasks. Similarly to the pace of technological changes, the field of recommender systems has been bustling with innovation. The techniques evolve because there is a need to constantly find the ones whose performance features improved accuracy and could later help to achieve business goals. Some of those problems can be of a very challenging nature. A research conducted for Zalando could be named as one of them. Heinz et al. (2017) had to come up with an approach to recommend fashion items, which are featured inter alia by multiply sizes, changing trends, seasonality and sometimes small stock. In their contribution they started with a static model and later continued by including the information about the time of sale. By employing a static model called Fashion DNA, which was obtained from article data in a form of a dense numerical representation they were able to solve the problem when previously not sold articles were arriving at the store. Moreover, sequences of purchases for a particular customer were fed into neural network and were updated after sales event occured. In order to be able to handle sequential data researchers applied recurrent neural networks (RNN) with long short-term memory (LSTM) cells. They were able to prove their superiority over non-personalized baseline model which calculates the recent fondness for an article into the future and into other static models (Heinz et al., 2017).

In another study Devooght and Bersini (2017) applied GRU architecture with two different objective functions to build a movies-recommending model based on user's session information. The functions helped in finding an optimal solution by providing a method for measuring its quality to a generally mathematical problem. By taking model parameters and data as arguments, it allowed evaluation to return a specific number. The parameters of the model could be tuned in order to find their values which either minimize or maximize this number. In their contribution, they decided to employ two of them: Categorical Cross-Entropy (CCE) and Hinge as objective functions for movie and e-commerce collaborative filtering and compare them with i neighborhood-based approaches. The former is commonly used as a loss function for language modeling task whereas the latter is dependent on support vector machines' objective function. Comparing it to other baseline models like User KNN they were able to achieve superior result applying RNN with categorical cross-entropy as objective function. In the end of their paper they indicated that RNNs results were featured by more diversity and less dependency on me most recurrent items than neighborhood- and factorization-based approaches (Devooght and Bersini, 2017).

2.4 Factorization-based Approaches

For the purpose of overcoming the constraints of models based only on sequence learning a number of hybrid approaches have been proposed in the recent years. They successfully combine sequence modeling methods with the matrix factorization approaches (Ludewig and Jannach, 2018). In one of the earlier studies on this issue Rendle et al. (2010) presented Factorized Personalized Markov Chain (FPMC) method, an approach which brings Markov Chains (MC), for sequential information, and matrix factorization for acquiring user's general taste. The model was originally developed for the domain of e-commerce, where the next basket should be recommended to a user. It applied a transition matrix for each user, which is affected by similar users' transition. The performance of the model was evaluated with precision and recall as well as with Half-life-utility and area under the ROC curve. Based on the aforementioned evaluation metrics the authors were able to show that their approach outperforms matrix factorization, factorized Markov chains and normal MC model (Rendle et al., 2010).

In a number of the later studies this idea was taken further by introducing and successfully applying modified FPMC algorithms in various forms. One of those improvements was Factored Item Similarity Models (FISM) introduced by Kabbur et al. (2013). The approach was based on an item-item factorization. In order to obtain item-item similarity matrix, structural equation modeling method was used. In contrast to FPMC, FISM did not use sequential item-item transitions. The researchers evaluated their model by applying 5-fold Leave-One-Out-Cross-Validation (LOOCV) and performance metrics like HR and weighted version of it: ARHR. Their new approach generated the best performance when compared to other models, inter alia, BPRMF[12] and IKNN with different similarity metrics. Furthermore, the authors argued that the proposed method could produce recommendations of high quality even when sparsity was the issue with increasing performance gap as sparsity became more present (Kabbur et al., 2013).

One of the other hybrid approaches proposed by He and McAuley (2016) - Factorized Sequential Prediction with Item Similarity Models (Fossil) - aimed at taking into account both sequential patterns and long-term user preferences. The method fuses similarity-based approaches, FISM, with sequentially aware factorized Markov chains to anticipate personalized sequential behaviour. The researchers also conducted a study on the influence of data sparsity. The performance of their model was found to be considerably better compared to other approaches, inter alia FPMC, FISM. Moreover, they proved that when the prediction task deals with the sparsity issue, Fossil's performance was especially strong.

Ludewig and Jannach (2018) argued that the aforementioned factorization-based models, in general, had been created to predict the next actions of users but had not been devised for session-based recommendation where the user was anonymous. Nevertheless, in their study they applied these models for the session-based recommendation setting in order to compare a great amount of nowadays existing recommendation models. The performance of those models was not satisfying, with worse performance than other model, e.g. GRU4Rec and CKNN (Ludewig and Jannach, 2018).

[12]Bayesian Personalized Ranking Matrix Factorization method

2.5 Limitations and Research Challenges of Recommender Systems

There are some potential limitations and established problems that are commonly considered when facing recommendation tasks. Several challenges are raised in survey on deep learning based recommender systems prepared by S. Zhang et al. (2019). As one of them they named the interpretability issue, what means that the models often tend to behave as black boxes. They underlined that in the case of deep learning models their non-interpretability of activations and hidden weights limited the explainability. Nevertheless, they argued that apart from interpretation of individual neurons, improved interpretability can be achieved through neural attention models (S. Zhang et al., 2019).

The amount of parameters in deep learning models requires them to be trained on a large datasets. Luckily, according to S. Zhang et al. (2019) it is relatively easy to collect huge amounts of data in the field of recommender systems in comparison to other domains, e.g. vision or language in which there is a scarcity of labeled data. Moreover, big scale datasets have been also released for academic purposes in the recent years, e.g. RSC15[13] and CLEF[14].

Another limitation of deep learning models that may also be considered as a problem of machine learning itself is the necessity of extensive hyperparameter optimization. It requires setting a value to models' parameters influencing the training process. In order to find the optimal hyperparameters models have to be trained and then tested. Additionally, training and then evaluating deep learning models on a large datasets requires a significant computational power and usually takes a considerable amount of time what is often considered as a major drawback and can discourage some from applying these techniques (S. Zhang et al., 2019).

Most of the recommender systems are said to be far form ideal. Schedl et al. (2018) argued that, for instance music recommender systems, are not taking into account multitude of factors on which they depend and are often only focused on one concept. The researchers underlined that apart from interaction information, extrinsic (Gillhofer and Schedl, 2015) and intrinsic (Ferwerda et al., 2015) information should be taken into account. The former corresponds to user's activity at the moment and the latter one to user's emotional state. Moreover, contextual aspects, e.g. social surroundings or the current weather conditions, play an important role in music choice. Only that specific example shows that recent recommenders of any domain should be taking into account more factors in order to provide more accurate recommendations (Schedl et al., 2018).

[13]Dataset published for Recsys Challenge 2015

[14]Dataset published for Newsreel Challenge

For a recommender systems introducing new users and items is one of the major problems because it does not possess valuable data about new items or novel users. For that reason the recommender is not able to properly propose new items to users and recommend for new users the items already existing in the catalog (Schedl et al., 2018). However, Lika et al. (2014) argued that the issue could be overcome. In their contribution they presented a method which was able to alleviate this obstacle by making use of users' demographic data. Then, by applying similarity techniques, they defined their potential neighbors. The idea behind this approach was that it is more possible to share preferences with people who possess similar characteristics and common background. Nevertheless, precise demographic data could be often not available in the real-world applications (Lika et al., 2014).

Alternatively, in order to alleviate the aforementioned problem a recommendation model that takes into consideration only the current user session can be employed. Tan et al. (2016) proved that by applying their upgraded RNN model they could obtain considerable improvement in the accuracy over other commonly used session-based models, e.g. KNN. Furthermore, they were able to prove that by applying popular approaches for improving deep neural networks such as batch normalization, dropout, or data augmentation, they could obtain even better results (Tan et al., 2016).

One of the challenges that has been preset in the recommendation system community already for more than ten years is the fact that most of the researches evaluate their models only with accuracy oriented metrics. McNee et al. (2006) argued that there is a lot beyond only accuracy in recommender systems. They claimed that a recommendation list should be judged based upon its usefulness as a complete entity and user contentment does not always go together with high recommendation accuracy. To address the issue, Ge et al. (2010) presented their new approaches that should entail the recommendation quality presented to a user in a better way. They discuss two measures: *Coverage* and *Serendipity*. The former defines the degree to which recommendations include the set of possible items and extent to which recommendations can be made to all possible users. The latter focuses on the novelty of generated recommendations so that they can surprise a user in a most positive way. The scientists argued that their approach gave additional perspectives on judging the recommendation quality (Ge et al., 2010).

Based on the literature reviewed in this section, we present methodology applied in this study. We include algorithms from all of the three groups described above. The methodology is used in order to answer the research questions presented in the experiment description further in this work.

Table 2.1: The summary of aforementioned models.

Authors	**Title**	**Year**	**Approach**							**Evaluation metrics**						
			RNN	KNN	MC	FISM	FOSSIL	FPMC	other	AUC/ROC	Recall	Coverage	MRR	Pre.	HR	other
Sebastian Heinz, Christian Bracher, Roland Vollgraf	An LSTM-Based Dynamic Customer Model for Fashion Recommendation	2017	x						x	x						
Robin Devooght, Hugues Bersini	Long and Short-Term Recommendations with Recurrent Neural Networks	2017	x	x	x		x	x	x		x	x				x
Blerina Lika, Kostas Kolomvatsos, Stathes Hadjiefthymiades	Facing the Cold Start Problem in Recommender Systems	2013		x					x							x
Yong Kiam Tan, Xinxing Xu, Yong Liu	Improved Recurrent Neural Networks for Session-based Recommendations	2016	x	x					x		x		x			
Balázs Hidasi, Linas Baltrunas, Alexandros Karatzoglou, Domonkos Tikk	Session-based Recommendations with Recurrent Neural Networks	2015	x	x					x		x		x			
Subramaniyaswamy V., Logesh Ravi	Adaptive KNN based Recommender System through Mining of User Preferences	2017		x					x		x			x		
Huifeng Guo, Ruiming Tang, Yunming Ye, Feng Liu, Yuzhou Zhang	A Novel KNN Approach for Session-Based Recommendation	2019		x								x	x		x	
Dietmar Jannach, Malte Ludewig	When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation	2017	x	x					x				x		x	
Balázs Hidasi, Alexandros Karatzoglou	Recurrent Neural Networks with Top-k Gains for Session-based Recommendations	2018	x	x							x		x			
Malte Ludwig, Dietmar Jannach	Evaluation of Session-based Recommendation Algorithms	2018	x	x	x	x	x	x	x		x	x	x	x	x	x
Steffen Rendle, Christoph Freudenthaler, Lars Schmidt-Thieme	Factorizing Personalized Markov Chains for Next-Basket Recommendation	2010			x			x	x	x	x			x		x
Santosh Kabbur, Xia Ning, and George Karypis	FISM: Factored Item Similarity Models for top-N Recommender Systems	2013		x		x			x						x	x
Ruining He, Julian McAuley	Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation	2016			x	x	x	x	x	x						
Together			7	10	4	3	3	4	11	3	7	3	6	3	4	5

3 Methodology

3.1 K-nearest Neighbors

In order to present KNN-based algorithms used for session-based recommendation problems one may find it necessary and useful to explain the basic functionality of original KNN algorithm which belongs to the family of supervised machine learning methods that can be applied to solve both regression and classification problem. It is often praised for it simplicity and for its capability of providing good performance results (Dziuda, 2010, p. 328). It assumes the close proximity of similar items, which is calculated based on a chosen similarity metric inter alia euclidean distance and cosine similarity. The choice of this metric depends on the type of problem one wants to solve and can have the impact on the performance of the model as seen in (Guo et al., 2019). One of the most important issues concerning the KNN approach is to estimate the correct value of k nearest neighbors which influences the algorithm. It directly affects the majority voting process, among the k closest samples, which is performed to determine the class to which a query sample belongs (Shakhnarovich et al., 2006).

3.2 Contextual KNN Approach

Table 3.1: Exemplary session data, adapted from (Guo et al., 2019).

Item id	$\beta 1$	$\beta 2$	$\beta 3$	$\beta 6$	$\beta 3$	$\beta 4$	$\beta 2$	$\beta 3$	$\beta 4$	$\beta 3$	$\beta 5$
Session id	i	i	i	i	j	j	k	k	k	l	l
Time	0	1	2	3	4	5	6	7	8	9	10

Table 3.2: Mapping pair (item id, interaction time) to session id, adapted from (Guo et al., 2019).

Session id	(Item id, interaction time)
i	($\beta 1$, 00), ($\beta 2$, 01), ($\beta 3$, 02), ($\beta 6$, 03)
j	($\beta 3$, 04), ($\beta 4$, 05)
k	($\beta 2$, 06), ($\beta 3$, 07), ($\beta 4$, 08)
l	($\beta 3$, 09), ($\beta 5$, 10)

Table 3.3: Mapping pair (session id, interaction time) to item id, adapted from (Guo et al., 2019).

Item id	(Session id, interaction time)
$\beta 1$	(i, 00)
$\beta 2$	(i, 01), (k, 06)
$\beta 3$	(i, 02), (j, 04), (k, 07), (l, 09)
$\beta 4$	(j, 05), (k, 08)
$\beta 5$	(l, 10)
$\beta 6$	(i, 03)

For the purpose of presenting the CKNN approach in a session-based context [15], Table 3.1 contains example data from four sessions. It illustrates the interactions of session and items in a given time. These interactions can be represented in a form of a **session-item bipartite network** $G = \{S, I, E\}$ where session, item and interaction sets are represented by S, I and E respectively. l is the cardinality of E and the number of items and sessions are depicted as n and m. With $a_{xi} = 1$ if (x,i) $\in E$ and (0 in other cases), the G's adjacency matrix can be displayed as $A \in R^{m \times n}$.

Dictionaries Map_{S2I} and Map_{I2S} map session and items to the pairs of (item, interaction time) and (session, interaction time) respectively, as presented in Table 3.2 and Table 3.3. CKNN recommendation approach consists of five steps, which are described as follows (Guo et al., 2019):

– **Step 0**: User triggers a recommendation by clicking an item.

– **Step 1**: **RL(x)** – a set of relevant sessions associated with items in the ongoing session x is found. For instance, $\{j, k, i\}$ is a set of sessions related to the current session $x = \{\beta_4, \beta_1\}$ (session i is associated with item β_1 and j, k with item β_4).

– **Step 2**: Because concentrating on the last events has proven to be effective in news recommendations and e-commerce most recent sessions k_{recent} from **RL(x)** are selected as the recent session set **RC(x)** (Covington et al., 2016). Selecting the most k_{recent} = 2 would imply **RC(x)** = $\{j, k\}$

– **Step 3**: For the current session x and every session j from RC(x) the similarity score $sim(x, j)$ is calculated. Afterwards, k_{top} – the sessions found to be the most similar are selected and added to a nearest neighbor session set **NN(x)**.

[15]in (Ludewig and Jannach, 2018) also called SKNN

– **Step 4**: Based on NN(x) and $sim(x, j)$, the score of item β for the ongoing session x is:

$$score_{KNN}(\beta, x) = \sum_{j \in NN(x)} sim(x, j) \times 1_j(\beta) \tag{3.1}$$

where $1_j(\beta)$ stands for session b containing item β and otherwise 0. Eventually the k items with the best score are returned as $result_x$. For similarity $sim(x, j)$ four different scenarios are examined: tanimoto, binary, cosine and jaccard.

3.3 GRU4Rec

3.3.1 GRU

RNN have been designed to properly model sequence data of variable-length. One of the main differences of RNN in comparison to other traditional feedforward deep learning models, is the presence of a hidden internal state in the units comprising the network (Hidasi et al., 2015). The following function is utilized to update the hidden state h of the common RNNs:

$$h_t = g(Wx_t + Uh_{t-1}) \tag{3.2}$$

Where g stands for a bounded and smooth function, for instance sigmoid function, and x_t for the unit input at time t. Given the current state of h_t, the output of the RNN is the probability distribution over the following element in the sequence.

A more elaborate, RNN-based, Gated Recurrent Unit (GRU) (Cho et al., 2014) aims at overcoming the problem of vanishing gradient by learning when and how much the hidden state of the unit should be updated. GRU's activation function is the linear interpolation between the preceding and the candidate activation $\hat{h_t}$:

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h_t} \tag{3.3}$$

The update gate is computed by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{3.4}$$

where σ is the logistic sigmoid function. $\hat{h_t}$, the candidate activation function, is given by:

$$\hat{h_t} = tanh(Wx_t + U(r_t \odot h_{t-1}) \tag{3.5}$$

r_t, the reset gate, is computed as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{3.6}$$

3.3.2 Customized GRU

For a session-based recommendation Hidasi et al. (2015) customizes their GRU based model. The actual state of the session is the input of their model, whereas the item of the next session's event is the output. The session state can be either the events in the session or the item of the actual event. The former setting, employs a weighted sum of representations, in which earlier occuring events are discounted. In the latter setting, items are 1-of-N encoded, i. e. the number of items is even with the length of the input vector.

The GRU layers are the core of the network with additional layers added before the output layers. The network's output is given in a form of item's predicted preference, i.e. the likelihood for each item of being the next one in the given session. The hidden state of the preceding layer is utilised as the input of the following layer, in a network architecture with multiple GRU layers. Optional connection of the input to GRU layers which lay deeper in the network improves their performance (Hidasi et al., 2015). The general architecture of their network is depicted in Figure 3.1.

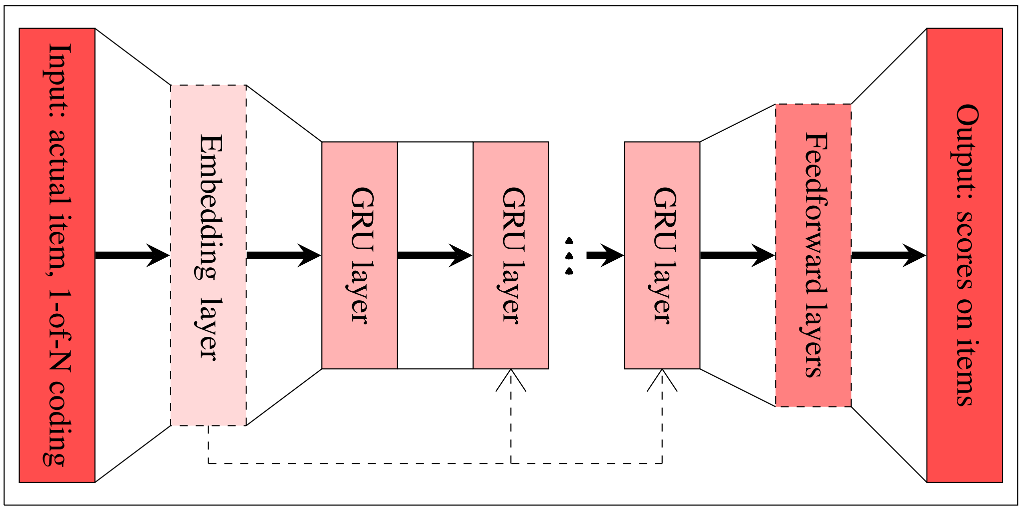


Figure 3.1: General network architecture, adapted from (Hidasi et al., 2015).

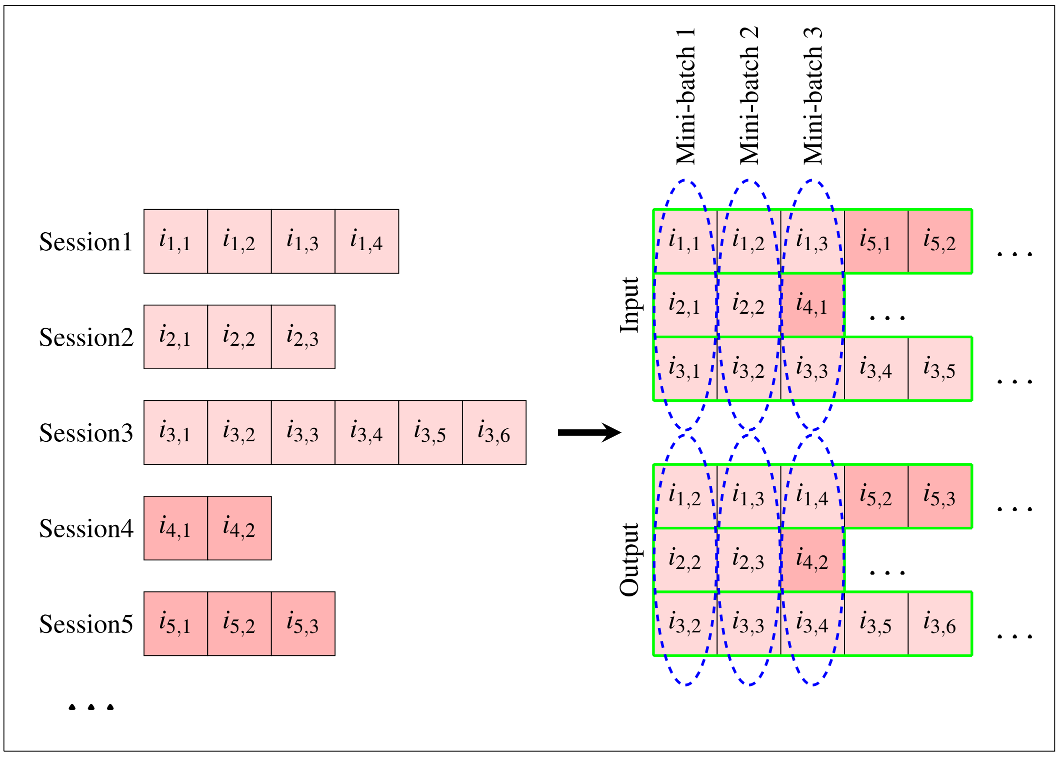
3.3.3 Session-Parallel Mini-batches

RNNs for the natural language processing frequently apply in-sequence mini batches that are not suitable for session-based recommendation problem (Hidasi et al., 2015). Therefore, session-parallel mini batches are used (Figure 3.2). First, an order for a session is created and then the first events from the first X sessions are used to create the first mini-batch's input. Similarly, second events from X session are used to form the second mini-batch and so on. In the situation when a session ends the next is positioned on its place. The assumption about the independence of the sessions requires the hidden state to be reset when the switch happens (Hidasi et al., 2015).

3.3.4 Output Sampling

Because number of items can grow to hundreds of thousands or even more what would make the scale of the algorithm enormous and unusable in practice, the output has to be sampled and consequently the score is computed for a small number of items. Therefore, not every weight will be updated. Moreover, apart from the calculation of the desired output, scores of some negative examples need to be calculated in order to adjust the weights and to acquire a higher ranking for the desired output (Hidasi et al., 2015).

Hidasi et al. (2015) explain that an arbitrary missing event could be interpreted as caused by user



Session1
$i_{1,1}$ $i_{1,2}$ $i_{1,3}$ $i_{1,4}$
Session2
$i_{2,1}$ $i_{2,2}$ $i_{2,3}$
Session3
$i_{3,1}$ $i_{3,2}$ $i_{3,3}$ $i_{3,4}$ $i_{3,5}$ $i_{3,6}$
Session4
$i_{4,1}$ $i_{4,2}$
Session5
$i_{5,1}$ $i_{5,2}$ $i_{5,3}$
...
Mini-batch 1
Mini-batch 2
Mini-batch 3
Input
$i_{1,1}$ $i_{1,2}$ $i_{1,3}$ $i_{5,1}$ $i_{5,2}$ ...
$i_{2,1}$ $i_{2,2}$ $i_{4,1}$ ...
$i_{3,1}$ $i_{3,2}$ $i_{3,3}$ $i_{3,4}$ $i_{3,5}$ ...
Output
$i_{1,2}$ $i_{1,3}$ $i_{1,4}$ $i_{5,2}$ $i_{5,3}$ ...
$i_{2,2}$ $i_{2,3}$ $i_{4,2}$ ...
$i_{3,2}$ $i_{3,3}$ $i_{3,4}$ $i_{3,5}$ $i_{3,6}$ ...

Figure 3.2: Session-parallel mini-batch formation, adapted from (Hidasi et al., 2015).

not knowing about the existence of the item what lead to no interaction. They argue that it is more probable that the user knows about the more popular items. Hence, there is a higher probability that a missing event of a more popular item expresses users dislike for it and items should be sampled proportionally, according to their popularity. In order to reduce the computational times and make the code less complex, items of other mini-batch training examples are used as negative examples (Hidasi et al., 2015).

3.3.5 Ranking Loss

Recommender systems are based on relevance-based ranking of items. Ranking can be either pairwise, pointwise or listwise. Pairwise ranking compares the rank or the score of pairs of negative and positive item, and then the loss imposes that the positive item's rank should be lower than negative item's. Pointwise ranking measures independently of each other the rank or the score of items whereas, the loss enforces a low rank for relevant items. Listwise ranking adapts ranks and scores of all items in order to compare them with the perfect ordering. Hidasi et al. (2015) found that pointwise ranking loss was unstable with their network and therefore in this work only the pairwise loss and listwise loss is used.

According to Hidasi and Karatzoglou (2018) the recommendation of the next item could also be considered as a classification task with the class labels being the items in the system, and sequences of items have to be assigned with the following items label. Moreover they realised that as number of samples is increasing, the BPR and Top1 (Hidasi et al., 2015) suffer from vanishing gradient problem. Hence, they derived BPR-max and Top1-max functions from BPR and Top1 functions. All of the loss functions used in this work are defined as follows:

– **BPR** is a matrix factorization method (Rendle et al., 2012). It applies pairwise ranking loss by comparing the score of positive and sampled negative examples. In this work, based on (Hidasi et al., 2015), the positive item's score is compared with sampled items and their average is used as the loss. The loss is as follows:

$$L_s = -\frac{1}{N_S} \cdot \sum_{j=1}^{N_S} log(\sigma(\hat{r}_{s,i} - \hat{r}_{s,j})) \tag{3.7}$$

where at the given point of session N_S stands for the sample size. $r_{s,k}$ represents score on item k whereas i and j are the desired (next in the session) and negative sample respectively.

– **Top1** - Hidasi et al. (2015) devised this ranking loss which is a regularized approximation of relative rank of the relevant item. The relative rank of a relevant item can be calculated by:

$$\frac{1}{N_S} \cdot \sum_{j=1}^{N_S} I\{(\hat{r}_{s,j} > \hat{r}_{s,i})\} \tag{3.8}$$

and $I\{\cdot\}$ is approximated with sigmoid. In order to avoid the situation in which some positive items would also act as negative and thus the scores would become increasingly higher, a regularization term is added to the loss. Hence, the final loss function is given by:

$$L_s = -\frac{1}{N_S} \cdot \sum_{j=1}^{N_S} \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2) \tag{3.9}$$

– **Top1-max** belongs to the family of listwise loss functions. The individual losses are weighted by s_j[16], corresponding softmax scores, and then summed. It is the process of continuous approximation to the maximum selection:

$$L_{Top1-max} = \sum_{j=1}^{N_S} s_j \Big(\sigma(r_j - r_i) + \sigma(r_j^2) \Big) \tag{3.10}$$

– **BPR-max** also belongs to listwise loss family functions. It aims at maximizing the probability that the target score is above the maximal sample score $r_{max} = max_j r_j$. $P(r_j = r_{max})$ and $P(r_i > r_j)$ can be approximated by s_j and $\sigma(r_i - r_j)$ respectively. The loss aims at minimizing the negative log probability and is given by:

$$L_{BPR-max} = -\log \sum_{j=1}^{N_S} s_j \sigma(r_j - r_i) \tag{3.11}$$

3.4 Factorization-based Models

As presented in the literature review part, plenty of new factorization-based methods were introduced in the recent years for the purpose of sequential recommendation. In this work three aforementioned approaches are adapted and applied to a session-based setting as proposed in (Ludewig and Jannach, 2018).

[16] $\sum s_j = 1$

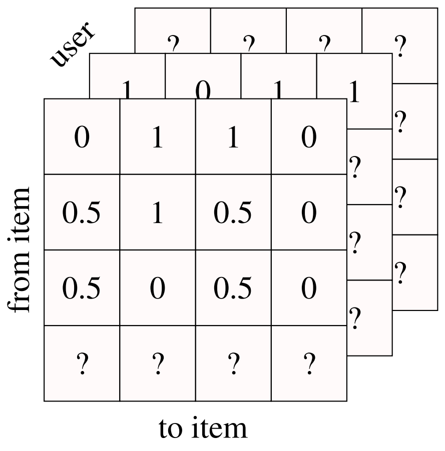


Figure 3.3: Personalized transition cube, adapted from (Ludewig and Jannach, 2018).

3.4.1 Factorized Personalized Markov Chains (FPMC)

When the model was originally devised it aimed at predicting the next shopping basket of users based on their past purchase history. It can be applied to the session-based recommendation setting by reducing the basket size to only one item and by considering the past basket purchases as the items in the current session. FPMC is obtained by combining Markov chains and factorized user-item matrix with three-dimensional tensor factorization method as presented in Figure 3.3. The transition probabilities between items are captured by the third dimension.

With the help of Canonical Tensor Decomposition the cube is factored into latent matrices, which are used to estimate the ranking as presented in the Equation 3.12 (Ludewig and Jannach, 2018):

$$\hat{r}_{u,l,i} = \langle v_u^{U,I}, v_i^{I,U} \rangle + \langle v_i^{I,L}, v_l^{L,I} \rangle + \langle v_u^{U,L}, v_l^{L,U} \rangle \tag{3.12}$$

where the score for the item i with user preferences u, with formerly examined item l, is denoted by $\hat{r}_{u,l,i}$. Six latent matrices, which resulted from three-dimensional decomposition, are represented by $v^{X,Y}$ with the latent factors for dimension X in respect to Y, e.g., $v^{I,L}$ stands for the latent factors of an item in respect to item examined in the past. Moreover, for instance $v_i^{I,L}$ represents factors for item i and $v_u^{U,L}$ single user's u factors. In order to calculate the ranking, the latent factors are combined with the dot product of form $\langle a,b \rangle$. Stochastic gradient descent (SGD) and the BPR loss function are used to train the latent factors. In the session-based problem setting the long-term user interaction histories are not available. Therefore, the session latent vectors are measured as the mean of individual items' latent factors in the session (Ludewig and Jannach, 2018).

3.4.2 Factored Item Similarity Models (FISM)

This approach can be directly applied to a session-based setting with cold-start problem, in a case when no user representation can be trained. However, no sequential transitions of item-to-item is incorporated. The following function aims at predicting ratings of an item (Kabbur et al., 2013):

$$\hat{r}_{u,i} = b_u + b_i + (n_u^+)^{-\alpha} \sum_{j \in R_u^+} p_j q_i^T \tag{3.13}$$

where $\hat{r}_{u,i}$ is a score for item i and user u. It consists of the sum of latent vector products $p_j q_i^T$ between already inspected items R_u^+ in a session and item i. b_u and b_i stand for bias and the total number of ratings by a user u is denoted by n_u^+ that is additionally normalized by parameter α. As recommended in (Ludewig and Jannach, 2018) BPR loss function is applied in case of optimizing the top-n recommendations.

3.4.3 Factorized Sequential Prediction with Item Similarity Models (Fossil)

Fossil method combines FISM with Markov chains to add information about the sequences into the model from (He and McAuley, 2016):

$$\hat{r}_{u,l,i} = \underbrace{\sum_{j \in R_u^+ \setminus \{i\}} p_j q_i^T}_{\text{long-term preferences}} + \overbrace{(w + w_u)}^{\text{personalized weighting}} \cdot \underbrace{n_l m_i^T}_{\text{sequential dynamics}} \tag{3.14}$$

where $\hat{r}_{u,l,i}$ represents a rating score as in the Equation 3.12. Long-term user preferences are denoted by the first term as in Equation 3.13. For the purpose of factoring in the user-independent probability that item l is followed by item i latent vector n_l is multiplied with other latent vector m_i for item l and i respectively. In addition, personalized weighting is added with w and w_u as a global and personalized factor respectively. In the scenario of this paper R_u denotes current session and the users are represented by the sessions and again BPR loss function is applied.

3.5 Baselines

The following baselines are included for the comparison with the rest of the models. All of them are not computationally expensive, apply very simple prediction scheme and consider the last item of a current session for making prediction:

– **POP** recommendation based on popularity. It refers to sorting the list and it is based on item score.

– **SPOP** it is designed to recommend the items with the highest popularity from the current session. With items changing gaining more events during the session, the recommendation list changes. This approach performs well in domains which are featured by high repetitiveness (Dang et al., 2019).

– **Markov Chains** (MC) focuses on the sequences in data. In this work, the rules are from a first-order MC (Norris, 1998) by which the transition probability of a pair of subsequent events is described. Ludewig and Jannach (2018) calculate the score by counting how often an item was immediately viewed after another. The absolute number is then transformed into transition probability by a normalization term.

– **IKNN** is often used as a baseline. It recommends items that are similar to the given item. The distance measure is calculated as follows:

$$s_{i,j} = \frac{\sum_s I\{(s,i) \in D \,\&\, (s,j) \in D\}}{(supp_i + \lambda)^{\alpha}(supp_j + \lambda)^{1-\alpha}} \tag{3.15}$$

The number of co-occurring items in a current session with each session is divided by the multiplied number of sessions in which the individual items appear as presented in Equation 3.15 where $supp_i$ stands for the number of sessions where i appears. Moreover, in order to avoid unintentional high similarities of infrequently visited items, regularization is included with λ. α aims at holding a balance between normalizing with the supports of two items. In case of $\alpha = 0,5$ the equation gives cosine similarity (Hidasi et al., 2015).

4 Experiment

4.1 Research Questions

In regard to the approaches presented in section two and three it remains difficult to see which technique should be favoured. More empirical research is therefore needed to move forward in this field of research. This experiment, based on specific datasets, aims to take a step forward with this challenge, by answering the following three questions:

1. *Is RNN-based approach superior to KNN-based approaches in terms of performance metrics, on the given datasets?*
2. *Which loss function performs the best with GRU4Rec model on the given datasets?*
3. *Which factorization-based model performs the best on the given datasets?*

By answering these questions we hope to see which models should be favored in future researches and projects.

4.2 Datasets and Source Code

The proposed approaches are evaluated on three datasets. The sessions of length one are not included into the analysis. The choice of the datasets was determinated by their availability and their thematic proximity.

Nowplaying[17] dataset consists of music session data with ratings from an online music platform Last.fm. It is split into sets with 905,211; 250,447 and 186,110 events for training, testing and validating respectively.

Movielens-1M[18] is a movie rating dataset which contains one million of ratings. All users in the dataset reviewed at least 20 movies. The dataset does not contain information about session Ids. Hence, during the preprocessing each user Id number is treated as a session Id. Because each user during one session is assigned a diffrent user Id, this operation does not influence the results. The dataset is split into training, test and validation set with 788,741; 144,894 and 66,290 events respectively. For the models where parameter tuning is not applicable only the training and test sets are used.

[17]https://www.dropbox.com/sh/dbzmtq4zhzbj5o9/AAAMMlmNKL-wAAYK8QWyL9MEa/Datasets?dl=0&subfolder_nav_tracking=1

[18]https://grouplens.org/datasets/movielens/

Yahoo-Music dataset was provided upon agreement as a part of Yahoo's Research Alliance Webscope program[19]. Due to its huge size, the dataset is cut so that before preprocessing it contains only one million events. In order for it to fit the session-based recommendation problem user-id is treated as a session Id, artist Id is treated as item-Id and a timestamp is assigned, as a proxy, to every session with each following event in the same session happening one second after another. The dataset is split into training, test and validation set with 397,244; 199,020 and 196,459 events respectively.

Table 4.1: Properties of the preprocessed datasets.

Dataset	**Nowplaying**	**Movielens-1M**	**Yahoo**
Events	1.47M	1M	994,407
Sessions	153,383	6040	16,685
Items	145,045	3592	13,930
Timespan in Days	535	1039	12

The python code applied in this work for preprocessing, model training and the evaluation is adapted from Ludewig and Jannach's (2018) and Hidasi et al.'s (2015) studies. The calculation is done in python and anaconda environment. Due to its different characteristics every dataset is preprocessed in a slightly different way. Moreover, deep learning models are trained with theano and tensorflow library for python.

4.3 Evaluation Metrics and Procedure

For the algorithms that do not require parameter optimization (POP, SPOP, MC, FISM, FOSSIL and FPMC) only train and validation sets are taken into consideration. The rest of the models is firstly optimized with the help of the test set and then evaluated on the same validation set as aforementioned models. Whereas the size of Yahoo's and Nowplaying's test and validation set is comparable, the Movielens-1M's test set is considerably smaller than the validation set.

Based on the literature review on recommender systems, the following performance metrics are adopted in the experiment as described in the study "A Novel KNN Approach for Session-Based Recommendation" (Guo et al., 2019): **Mean Reciprocal Rank (MRR)**, **Hit Rate (HR)**[20] and

[19]http://research.yahoo.com

[20]The RRM and HR are also referred to in the further development of this work as "accuracy oriented evaluation

Coverage. For all of the following measures L stands for the length of the recommendation list $R_i@L$ for current session i. Furthermore, the cardinality of item set I and test set $Test$ is depicted by $|I|$ and $|Test|$ respectively:

$$MRR@L = \frac{1}{|Test|}(\sum_{j \in Test} \frac{1}{rank_j}) \tag{4.1}$$

where MRR's item rank, with items interacted in sample j is depiced by $rank_j$.

$$HR@L = \frac{1}{|Test|}\sum_{i \in Test}(hit_i) \tag{4.2}$$

if in $R_i@L$ an item of ongoing session i is recommended, the hit_i equals one for the HR metric.

$$Coverage@L = \frac{1}{|I|}|\bigcup_{j \in Test} R_j@L| \tag{4.3}$$

Catalog coverage is the percentage of recommended items in L top places that ever appear in all recommendation lists.

4.4 Parameter Optimization

As already mentioned in the related research part some of the models used in this experiment require parameter optimization. Apart from the IKNN model, all of the baselines are not optimized in this experiment. For deep learning models, parameter optimization is a very time-consuming process due to long training time and a considerable amount of parameters. That means that IKNN, CKNN and GRU4Rec need to be optimized by setting their parameter values before the model is trained to a specific value. It is done as follows:

IKNN - in the case of IKNN model only three parameters are being checked on, namely: Nsims (the number of most similar items to which non-zero similarity scores are given), lambda (responsible for regularization), and alpha (keeps the balance between normalizing with two items' support). Nsims and lambda are optimized in pairs. The best-performance-giving alpha for each dataset is chosen for the best pair of previous two parameters.

CKNN - k is the number of neighboring sessions from which the item scores are calculated. Sample size is the subset length of training sessions from which the nearest neighbor is calculated. Both of this parameters are optimized in pairs along with one of the four similarity metrics[21].

metrics".

[21]The four similarity metrics are the following: tanimoto, jaccard, cosine and binary.

GRU4Rec - in the third case the training and evaluation of the deep learning models are considerably more time-consuming and it has more possible parameters to be optimized than the aforementioned approaches. Hence, the number of times the model is going to be trained will be decreased. Due to different characteristics of every dataset the optimal value of each parameter varies between each dataset. Four different loss functions are tested: Top1, Top1-max, BPR and BPR-max. For final activation and hidden activation functions linear and tanh are chosen respectively. Furthermore, different values for learning rate, dropout rate, momentum[22] and the number of epochs and layers are chosen to find the set of parameters giving the best result.

5 Results

5.1 Description of the Results

The Table 5.1 presents the baseline models' performance on the three different datasets. The best results for both of the accuracy metrics were obtained with MC model, exceeding other baselines in almost every metric, apart from IKNN's HR@20 for Nowplaying dataset. Especially for the Yahoo dataset the results of Markov Chains are substantially better than of the other baselines.

Table 5.1: Baseline models' performance on different datasets.

Baseline\dataset	**YOOCHOOSE**		**MOVIELENS-1M**		**YAHOO**	
	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
MC	0.154	**0.086**	**0.201**	**0.064**	**0.776**	**0.388**
Item-KNN	**0.170**	0.056	0.150	0.040	0.148	0.033
POP	0.020	0.007	0.036	0.007	0.091	0.020
SPOP	0.084	0.047	0.004	0.0004	0.015	0.001

The three following tables (Table 5.2, 5.3 and 5.4) contain the results of all of the models for Movielens-1M, Yahoo and Nowplaying. Each one of them contains the results of marix factorization-based approaches, one baseline, CKNN for session-based recommendation, and GRU4Rec algorithm with four different loss functions. BPR loss for the GRU4Rec model on all three datasets was unstable numerically in many cases. This could have caused slightly worse

[22]it is a strategy in machine learning that is used to accelerate the stochastic gradient descent by oscilation dampening in the right direction. In this case Nesterov momentum is used (Gomila, n.d.).

performance of this loss compared to other losses, because not all of the combinations of parameters could be tried out. Hence, there is possibility that the model could not be optimized to its fullest capacity.The combination of best performing parameters for deep learning models, CKNN and IKNN can be found in the Appendix (Tables A.1, A.2, A.3, A.4, A.5).

Surprisingly, for the Movielens-1M dataset the baseline model MC obtains the best results for nearly every metric, excluding HR@20, for which the GRU4Rec model with BPR-max loss performed the best (exceeding other deep learning models only slightly). It can be easily said that MC is the best performing model for this dataset. The best results for CKNN were obtained with binary similarity metric. Moreover, the results of CKNN-binary model for session-based recommendation are substantially inferior when compared to almost every model, including GRU4Rec algorithm. On the contrary, when comparing the GRU4Rec model with different losses, one cannot answer which of the losses gave the best result. As given in Table 5.2 the BPR-max loss function scored the best only for the HR@20 whereas for the MRR@20 and HR@10 it obtained slightly worse results than Top1-max loss. Top1-max loss performed the best among other losses in four out of six metrics presented. The best performing approaches out of matrix factorization-based models is FPMC that scores much better than FOSSIL and FISM across all of the evaluation metrics, even exceeding deep learning approaches in terms of coverage.

Table 5.2: Other models' performance on Movielens-1M dataset.

Model\Dataset	**MOVIELENS-1M**					
	HR@20	MRR@20	Cov@20	HR@10	MRR@10	Cov@10
FISM	0.034	0.006	0.228	0.016	0.005	0.162
FOSSIL	0.009	0.001	0.006	0.004	0.001	0.003
FPMC	0.130	0.031	0.799	0.081	0.028	0.685
CKNN-binary	0.036	0.006	0.169	0.017	0.005	0.129
Baseline-MC	0.201	**0.064**	**0.933**	**0.141**	**0.060**	**0.865**
GRU4Rec-Top1	0.223	0.047	0.502	0.132	0.041	0.424
GRU4Rec-BPR	0.208	0.046	0.639	0.123	0.040	0.553
GRU4Rec-Top1-max	0.222	0.055	0.636	0.137	0.049	0.578
GRU4Rec-BPR-max-1	**0.224**	0.053	0.556	0.136	0.047	0.499

For the first dataset from the music domain - Yahoo - the best performing model in four out of six categories is GRU4Rec with Top1-max loss function. The approach substantially outperforms the Top1 loss function, but performs worse in terms of coverage to FPMC, MC and GRU4Rec with BPR loss function. The best performing similarity for CKNN is the tanimoto similarity. Its performance overall could again be considered as inferior compared to almost all of the other approaches, apart from FISM model. In terms of accuracy oriented evaluation metrics the GRU4Rec with Top1-max scores the best and models with BPR loss and BPR-max loss do not fall far behind. In this case only the Top1 loss function performs relatively poor compared to other losses. Again the baseline MC model's result is relatively good, not achieving much less than the other models in terms of the accuracy oriented metrics. In addition, the FPMC model again performs much better than other matrix factorization based approaches and even achieves the best result for coverage leaving behind all of the other models for Yahoo dataset.

Table 5.3: Other models' performance on Yahoo dataset.

Model\Dataset	**Yahoo**					
	HR@20	MRR@20	Cov@20	HR@10	MRR@10	Cov@10
FISM	0.125	0.025	0.298	0.070	0.022	0.208
FOSSIL	0.208	0.038	0.560	0.114	0.031	0.428
FPMC	0.511	0.147	**0.994**	0.363	0.137	**0.956**
CKNN-tanimoto	0.134	0.025	0.081	0.071	0.021	0.065
Baseline-MC	0.776	0.388	0.905	0.679	0.381	0.710
GRU4Rec-Top1	0.707	0.281	0.278	0.576	0.252	0.165
GRU4Rec-BPR	0.833	0.432	0.850	0.735	0.425	0.715
GRU4Rec-Top1-max	**0.837**	**0.447**	0.802	**0.745**	**0.441**	0.667
GRU4Rec-BPR-max-0.5	0.835	0.443	0.784	0.742	0.437	0.660

For the second dataset from the music domain - Nowplaying - the relative results are a bit different (Table 5.4). CKNN for session-based recommendation scores the best in HR@20 and HR@10 out of all of the models. Moreover, the model with the best coverage with recommendation list of length 20 and 10 is again FPMC. The GRU4Rec algorithm with the BPR-max loss scores the best in both MRR@20 and MRR@10. In this situation it cannot be decided whether KNN-based model outperformed the RNN-based models. Moreover, the losses Top1-max and BPR-max perform much better than two other losses on every evaluation metric but BPR-max performs better on accuracy oriented evaluation metrics and Top1-max on Coverage. Out of three matrix factorization-based models the FPMC is performing considerable better than FISM and FOSSIL in every evaluation metric.

Table 5.4: Other models' performance on Nowplaying dataset.

Model\Dataset	**Nowplaying**					
	HR@20	MRR@20	Cov@20	HR@10	MRR@10	Cov@10
FISM	0.024	0.013	0.643	0.019	0.012	0.508
FOSSIL	0.037	0.009	0.849	0.023	0.008	0.687
FPMC	0.055	0.038	**0.980**	0.049	0.037	**0.913**
CKNN-tanimoto	**0.217**	0.068	0.644	**0.153**	0.063	0.478
Baseline Markov	0.154	0.086	0.623	0.134	0.085	0.553
GRU4Rec-Top1	0.144	0.054	0.348	0.105	0.052	0.271
GRU4Rec-BPR	0.129	0.050	0.342	0.094	0.048	0.266
GRU4Rec-Top1-max	0.179	0.082	0.676	0.143	0.079	0.676
GRU4Rec-BPR-max-1	0.186	**0.090**	0.621	0.151	**0.087**	0.504

5.2 Comparison with Literature

As seen in the studies of Hidasi et al. (2015) and Ludewig and Jannach (2018) GRU4Rec almost always performs very well and can be assigned to a best performing group of algorithms on every dataset.

As presented in the work of Hidasi and Karatzoglou (2018), Top1-max and BPR-max achieve better results in accuracy oriented metrics. A comparable results was also reproduced in this work where BPR-max and Top1-max in almost all of the cases outperformed BPR and Top1. As Hidasi and Karatzoglou (2018) already pointed out this might be due to the fact that that both Top1 and BPR suffered from the vanishing gradient problem which caused the learning process to stop and thus giving relatively worse results.

Depending on the dataset the CKNN for session-based recommendation was either in line with the study's findings of Jannach and Ludewig (2017) or produced one of the worse results out of all algorithms on almost all evaluation metrics. Hence, the results show that not always nearest neighbors-based algorithms should be considered as competitive baselines for session-based recommendation.

The results are not in line with Rendle et al. (2010) who argued that FPMC outperforms, inter alia MC. In this study it happened only on two datasets in terms of coverage but the MC consistently outperformed FPMC when it came down to accuracy oriented metrics. On the contrary, the result of this work are in line with the work of Ludewig and Jannach (2018), the MC's performance in almost all of the cases surpassed the matrix factorization-based models - FISM, FOSSIL, FPMC - in terms of accuracy oriented metrics. MC probably performs so well in this setting because it concentrates only on the sequential information when every session is a different user what probably caused other matrix factorization-based models not use their full potential.

The results of CKNN for session-based recommendation were far from promising on two out of three datasets. Grčar et al. (2006) argued that KNN algorithm's performance is good in the datasets with low sparsity but got worse as the sparsity in the dataset rised. Moreover, high sparsity can be translated to low values of coverage (Schedl et al., 2018). This statement can be reflected by the coverage results of the CKNN for session-based recommendation on Movielens-1M and Yahoo datasets where the coverage is very low in comparison to other algorithms. The results are not in line with the work of Ludewig and Jannach (2018), in which to the researchers surprise the GRU4Rec model did not score substantially better than a much simpler and computationally easier CKNN for session-based recommendation model. But the overall final thoughts that the researchers expressed is that they could identify very well performing algorithms on every dataset but on the other hand the relative performance of algorithms varied across different datasets due to the factors that were not yet completely understood to the community (Ibid.).

Session-based recommendation could be a good approach to reduce the dependency on user profiling and still keep the high accuracy. Nevertheless, at least in some domains, they should try to incorporate other factors that could possibly determine users actual mood and intentions. Moreover, in domains such as music recommendation it is hard to determine the intent of the user to listen to something new or to listen to already known material (Kapoor et al., 2015).

6 Conclusion

All in all the RNN-based algorithms perform consistently very well when it comes to accuracy oriented evaluation metrics. Based on given results one is unable to fully answer all of the research questions but it is possible to give future directions which can help to better understand the problem.

The presented results make it possible to answer the first question GRU4Rec algorithm performs better than CKNN for session-based recommendation in almost every case. Thus if one is interested in consistent performance they should consider the GRU4Rec algorithm as a one to go with. Moreover, it is hard to give a clear answer to the second research question. There is no doubt that both Top1-max and BPR-max perform better than the other losses but the difference between Top1-max's and BPR-max's performances is not so obvious. There is a slight hint on every dataset indicating that Top1-max yields better coverage, but on the different evaluation metrics the best performing loss is hard to appoint. The third question can be answered without any doubt. Out of three matrix factorization-based approaches the FPMC was the one that was the best across all of the datasets and evaluation metrics.

Every study could have been conducted better. In our study, we could have tested some of the recently introduced types of KNN models for session-based recommendation. Moreover, we could have tested more of the combinations of parameters for GRU4Rec model, which could have resulted in better performance of this model.

As Ludewig and Jannach's study (2018) and the one presented here it is not yet known why the relative performance of different algorithms can vary across different datasets and why simple algorithms can perfrom better on some of them, e.g. MC model. Hence one should research further on this topic in order to fully understand this phenomenon.

References

Batmaz, Z., Yurekli, A., Bilge, A., & Kaleli, C. (2018). A review on deep learning for recommender systems: Challenges and remedies. *Artificial Intelligence Review*, *52*, 1–37. https://doi.org/10.1007/s10462-018-9654-y

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. *Int J Digit Libr*, *17*(4), 305–338. https://doi.org/10.1007/s00799-015-0156-0

Bennett, J., Lanning, S., & Netflix, N. (2007). The Netflix Prize, In *In KDD Cup and Workshop in conjunction with KDD*.

Bonnin, G., & Jannach, D. (2014). Automated Generation of Music Playlists: Survey and Experiments. *ACM Comput. Surv.*, *47*(2), 26:1–26:35. https://doi.org/10.1145/2652481

Burke, R., Felfernig, A., & Göker, M. H. (2011). Recommender Systems: An Overview. *AI Magazine*, *32*(3), 13–18. https://doi.org/10.1609/aimag.v32i3.2361

Chen, J., Nairn, R., & Chi, E. (2011). Speak Little and Well: Recommending Conversations in Online Social Streams, In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vancouver, BC, Canada, ACM. https://doi.org/10.1145/1978942.1978974

Chen, X., Xu, H., Zhang, Y., Tang, J., Cao, Y., Qin, Z., & Zha, H. (2018). Sequential Recommendation with User Memory Networks, In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, Marina Del Rey, CA, USA, ACM. https://doi.org/10.1145/3159652.3159668

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches, arxiv 1409.1259. Retrieved October 10, 2019, from http://arxiv.org/abs/1409.1259

Covington, P., Adams, J., & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations, In *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA.

Dang, T. K., Nguyen, Q. P., & Nguyen, V. S. (2019). Evaluating session-based recommendation approaches on datasets from different domains, In *Fdse*.

Devooght, R., & Bersini, H. (2017). Long and Short-Term Recommendations with Recurrent Neural Networks, In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, Bratislava, Slovakia, ACM. https://doi.org/10.1145/3079628.3079670

Dziuda, D. M. (2010). *Data Mining for Genomics and Proteomics: Analysis of Gene and Protein Expression Data by Dziuda, Darius M.(July 6, 2010) Hardcover*. Wiley-Interscience.

Ferwerda, B., Schedl, M., & Tkalcic, M. (2015). Personality & Emotional States : Understanding Users ' Music Listening Needs.

Garfinkel, R., Gopal, R. D., Pathak, B. K., Venkatesan, R., & Yin, F. (2006). *Empirical Analysis of the Business Value of Recommender Systems* (SSRN Scholarly Paper No. ID 958770). Social Science Research Network. Rochester, NY. Retrieved October 4, 2019, from https://papers.ssrn.com/abstract=958770

Ge, M., Delgado-battenfeld, C., & Jannach, D. (2010). Beyond accuracy: Evaluating recommender systems by coverage and serendipity, In *In RecSys '10*.

Gillhofer, M., & Schedl, M. (2015). Iron Maiden While Jogging, Debussy for Dinner? https://doi.org/10.1007/978-3-319-14442-9_44

Gomila, D. F., Jude. (n.d.). *Nesterov momentum - Wiki*. Retrieved February 5, 2020, from https://golden.com/wiki/Nesterov_momentum

Grčar, M., Mladenić, D., Fortuna, B., & Grobelnik, M. (2006, October 20). Data Sparsity Issues in the Collaborative Filtering Framework. https://doi.org/10.1007/11891321_4

Guo, H., Tang, R., Ye, Y., Liu, F., & Zhang, Y. (2019). A novel knn approach for session-based recommendation, In *Pakdd*.

He, R., & McAuley, J. (2016). Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation, arxiv 1609.09152. Retrieved November 30, 2019, from http://arxiv.org/abs/1609.09152

Heinz, S., Bracher, C., & Vollgraf, R. (2017). An LSTM-Based Dynamic Customer Model for Fashion Recommendation, arxiv 1708.07347. Retrieved October 9, 2019, from http://arxiv.org/abs/1708.07347

Hernández del Olmo, F., & Gaudioso, E. (2008). Evaluation of recommender systems: A new approach. *Expert Systems with Applications*, *35*(3), 790–804. https://doi.org/10.1016/j.eswa.2007.07.047

Hidasi, B., & Karatzoglou, A. (2018). Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management - CIKM '18*, arxiv 1706.03847, 843–852. https://doi.org/10.1145/3269206.3271761

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2015). Session-based Recommendations with Recurrent Neural Networks, arxiv 1511.06939. Retrieved October 12, 2019, from http://arxiv.org/abs/1511.06939

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Huang, C.-L., & Huang, W.-L. (2009). Handling sequential pattern decay: Developing a two-stage collaborative recommender system. *Electronic Commerce Research and Applications*, *8*, 117–129. https://doi.org/10.1016/j.elerap.2008.10.001

Jannach, D., Lerche, L., & Jugovac, M. (2015). Adaptation and Evaluation of Recommendations for Short-term Shopping Goals, In *Proceedings of the 9th ACM Conference on Recommender Systems*, Vienna, Austria, ACM. https://doi.org/10.1145/2792838.2800176

Jannach, D., & Ludewig, M. (2017). When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation, In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, Como, Italy, ACM. https://doi.org/10.1145/3109859.3109872

Kabbur, S., Ning, X., & Karypis, G. (2013). FISM: Factored Item Similarity Models for top-N Recommender Systems, In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Chicago, Illinois, USA, ACM. https://doi.org/10.1145/2487575.2487589

Kapoor, K., Kumar, V., Terveen, L., Konstan, J. A., & Schrater, P. (2015, September 16). "I like to explore sometimes": Adapting to Dynamic User Novelty Preferences, In *Proceedings of the 9th ACM Conference on Recommender Systems*, Vienna, Austria, Association for Computing Machinery. https://doi.org/10.1145/2792838.2800172

Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, *41*, 2065–2073. https://doi.org/10.1016/j.eswa.2013.09.005

Ludewig, M., & Jannach, D. (2018). Evaluation of Session-based Recommendation Algorithms. *User Model User-Adap Inter*, *28*(4-5), arxiv 1803.09587, 331–390. https://doi.org/10.1007/s11257-018-9209-6

Ludewig, M., Mauro, N., Latifi, S., & Jannach, D. (2019). Empirical Analysis of Session-Based Recommendation Algorithms, arxiv 1910.12781. Retrieved December 10, 2019, from http://arxiv.org/abs/1910.12781

McNee, S. M., Riedl, J., & Konstan, J. A. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems, In *Chi '06 extended abstracts on human factors in computing systems*, Montréal, Québec, Canada, Association for Computing Machinery. https://doi.org/10.1145/1125451.1125659

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted Collaborative Filtering for Improved Recommendations, In *Eighteenth National Conference on Artificial Intelligence*, Edmonton, Alberta, Canada, American Association for Artificial Intelligence. Retrieved October 3, 2019, from http://dl.acm.org/citation.cfm?id=777092.777124

Meyer, C., & Noble, S. (2013). *How retailers can keep up with consumers — McKinsey*. Retrieved January 31, 2020, from https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

Netflix Prize: Review Rules. (n.d.). Retrieved October 6, 2019, from https://www.netflixprize.com/rules.html

Norris, J. R. (1998, October 15). *Markov Chains* (1.). Cambridge, UK ; New York, Cambridge University Press.

Quadrana, M., Cremonesi, P., & Jannach, D. (2018). Sequence-Aware Recommender Systems, arxiv 1802.08452. Retrieved January 30, 2020, from http://arxiv.org/abs/1802.08452

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2012). BPR: Bayesian Personalized Ranking from Implicit Feedback, arxiv 1205.2618. Retrieved November 26, 2019, from http://arxiv.org/abs/1205.2618

Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2010). Factorizing Personalized Markov Chains for Next-basket Recommendation, In *Proceedings of the 19th International Conference on World Wide Web*, Raleigh, North Carolina, USA, ACM. https://doi.org/10.1145/1772690.1772773

Resnick, P., & Varian, H. R. (1997). Recommender Systems. *Commun. ACM*, *40*(3), 56–58. https://doi.org/10.1145/245108.245121

Schedl, M., Zamani, H., Chen, C.-W., Deldjoo, Y., & Elahi, M. (2018). Current challenges and visions in music recommender systems research. *Int J Multimed Info Retr*, *7*(2), 95–116. https://doi.org/10.1007/s13735-018-0154-2

Shakhnarovich, G., Darrell, T., & Indyk, P. (2006). *Nearest-neighbor methods in learning and vision: Theory and practice (neural information processing)*. The MIT Press.

Singhal, A., Sinha, P., & Pant, R. (2017). Use of deep learning in modern recommendation system: A summary of recent works. *ArXiv*, *abs/1712.07525*.

Subramaniyaswamy, V., & Logesh, R. (2017). Adaptive KNN based Recommender System through Mining of User Preferences. *Wireless Pers Commun*, *97*(2), 2229–2247. https://doi.org/10.1007/s11277-017-4605-5

Tan, Y. K., Xu, X., & Liu, Y. (2016). Improved Recurrent Neural Networks for Session-based Recommendations, arxiv 1606.08117. Retrieved October 11, 2019, from http://arxiv.org/abs/1606.08117

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep Learning based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.*, *52*(1), arxiv 1707.07435, 1–38. https://doi.org/10.1145/3285029

Zhang, Y., Ai, Q., Chen, X., & Croft, W. B. (2017). Joint Representation Learning for Top-N Recommendation with Heterogeneous Information Sources, In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, Singapore, Singapore, ACM. https://doi.org/10.1145/3132847.3132892

Zheng, L., Noroozi, V., & Yu, P. S. (2017). Joint Deep Modeling of Users and Items Using Reviews for Recommendation, arxiv 1701.04783. Retrieved October 9, 2019, from http://arxiv.org/abs/1701.04783

Appendix A Parameter Configuration

Table A.1: IKNN's best performing parameters on three datasets.

	Datasets								
	NOWPLAYING			**MOVIELENS-1M**			**YAHOO**		
Model\Parameters	Nsims	lambda	alpha	Nsims	lambda	alpha	Nsims	lambda	alpha
IKNN	300	600	0.9	150	300	0.35	1000	2000	0.5

Table A.2: CKNN's best performing parameters on three datasets.

	Datasets								
	NOWPLAYING			**MOVIELENS-1M**			**YAHOO**		
Model\Parameters	k	size[23]	sim[24]	k	size	sim	k	size	sim
CKNN	200	400	tanimoto	150	300	binary	200	400	tanimoto

Table A.3: Best performing parameters for GRU4Rec on **Yahoo**

Loss\Parameters	epochs	layers	batch size	dropout	learning rate	momentum	n sample
Top1	18	160	64	0.01	0.18	0.5	1000
BPR	19	190	64	0.01	0.07	0.55	1000
Top1-max	28	160	64	0.1	0.04	0.6	1000
BPR-max-0.5	25	140	64	0.1	0.04	0.4	1000

[23] sample size
[24] similarity

Table A.4: Best performing parameters for GRU4Rec on **Movielens-1M**.

Loss\Parameters	epochs	layers	batch size	dropout	learning rate	momentum	n sample
Top1	18	140	64	0.1	0.18	0.5	1000
BPR	14	100	64	0.1	0.02	0.25	1000
Top1-max	14	80	64	0.1	0.15	0.4	1000
BPR-max-1	15	50	64	0.1	0.15	0.35	1000

Table A.5: Best performing parameters for GRU4Rec on **Nowplaying**.

Loss\Parameters	epochs	layers	batch size	dropout	learning rate	momentum	n sample
Top1	14	120	64	0.1	0.14	0.4	1000
BPR	10	50	32	0.1	0.15	0.5	1000
Top1-max	14	120	64	0.1	0.12	0.6	1000
BPR-max-1	14	120	64	0.1	0.15	0.5	1000