University of Bergen
Department of informatics

Multi-List Recommendations for Personalizing Streaming Content

Author: Anastasiia Vlasenko
Supervisors: Prof. Dr. Christoph Trattner, Assoc. Prof. Pekka Parviainen

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

May, 2023

Abstract

The decision behind choosing a recommender system that yields accurate recommendations yet allows users to explore more content has been a topic of research in the last decades. This work attempts to find a recommender system for TV 2 Play, a movie streaming platform, that would perform well on implicit feedback data and provide multi-lists as recommendations. Several approaches are examined for suitability, and Collaborative Filtering and Multi-Armed Bandits are decided upon. The models for each approach are built using the pipeline utilized by TV 2 Play. The models are then compared in performance on several evaluation metrics in the first stage of offline testing, yielding Alternating Least Squares and Bayesian Personalized Ranking as the best-performing models. The second stage of offline testing includes testing the two models and their variants with the BM25 weighting scheme applied against each other. The unweighted Bayesian Personalized Ranking model has shown the highest user-centric metrics while maintaining relatively high recommendation-centric metrics, which led to that model being tested in online settings against the algorithm currently used by TV 2 Play team. The online testing has revealed that our model underperforms compared to the TV 2 Play model when used on the kids' page but produces equally good results on the movies page. The results can be attributed to the differences in behavioral content consumption patterns between users.

Acknowledgements

I would like to, first and foremost, express my deepest gratitude to my supervisors, Christoph Trattner and Pekka Parviainen, who have not only supported me throughout the whole year but have also shown me that a person can never learn enough and that there is always room for improvement. Their guidance has been incomparable and helped me grow as a person and professional, for which I am eternally grateful. I have also had the privilege to work with Lars Skjærven and Astrid Johnsen Tessem from Content Discovery Team, TV 2 Play. The knowledge that Astrid and Lars have shared with me and the time they both have invested in this work can never truly be repaid. This opportunity has truly become a transformational learning experience for me. Last but not least, I want to thank my fiancé. Jakub cheered me up when it seemed there was no way to solve a problem. He has created a working atmosphere in our home, taking all the matters into his hands. This thesis would not have been completed without the cooperation of these people, for which I will forever be grateful.

Anastasiia Vlasenko

Bergen, Norway, 31 May, 2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In a world where the amount of products and content is rapidly increasing, navigating through all possible choices is becoming more complex. This can be seen in such domains as music, videos, news, ads, tourism, and online shopping, among other things [15, 26, 40]. In such situations nowadays, a consumer is often presented with recommendations made by an algorithm to narrow down choices and potentially help them find something of interest. The use of recommender systems in the last decades has completely changed the narrative for both consumers and providers of products [51]. Over the years, several new recommender systems algorithms were developed, including filtering, hybrid models, and, more recently, deep learning models [39]. The choice of a proper recommender algorithm is crucial for getting senseful recommendations for each user; thus, analyzing the needs of consumers and the available data define the type of recommender to be used. Conveniently, users' digital footprint allows for non-intrusive data collection, as real-life data can be sparse due to having to rate products manually. However, this poses some ethical concerns regarding data collection, which will be further discussed in Chapter 5.

A lot of the service providers, including video content and music, tend to present their products as multi-lists (also referred to as carousels) rather than single lists [35], which creates a new challenge in the recommendations domain. For the content providers, it allows for an

opportunity to expose more of the items and gives the users an opportunity to choose between generated categories instead of scrolling through a single list of recommendations [36]. Multi-lists are generally presented as several items grouped together by some criteria, e.g., genre or artist, where a user can get an overview of several recommendation lists by looking at the names of the lists and the first suggested items [12]. This approach to recommendations has gained a lot of attention and is in particular used by TV 2 Play streaming platform[1], which will be the focus of this work. The current algorithm, later on, referred to as Feed Collaborative Filtering (Feed-CF), recommends a multi-list, also referred to as feed, which is a row of editorially grouped items, based on comparing a user to other users that are similar and provides a multi-list that they are most likely to enjoy out of a predefined collection. However, there is always room for improving recommendations and the user experience, which is the goal behind the collaboration between TV 2 and the author. TV 2 Play is a streaming platform with a wide variety of video content, including sports, documentaries, movies, etc. This thesis attempts to improve user recommendations on the kids' page [2]. By starting with a smaller pool of items, there is potential to build a more robust algorithm that could be extended onto the other pages of the platform.

1.2 Problem

To our best knowledge, no work has directly compared the performance of several recommender approaches on multi-lists on a live streaming platform. Knowing that the results of offline testing can be substantially different from how users perceive and interact with the recommended multi-lists online, this work attempts to do the groundwork of exploring and building the recommender systems that would perform well on multi-lists, as well as suggest content that is tailored towards each user. Additionally, previous research has shown that already popular content tends to get more exposure, thus gaining even more popularity [1]. Measuring the popularity and diversity of the suggested items is crucial to understand what attracts users and how this knowledge can be used in future work. The main problem can be then addressed as follows:

[1]https://play.tv2.no
[2]https://play.tv2.no/barn

Given the set of predefined multi-lists and evaluation metrics, which recommender algorithm gives the best performance on a streaming platform, and how can the recommendations produced by the model be described by the metrics?

A snapshot of the kids' page from TV 2 Play can be found in Figure 1.1. Each row of items is a multi-list with a distinct name and contains several elements grouped by predefined criteria chosen by the editorial staff.

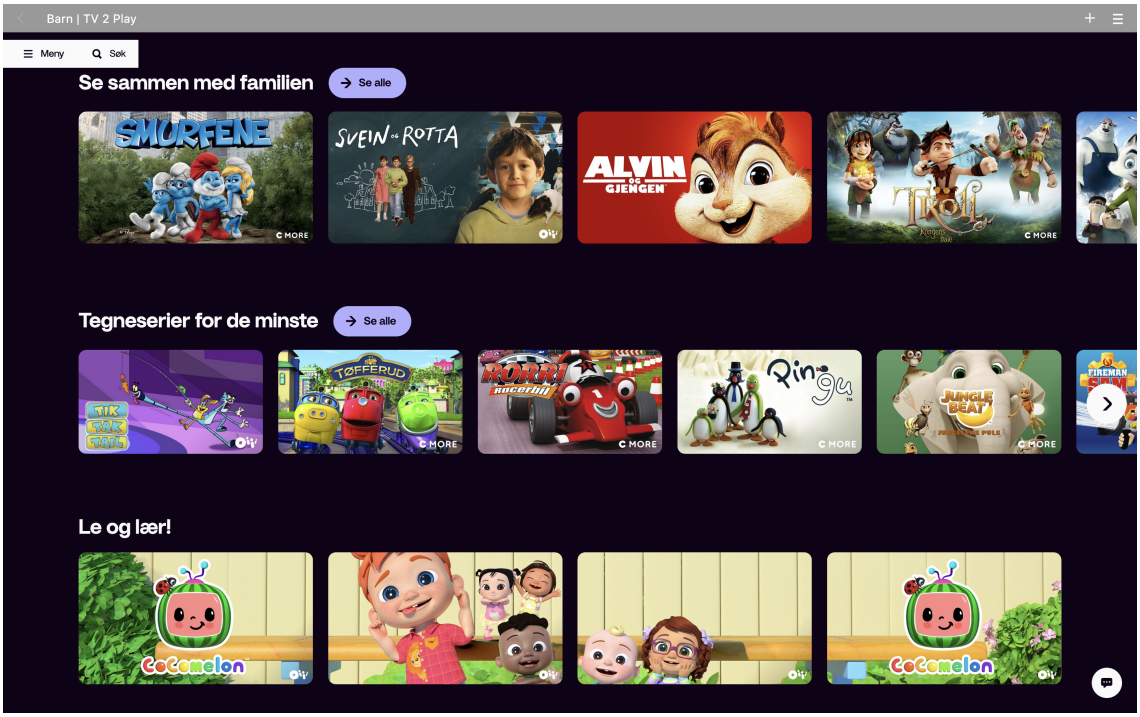

Figure 1.1: The outlay of the kids' page on TV 2 Play. "Se sammen med familien" ("Watch together with the family"), "Tegneserier for de minste" ("Cartoon series for the smallest"), and "Le og lær!" ("Laugh and learn!") represent the multi-lists. The snapshot was taken on 30.04.2023.

The main goal of TV 2 Play is to make the experience of using the platform more enjoyable and personalized to retain loyal and attract new customers. Since research and optimization of new algorithms are time-consuming, it is problematic for a company to allocate resources to tasks that could fail. Thus, the focal point of this work is to attempt to build a recommender system that could outperform Feed-CF, laying a foundation for the TV 2 Play team. The choice behind the recommender systems to implement is discussed in Chapter 2. Measuring performance is done by considering the accuracy of recommendations (user-centric evaluation

metrics) and the metrics that focus on the quality of recommendations (recommendation-centric metrics). The main problem with deciding what recommender to choose lies in the trade-off between user- and recommendation-centric metrics [14]. One has to decide whether it is beneficial to sacrifice some of the accuracy towards an opportunity to expose more items to users.

1.3 Research questions

This work is motivated by the following research questions:

1) How do different recommender systems compare in the performance of recommending multi-lists on user-centric and recommendation-centric metrics in the offline setting on TV 2 Play data?

2) How does the chosen model perform on recommending multi-lists in comparison to Feed-CF in the online setting on TV 2 Play data?

1.4 Thesis outline

In Chapter 2, we will look closer into the recent advancements in the field, the theory behind multi-lists, and the state-of-art models that can be used for the chosen problem. Further, in Chapter 3, we discuss how these models are set up, what kind of data are accessible, and how it is used for the analysis. Chapter 4 covers the results of chosen algorithms in the offline and online tests. Finally, Chapter 5 focuses on the main findings of this thesis and how they can be used for potential improvement in future work and this thesis' limitations.

Chapter 2

Background

The aim of this chapter is to discuss the most widely used algorithms for movie recommendations and their limitations, as well as their complexity and how applicable they are to the problem of recommending multi-lists. Section 2.1 describes in detail how recommending multi-lists differs from other approaches and what considerations should be taken into account when doing so. Section 2.2 goes on to describe the nature of the implicit feedback data, which the algorithms in this thesis are built on. Further, Section 2.3 provides an extensive overview of the most popular recommender algorithms, alongside their characteristics and potential problems that may arise when using them. Finally, Section 2.4 covers the decision-making process behind choosing a recommender system that would be appropriate for the problem at hand.

2.1 Recommending multi-lists

First of all, it is important to understand what is being recommended to users of the kids' page. In most cases, recommender systems suggest items, whether it is an ad, a song, or a news article. However, for the purpose of this research, we are looking to recommend lists of items, which will be referred to as multi-lists or feeds. The use of multi-lists has been popularized particularly by streaming platforms, such as Netflix, YouTube, and Spotify [13]. Having several items grouped in one list and presented to users helps a) divide items into

different groups by topic or genre and b) decrease the time needed to choose something of interest, thus, avoiding a situation where the user is presented with too many options and is unable to make a decision, namely choice overload [4]. However, the length of these lists possibly plays an even bigger role, as it is evident that the timeframe for capturing customers' attention is 60 to 90 seconds, which users spend to quickly analyze the content of recommended lists before making a decision [13]. In short, multi-lists aim to provide a more comprehensive and diverse set of recommendations to users. In general, studies agree that multi-lists can be useful when presenting a large number of recommendations and, in some cases, can be favored, providing seemingly more exploration and a more diverse set of recommendations [20]. Another study with a focus on the movie domain concluded that using multi-lists can be associated with lower exiting probability (probability that a user exits the platform without choosing an item), as well as lower navigation effort (number of interactions made by a user before making a decision) [36]. However, multi-lists can also lead to the longer time needed for decision-making in some cases [20], so the benefits of using multi-lists should be considered against possible limitations. The algorithms used further in this work recommend multi-lists without looking into the single items inside the lists. Thus, it is impossible to compare the performance of models that provide top-N recommendations as a single list to those being split into multi-lists, but it is important to understand the difference between the two and potential problems that might occur. Further on, by referring to recommendations, we refer to multi-lists solemnly, as it is the focal point of this work. An example of what multi-lists and single lists are often presented as can be found in Figure 2.1.

Multi-lists are often internally constructed either by machine learning models or by the editorial staff that chooses what goes inside each list [36]. Instead of focusing on creating a "perfect" list of recommendations, users are presented with several choices of categories, which makes it possible to cater to the users' mood or changing interests [35]. It is also evident that some of the content providers set up their multi-list recommendations differently. While some platforms like Netflix de-duplicate their recommendations (make sure that an item appears in a list only once) [13], others omit the de-duplication process, which does not influence the time users spend on making a decision greatly [20]. Overall, there are different approaches to creating recommendation lists, so it is crucial to choose the one that aligns with the platform's goals.

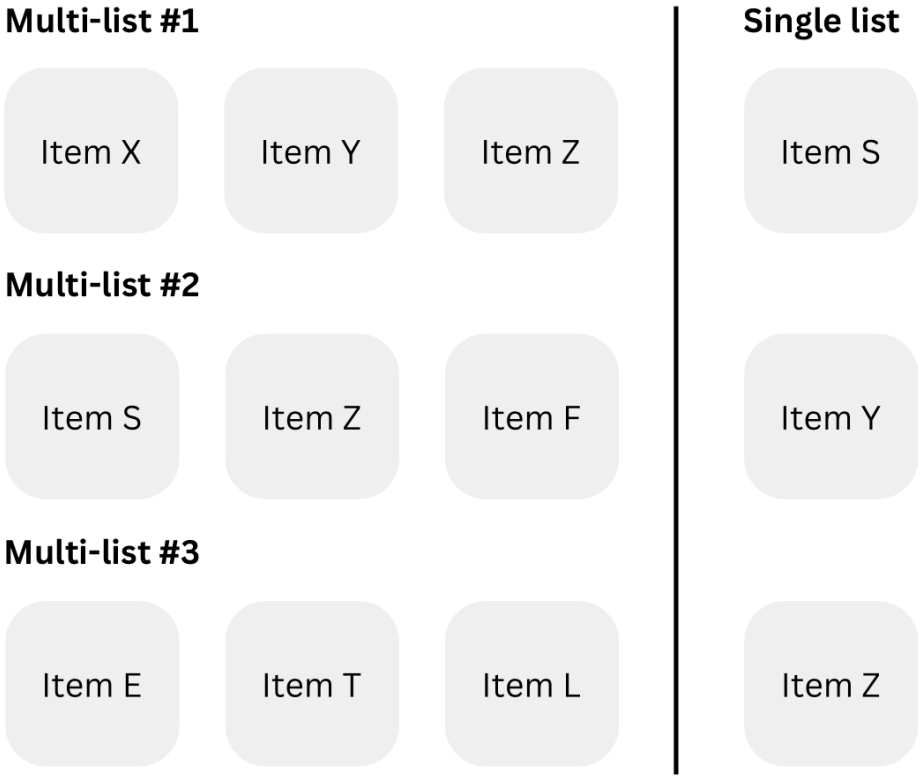

Figure 2.1: The way multi-lists are often presented is on the left-hand side, while the right-hand side shows a way to present a single list.

2.2 Implicit feedback

Another important distinction of this thesis is the nature of the data used for making recommendations. TV 2 Play platform does not collect ratings of movies or series but uses implicit feedback for making recommendations. The only way for a user to express their preferences explicitly is to "like"/"dislike" an item or to add an item to their favorites, also referred to as explicit favorites. Therefore, implicit data are collected indirectly from users via interactions with different items [40]. These may include clickstream data, time spent on the website, purchases, and browsing history. Customers are known to be reluctant to provide explicit feedback. Additionally, implicit feedback does not provide any information on negative preferences, which should be excluded from the recommendations [18]. While implicit data can be a less accurate representation of user interests, the volume of it and the lack of intrusiveness in collecting such feedback holds a lot of potential for its use [33].

With the lack of explicit feedback in the provided data, the focus lies on the representation of users' interests through implicit feedback further in the analysis. This work is based on using implicit feedback data, which namely consist of impressions (what a user has been shown), clicks (what a user has clicked on), and click-through rate (the number of clicks divided by the number of views) for each user.

2.3 Recommender systems overview

Overall, recommender systems have taken over the world of e-commerce and streaming platforms, where predicting users' behavior and potential needs gives an advantage in the competition for customer attention [40]. What started as a tool to help customers overcome the overload problem has evolved into one of the most prominent techniques to appeal to users and increase customer acquisition [39]. Major progress has been made by academia and the industry, allowing us to have a broad range of different algorithms to choose from.

2.3.1 Content-Based recommender systems

The main goal of Content-Based recommenders is to find items similar to what a user has explicitly "liked" in the past based on the features extracted from items [32]. In the movie domain, examples of such features can be movie genres, directors, or movie descriptions, among other things. The stages of building the Content-Based recommender consist of 1) analyzing the content, 2) learning users' profiles, and 3) filtering component [27]. The first stage entails the extraction of relevant item features from the unstructured data, and the representation of said features in a form suitable for the following stages. Then, the user profiles are constructed by analyzing and generalizing the data available about users, including their previous interactions with the items, which can be, for instance, ratings given to a set of movies. This stage is normally performed via a set of machine learning techniques to infer user interests. Finally, the filtering component entails matching the items to user profiles and evaluating the quality of recommendations via similarity metrics. The simplified overview can be found in Figure 2.2.

Generally, Content-Based recommender systems are easy to implement and can be very effective when the amount of data on user preferences is limited [31]. Another advantage of

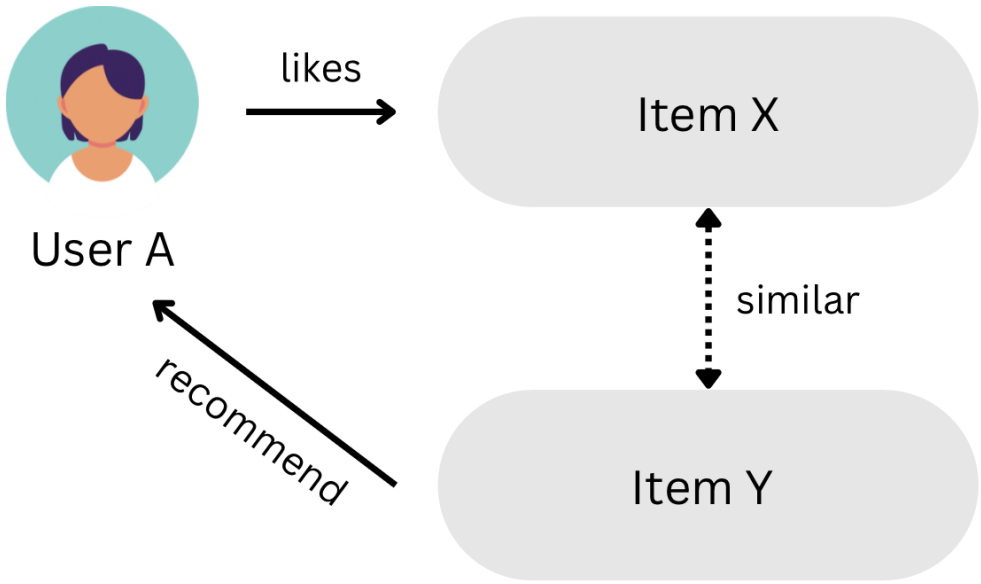

Figure 2.2: High-level architecture of Content-Based Filtering recommender systems.

using a Content-Based approach is that each user has their profile, meaning that if their preferences change over time, Content-Based systems are able to pick up on that [40]. Another key advantage of this approach is overcoming the first-rater problem when none of the users have previously rated new items, given that the new items have sufficient descriptions. On the other hand, some major disadvantages include the need for in-depth domain knowledge in order to extract relevant item features. Also, in contrast to being able to adapt to ever-changing user interests, the Content-Based Filtering approach struggles to expand on the users' existing choices. Consequently, since this approach recommends items similar to what a user has rated, it tends to have quite limited capacity to suggest novel items or something that does not precisely look similar to user interests, which makes the recommendations less diverse. Finally, when a new user is added to the system, the model will take time to be able to learn user preferences and provide reliable recommendations since that user has not given any ratings previously [40, 27, 39]. Some popular implementations of the approach include Nearest Neighbor Classifier, Decision Trees, and Rocchio's Algorithm [32].

2.3.2 Collaborative recommender systems

Collaborative Filtering is a method that relies on the similarity between users to make recommendations. It predicts the utility of an item for a particular user based on other users' feedback. This approach entails building an m x n matrix, where m is the number of users and n is the number of items with the ratings or implicit feedback, e.g., click-through rate, in the matrix cells. Naturally, in state-of-the-art applications, the matrix is very sparse since even active users cannot give feedback on thousands or millions of items, which platforms provide [43]. A high-level overview can be found in Figure 2.3. Collaborative methods are divided into two main groups: memory-based and model-based approaches [43]. A memory-based approach refers to making predictions from the entire user-item collection and can only recommend items to users already in the system. The approach tends to be computationally demanding due to the large amounts of data required to produce recommendations [40, 2]. The main idea is to predict the ratings a user A will give to unseen items by, e.g., averaging other users' ratings. On the other hand, model-based algorithms use the data matrix to learn a model, making predictions for all users, including those not in the data. The model is built using, for instance, Bayesian network, clustering, or rule-based approaches [43]. Combining memory- and model-based algorithms can lead to better predictions in some applications, but naturally, each algorithm has its strengths and weaknesses, depending on the area of use.

Although a widely used approach, Collaborative Filtering is not the solution to everything, as this method does have its limitations. As mentioned previously, sparsity is a big problem for this type of recommenders. Making accurate predictions from few ratings can be challenging, so several approaches exist to battle this problem. An example of such could be Alternating-Least-Squares with Weighted-λ-Regularization, as described by Zhou et al. [52], or Probabilistic Matrix Factorization that scales linearly with the number of observations [42]. This method also struggles to recommend items to new users, as does Content-Based Filtering. Acquiring enough ratings for new users is crucial for making good recommendations. In addition, new items in the collection are often overlooked by the algorithm initially, as there is not enough feedback data to suggest them to specific users. However, Collaborative Filtering is a strong contestant in terms of accuracy and handling large amounts of data and is considered the most popular and widely implemented recommender system [38]. The reasons for that are not having to possess domain knowledge, as is the case with Content-Based Filtering, quality improvement of recommendations as more and more users join and

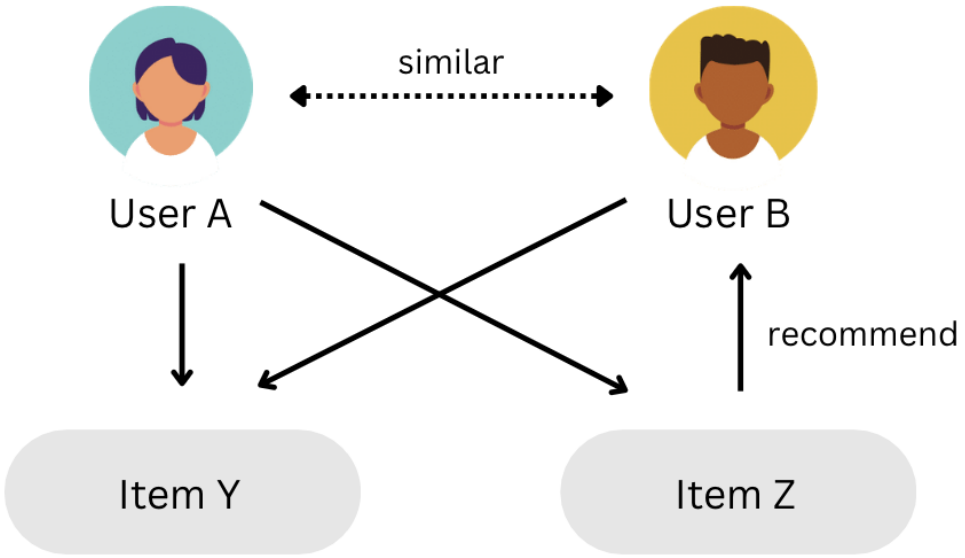

Figure 2.3: High-level architecture of Collaborative Filtering recommender systems.

give feedback, and serendipity or providing users with unexpected yet relevant suggestions, thus, expanding on their existing tastes [28, 40].

GroupLens, Video Recommender, and Ringo were the first systems to use this approach to automate prediction. Further, Amazon and other goods providers have picked up on the technology and started using it on their platforms [2]. When it comes to the most widely used Collaborative Filtering algorithms, there are Neighborhood Models (e.g., kNN Classifier) and Matrix Factorization (e.g., Singular Value Decomposition). From that, many other algorithms originate, namely, Alternating Least Squares and Bayesian Personalized Ranking, which will be discussed later.

2.3.3 Hybrid approaches

Hybrid systems are a combination of Content-Based and Collaborative Filtering. The main goal is to overcome issues of approach A by techniques used in approach B [38]. Overall, building a hybrid system depends on the data available and what limitations of other

recommender systems a developer tries to deal with. Thus, it is difficult to provide a concise overview without going into too much detail. The following is an example of a hybrid approach. A weighted hybrid recommender can be built as a model, which combines the results of several recommenders in the system and gives different weights to each algorithm's recommendations. For instance, P-Tango uses something similar, where both Collaborative and Content-Based Filtering are given the same weight, which adjusts over time based on the users' feedback to provide recommendations [8]. Another example is a hybrid model that, instead of using numeric scores, treats outputted recommendations from each recommender as a set of votes, e.g., the highest ranked item is given five points, and the lowest ranked is given one point. All of the points are combined, allowing to "vote" for the best recommendations and their ranking [31]. Thus, combining several algorithms can prove effective. Still, it is necessary to consider the available resources, both timely and computational, when deciding to implement a hybrid recommender system, as it can require a lot of work. Knowing the strengths and weaknesses of each method considered is crucial, as well as being able to choose the proper approaches depending on the goal of the recommender system and the data obtained.

2.3.4 Multi-Armed Bandits recommenders

Multi-Armed Bandits is a framework for solving the exploration-exploitation problem to make decisions over time under uncertainty [45]. Generally, the Multi-Armed Bandits approach attempts to solve the problem of choosing between several options, where the rewards for each are unknown [5]. By defining a set of arms of a Multi-Armed Bandit recommender to be a set of items from which to recommend to users, the pay-off to be, e.g., ratings for these items, and the decisions to be previous user-item interactions, we can use this approach in the domain of recommender systems. One of the fundamental MAB algorithms is EpsilonGreedy, where a value of epsilon is defined as the degree of exploration [45]. The first stage consists of estimating the average rewards for each arm. Then, by setting epsilon to, for example, 0.3, we say that the model will choose the arm (item) with the highest reward (e.g., rating) 7 times out of 10 (exploitation) and in the other 3 times, the model will choose a random arm (exploration). That way, the algorithm allows for the introduction of novel items into the recommendations while still providing accurate suggestions. The following equation is the depiction of EpsilonGreedy:

$$\textit{Arm to choose at iteration } (i) = \begin{cases} \textit{max reward arm} & \text{with probability 1-}\epsilon \\ \textit{random arm} & \text{with probability } \epsilon \end{cases} \tag{2.1}$$

The downside of this approach is that it continues to recommend non-optimal items, even when it is established that a user does not like them due to the introduced randomness [25]. EpsilonGreedy is considered a parametric non-contextual algorithm, but other implementations of Multi-Armed Bandits include non-parametric (e.g., Thompson Sampling and Random) and contextual algorithms (e.g., LinGreedy and LinUCB). Introducing context to bandits allows to expand on the knowledge about users or a set of items. Thus, rewards for each arm depend on context and can change significantly compared to non-contextual bandits [45]. Detailed descriptions of several other bandit approaches can be found in Section 2.4.

2.4 Choosing recommender system approaches

Among the models discussed previously, Collaborative Filtering and Multi-Armed Bandits stand out the most. By implementing Collaborative Filtering, we can, first of all, directly see how our implementation potentially differs from what the TV 2 Play team has developed since Feed-CF is based on Collaborative Filtering. This approach is also more suitable for the data at hand since there is virtually no way to find out what content is inside each feed, which makes Content-Based Filtering ineffective. However, by having historical data on user-item interactions, we are able to build a model to recommend feeds to users. When it comes to the decision behind choosing the Multi-Armed Bandit recommender system, there are two main reasons for that. The first one is the business need. Researching and implementing new recommender systems is time-consuming, meaning that TV 2 Play developers are limited in what they can try. Moreover, there is no guarantee that this method would outperform Feed-CF, so this thesis aims to investigate whether implementing a MAB recommender is beneficial for the business in the future. Secondly, Multi-Armed Bandits are a strong contestant, as evident from previous studies [25, 24]. Recommendations that provide not only similar items but allow to explore beyond users' interests can potentially increase user engagement on a platform and, thus, should be taken into account.

The following is an overview of the Collaborative Filtering and Multi-Armed Bandits methods upon which we build the models in Chapter 3. For Collaborative Filtering, Alternating Least Squares (ALS), Bayesian Personalized Ranking (BPR), and Logistic Matrix Factorization (LMF) are used, where all of them are based on Matrix Factorization [16, 37, 21]. Matrix Factorization is a technique that allows to represent a user-item matrix by finding two matrices, the product of which equals to it; in movie recommendations, these are user and item matrices that capture underlying features [23], which are depicted in Figure 2.4.

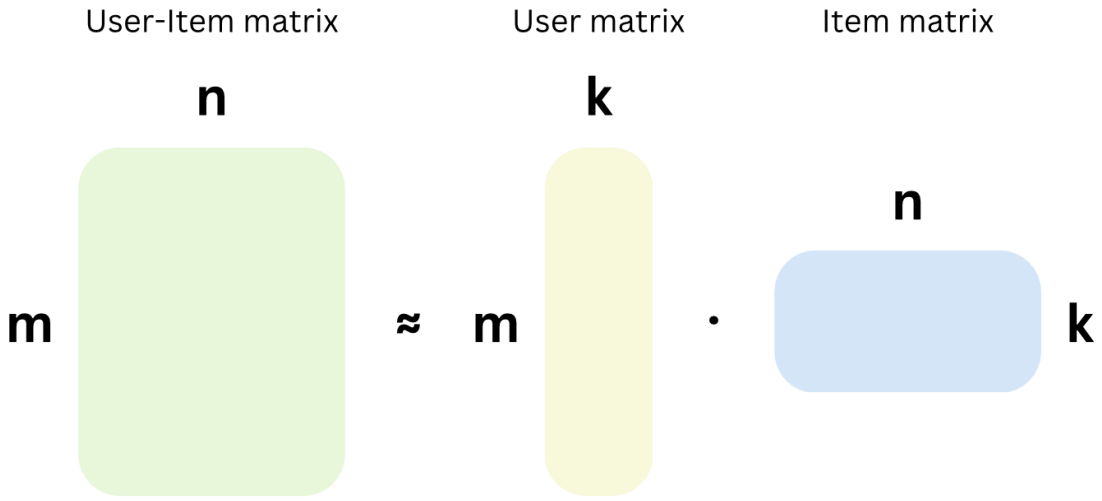

Figure 2.4: Simplified example of Matrix Factorization approach.

Building on top of that, ALS decomposes the user-item interaction matrix into two lower-rank matrices representing user and item latent factors by alternating between fixing one matrix and optimizing the other via least squares regression [52, 18]. The user and item latent factors are the lower-dimension latent space features projected from the original user-item interaction matrix. Similarly to ALS, BPR utilizes matrix factorization and decomposes the user-item matrix into user and item latent factors. However, BPR relies on learning item rankings rather than explicit ratings. For example, if a user has interacted with item A but has not interacted with item B, item A will be ranked higher than item B. This approach maximizes the likelihood of observing the correct order of items in the user's preference list in an attempt to provide the most relevant recommendations at the top [37]. Finally, LMF focuses on representing features of users and items hidden in the data, allowing the recommender system to make predictions based on the extracted latent factors. It is based on

a logarithmic transformation of the user-item matrix, which proves to be efficient in dealing with missing and sparse data [21]. Additionally, LMF is a more general implementation of matrix factorization compared to ALS and BPR, which utilize other techniques in order to make recommendations. All of the methods described above are fit to work on implicit feedback data and, moreover, on large-scale data.

Several MAB models are used in the analysis, namely, EpsilonGreedy, LinGreedy, Random, Thompson Sampling, UCB1, LinUCB, and Softmax. Of the non-parametric non-contextual bandits, there are Random and Thompson Sampling. As the name suggests, a Random MAB recommender outputs recommendations, as the name suggests, randomly and, thus, is essentially the same as a Random recommender system. Thompson Sampling, however, builds up a probability model from the obtained rewards, which are then sampled to choose an arm. Russo et al. suggest the following rationalization of Thompson Sampling with Bernoulli distribution over a time period T:

$$Regret\ (T) = \sum_{t=1}^{T} (\max_{1 \leq k \leq K} \theta_k - \theta_{x_t}) \tag{2.2}$$

where K is the number of arms to choose from, $x_t \in \{1, \ldots, K\}$ is the arm selected at time t, and $\theta = (\theta_1, \ldots, \theta_K)$ depicts success probabilities since the decision of choosing an arm can either be "successful" and return "1" or "unsuccessful" and return "0" [41]. The prior distribution to the success probability of each arm is assigned based on prior beliefs; the posterior distribution of the chosen arm is then updated to include the knowledge gathered after the outcome, which was either 0 or 1. Regret expresses the difference between the decisions that are made and the decisions that would bring the highest reward [5].

Of the parametric non-contextual bandits, EpsilonGreedy (discussed in Section 2.3) and UCB1 are the most suitable for the problem. Upper Confidence Bound MAB recommender, given a history of rewards and decisions for each stage, calculates the upper confidence bound for deciding which arm should be chosen at this stage. While it is considered an optimistic approach, it is important to note that UCB1 performs the best on data that follows Bernoulli or Gaussian distribution [3]. UCB1 allows for the exploration of arms that are not chosen as often by giving them a "boost". Finally, parametric contextual bandits include LinGreedy, LinUCB, and Softmax. LinUCB calculates expected rewards for each arm by using a linear function and, then, by calculating the upper confidence bound for each estimate, makes a

choice. This approach assumes that the rewards are linearly related to a set of features and estimates the parameters of the relationship linearly [7]. LinGreedy is a variant of the EpsilonGreedy algorithm. However, it uses a linear function to estimate the rewards for each arm and tends to focus only on exploitation [6]. Last but not least, Softmax calculates an estimate of each arm's pay-off by calculating the probability of each arm being an optimal choice, which is controlled by the temperature parameter (tau) [46]. The arm is then selected according to the probability distribution. Each of the last three methods uses context when estimating rewards. A short overview of the discussed approaches can be found in Table 2.1.

Algorithm	Description
Alternating Least Squares	Matrix Factorization-based, alternates between fixing the user factors and solving for the item factors and vice versa
Bayesian Personalized Ranking	Matrix Factorization-based, optimizes the user-item pairs by maximizing the posterior probability of the ranking
Logistic Matrix Factorization	Matrix Factorization-based, models the probability of a user interacting with an item using logistic regression
Random	Selects arms independently of rewards, on random
Thompson Sampling	Balances exploration and exploitation by using Bayesian inference to model the uncertainty of the reward distribution of each arm
EpsilonGreedy	Balances exploration and exploitation by choosing an arm randomly with ϵ probability and choosing an optimal arm with 1-ϵ probability
UCB1	Balance exploration and exploitation by maintaining an estimate of the mean reward and confidence bound for each arm and selects the arm with the highest UCB
LinGreedy	A variant of EpsilonGreedy, computes UCB with a linear function and selects arms to exploit with 1-ϵ probability and to explore with ϵ probability
LinUCB	A variant of UCB1, uses a linear function for each arm to map the context to rewards, selects the arm with the highest UCB
Softmax	Normalizes the exponentiated expected rewards to obtain the probability of selecting each arm, then selects an arm based on the probabilities

Table 2.1: A short overview of Collaborative Filtering and Multi-Armed Bandit algorithms used in this work.

Chapter 3

Methodology

This chapter focuses on all the steps that led to the final decision of choosing the best model. It starts off by describing the data and how it was pre-processed and manipulated in Section 3.1, then Section 3.3 introduces the evaluation metrics that were used to describe the recommendations. Finally, Section 3.4 covers the different approaches at hand and what constitutes each of them, as well as the evaluation strategies.

This work is built on quantitative methodology and aims, firstly, to compare the performance of several popular recommender systems with implicit feedback data in offline settings. Then, having identified the best-performing algorithm, the model used by the TV 2 Play streaming platform is compared to the recommender with the highest precision@k from the offline setting in the online environment. By using quantitative research methods, the effectiveness of different recommender systems is examined, namely Collaborative Filtering and Multi-Armed Bandits on implicit feedback, in the case of a video content streaming platform. The results of both offline and online testing are then analyzed using descriptive and inferential statistics. The research design is, thus, experimental in nature.

The research objectives include the following: 1) to explore the effectiveness of Collaborative Filtering and Multi-Armed Bandit recommenders on multi-lists, 2) to compare Collaborative Filtering and Multi-Armed Bandit recommenders on evaluation metrics and identify the best performing algorithm in the offline testing, 3) to compare the performance of Feed-CF to the best performing algorithm by conducting A/B testing in the online settings, 4) to evaluate the results of the A/B testing and suggest the appropriate changes, if necessary.

3.1 Data

The dataset used in this work comes from the TV 2 Play team. It includes data gathered from the kids' page on TV 2 Play between 01.08.2022 and 31.08.2022, as the data analysis took place in September 2022. The raw data consist of 3,856,033 rows and 9 columns, the description of each variable is given in Table 3.1

Column name	Description
date	Date of the recorded interaction
profileId	Anonymous user identificator
appName	App through which TV 2 Play was accessed
pageTitle	The name of the page, which the user has accessed
feedId	Unique multi-list/feed identificator
feedName	Name of a multi-list/feed
feedPosition	Where on the page a multi-list is located
eventAction	Multi-list impression or multi-list click
count	Number of user interactions with a multi-list

Table 3.1: Original dataset columns and descriptions.

Regarding the multi-lists themselves, there are two types of identificators: feedId and feedName. While feedId is completely unique, meaning that multi-lists that have different ids have different content, feedName can be thought of as a group of feedId items that comprise a category, e.g., in Table 3.2 it can be seen that two different feedId, 196789 and 4091, have the same feedName - "Recommended for you". It should be noted that whenever we refer to multi-lists, we mean feeds and vice versa.

feedName	feedId
Recommended for you	196789
Forest, sun, and friendship	7320
Movies from Disney+	190897
All kids' series from A to Z	196097
Recommended for you	4091

Table 3.2: Examples of the relationship between feedId and feedName.

3.1.1 Data cleaning and manipulation

The data pre-processing is performed fully in Python. First of all, the raw data are filtered to leave out any records that do not come from the kids' page of the streaming platform, which results in 1,315,292 rows left to work with. Further, the filtered data are grouped by profileId, feedId, feedName, eventAction, date, appName, and pageTitle. Then, the eventAction is divided into feed impressions, later referred to as views, and feed clicks, also referred to as clicks. The multi-lists with fewer than 100 appearances in the dataset are excluded (582 rows excluded) to ensure that the algorithms are able to pick up underlying trends. After grouping the data again, the sums of clicks and views are calculated for each user and multi-list on each separate day, and the click-through rate (CTR) is computed by dividing the number of clicks by the number of views. Since there are some unavoidable inconsistencies in the data, for some records the number of clicks is larger than the number of views, which is realistically impossible. To avoid any confusion, all data records with a click-through rate over 1.0 are set to have CTR of 1.0. Afterward, having manually edited some of the data entries (e.g., all multi-lists that start with "Fordi du så..."[1] are set to have the same feedName - "Fordi du så X" and the same feedId - 999999), all of the profileId, feedId, and feedName are converted to unique categories, from which result 46 unique feedname entries and 87 unique feedId entries in the data. The matrix density for the feedId is 9.17%, and for the feedName it is 17.21%.

Since two of the algorithms used in Multi-Armed Bandits, Thompson Sampling and UCB1, require feedback data to follow Bernoulli distribution, the data require binarization. Consequently, the click-through rate, which ranges from 0.0 to 1.0 in the original manifestation, is manipulated to have either 0 or 1, depending on whether the score is below or above the click-through rate mean (calculated from the full final dataset), which is approximately at 0.038. Thus, a new column is added to the dataset to contain the binary representations of the click-through rate.

After all of the manipulations, the data are sorted by date, from oldest to the newest entries to ensure that the training is done on the historical data and the future is predicted from the past. The sorted data are finally divided into training, validation, and test sets in 0.7, 0.15, and 0.15 proportions, respectively. When it comes to the hold-out datasets, the

[1] "Because you watched..."

TV 2 Play procedure is mimicked, where the training set consists of two weeks' worth of data entries, so the training dataset is further filtered to only include entries from 08.08.2022 til 21.08.2022. The overview of the final dataset that is used for further training and analysis is presented in Table 3.3. Overall, there are 768,108 data entries and 13 columns to work with.

Column name	Description
profileId	Anonymous user identificator
feedId	Unique feed identificator
feedName	Name of a feed
date	Date of the recorded interaction
appName	App through which TV 2 Play was accessed
pageTitle	The name of the page, which the user has accessed
clicks	Number of times a feed has been clicked on by a user
views	Number of times a feed has been shown to a user
score	Click-through rate for each feed and each user
profile_id	User identificator coded into a category
feed_id	Feed identificator coded into a category
feed_name	Feed name coded into a category
score_binary	Click-through rate represented as binary values

Table 3.3: Variables from the final dataset and their descriptions.

Both the full dataset and the hold-out datasets are saved as .csv files to ensure that each approach and each algorithm are trained and tested on the same data, the results of which can later be compared and reproduced.

3.1.2 Data analysis

To better understand the data at hand and the relationships between the variables, conducting exploratory data analysis is crucial. This was performed in R. Evidently, there were no missing data in the raw dataset. When it comes to duplicates, they were manually treated during the data pre-processing stage, for instance, multi-lists "Serier og filmer fra Disney"[2] and "Serier og filmer fra Disney+" were given the same feedName and feedId.

[2]"Series and movies from Disney"

When it comes to the numerical variables, there are only 3 of them - views, clicks, and score. Hypothetically, there is a positive linear relationship between views and clicks, where the larger the former variable is, the larger the latter variable is supposed to be. Looking at Figure 3.1, we can see that there is indeed a positive linear relationship between views and clicks. Moreover, the correlation coefficient of 0.41 is statistically significant, which means that the larger the value of views is, the larger the value of clicks is. Looking at the outliers, there are 6,275 data entries where the score is equal to 1.0. However, most of those data records have equal numbers of views and clicks, while some others have 0 views, which can be treated as one of the data gathering inconsistencies.

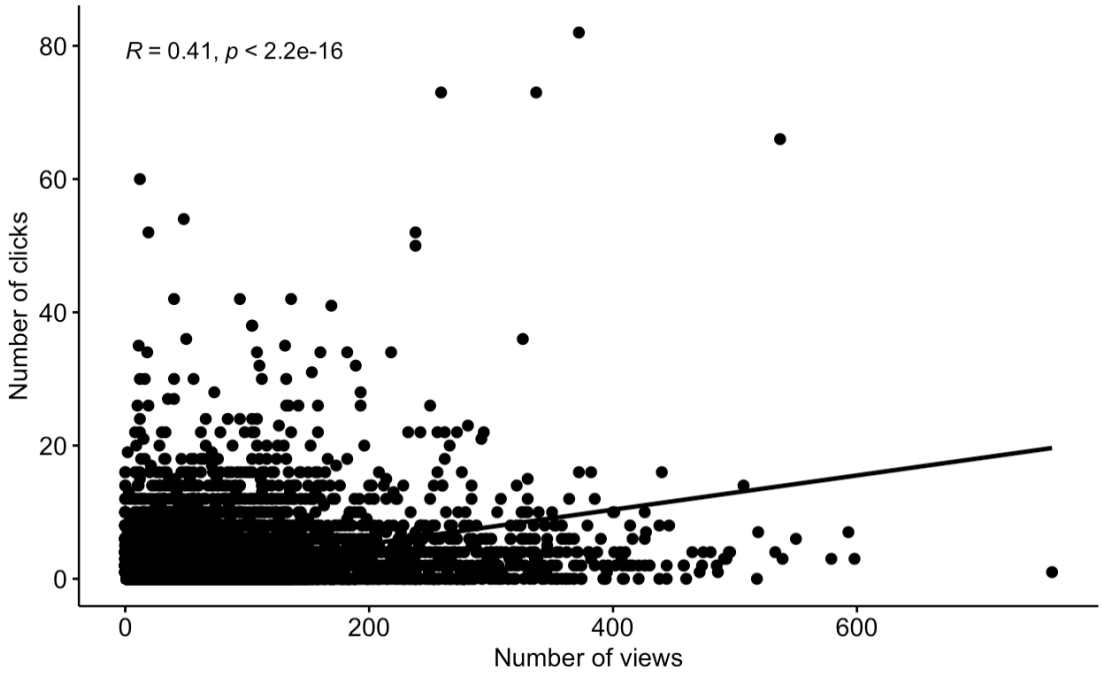

Figure 3.1: A plot that depicts the relationship between the number of views and the number of clicks on the kids' page on TV 2 Play in the period between 08.08.2022 and 31.08.2022. Clicks are moderately positively associated with views.

Focusing on the click-through rate, Figure 3.2 depicts the top-10 most clicked-through multi-lists on the kids' page. One can see that there are several multi-lists that either suggest to re-watch some movies or series, or multi-lists that utilize personal recommendations for users (e.g., "Anbefalt"[3] and "Filmer til deg"[4]), as well as suggesting content that was bought

[3]"Recommended"
[4]"Movies for you"

or rented, for example, "Dine kjøpte og leide filmer"[5].

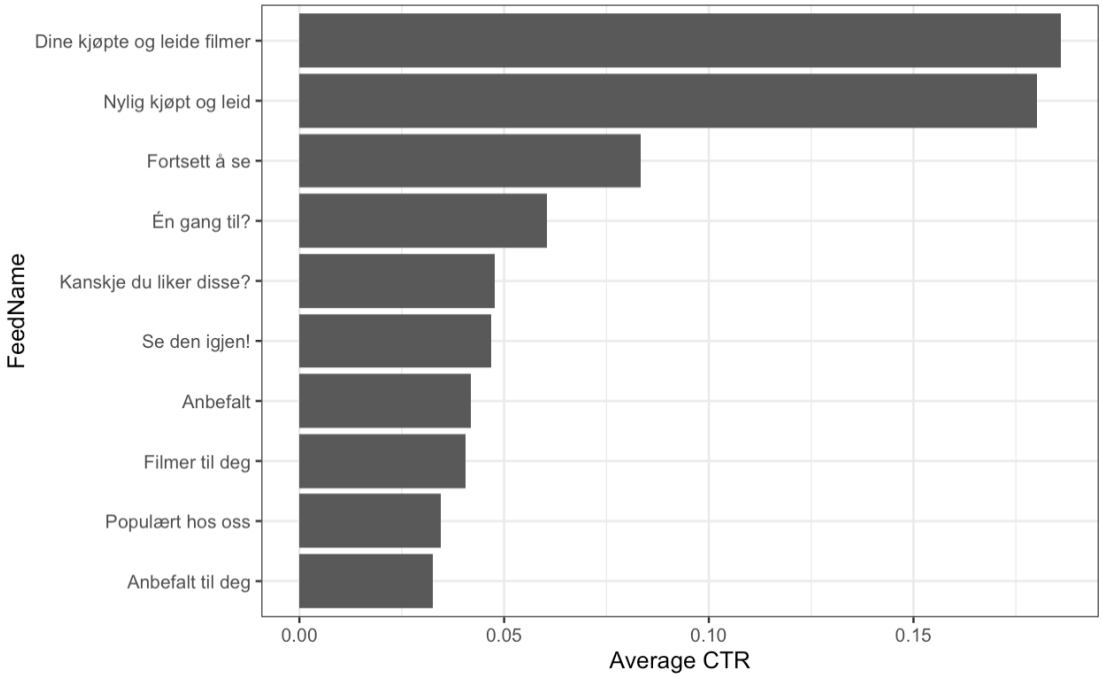

Figure 3.2: A bar chart that shows the top-10 most popular multi-lists (in terms of average click-through rate) on the kids' page on TV 2 Play in the period between 08.08.2022 and 31.08.2022.

Having the date variable, we can make use of it and look into the patterns of what users click on more for each day of the week. Figure 3.3 shows that for almost every day of the week, except Friday, the top-3 multi-lists with the highest average click-through rate stay the same: "Nylig kjøpt og leid"[6], "Fortsett å se"[7] and "Dine kjøpte og leide filmer"[8]. The results are not surprising, as people tend to watch movies or shows fully before moving on to something else, as well as watching something they were so interested in, that they invested money in buying it.

Finally, we can take a look at the top-10 multi-lists that are shown to the users compared to the top-10 multi-lists that users click on most often, shown in Figure 3.4. It can be seen that all multi-lists, except "Serier fra Oiii"[9] from the views plot and "Se den igjen!"[10] from

[5] "Movies that you bought and rented"
[6] "Recently bought and rented"
[7] "Continue watching"
[8] "Movies that you bought and rented"
[9] "Series from Oiii"
[10] "Watch it again!"

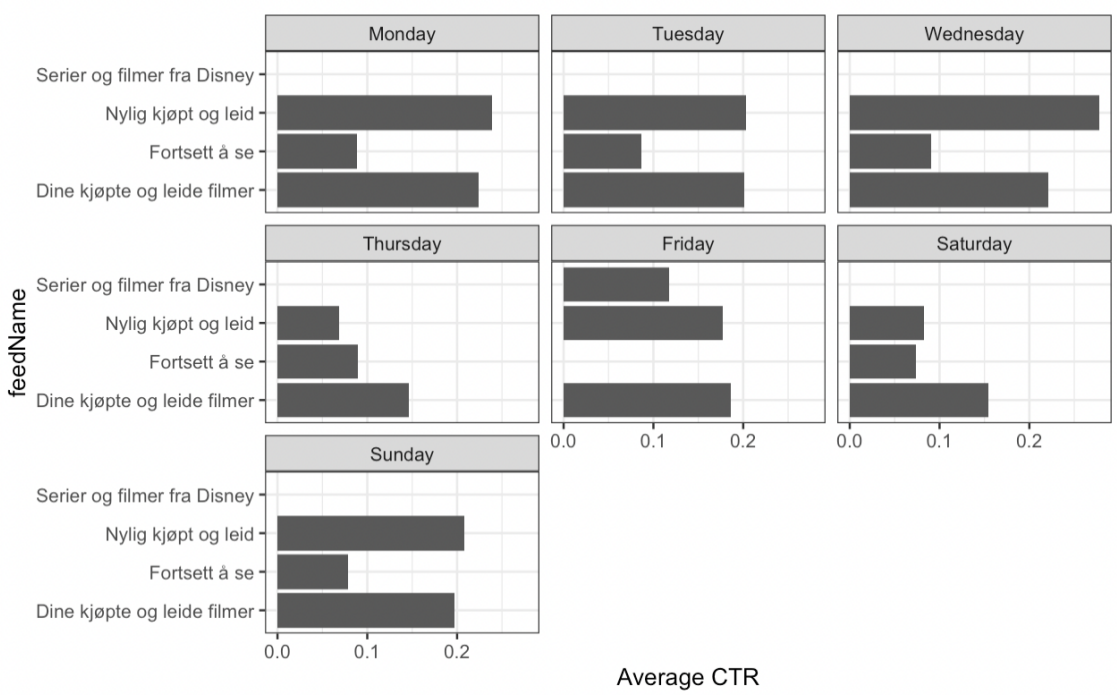

Figure 3.3: A plot showing the top-3 multi-lists with the highest average click-through rate for each day of the week on the kids' page on TV 2 Play in the period between 08.08.2022 and 31.08.2022.

the clicks plot, are identical, which leads us to conclude that the multi-lists that are shown more often are also more clicked.

Some of the following tests will require the use of statistics. ANOVA, also referred to as the Analysis of Variance, allows to test whether at least one of the groups, when comparing more than two independent groups, differs from the rest. However, it only manages to conclude whether the means of the groups are equal or whether the mean of at least one of the groups differs, not providing a detailed description of which group had a different mean [29]. This is where Tukey's HSD can be helpful. Tukey's honestly significant difference controls the family-wise type I error rate. In other words, it determines the probability of obtaining a difference that is as large or larger than the difference between the smallest and the largest means, if the means of all groups were equal [29]. However, the assumptions of ANOVA and Tukey's HSD include data coming from a normal distribution. Looking at the data at hand, neither the independent variables nor the dependent variable come from normal distribution, thus, non-parametric tests have to be used. Kruskal-Wallis and Dunn's tests are appropriate measures when dealing with data that do not meet ANOVA and Tukey's

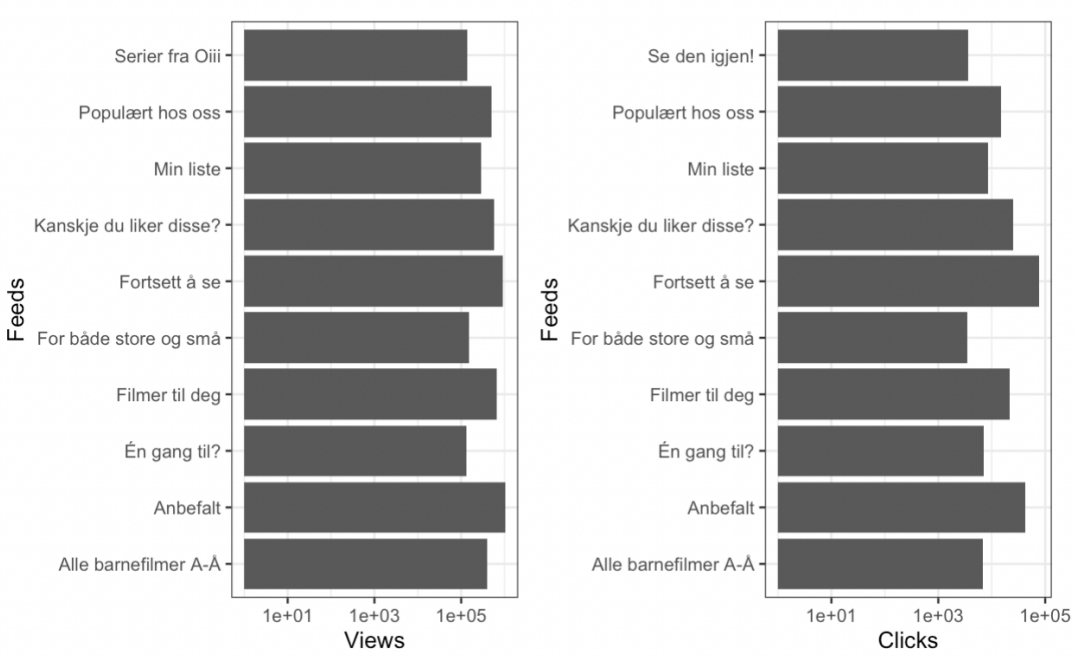

Figure 3.4: A plot showing the top-10 multi-lists shown to the users and clicked by the users on the kids' page of TV 2 Play side-by-side. The y-axis is logarithmically transformed and the axes are then flipped. The data come from the kids' page on TV 2 Play in the period between 08.08.2022 and 31.08.2022

HSD assumptions. Both Kruskal-Wallis and Dunn's are non-parametric tests that can help check whether data come from the same distribution and are applicable to more than two groups [30, 10]. By using those inferential statistics, we can look into the results of the Kruskal-Wallis test with the day of the week as an independent variable and click-through rate as a dependent variable. The null hypothesis states that there are no differences between the medians of click-through rate between days of the week. According to the test, there is enough evidence to conclude that at least one of the groups has a median that is not equal to the rest. To investigate the relationship further, Dunn's test is conducted, which reveals that most of the groups, for example, Monday-Friday and Wednesday-Sunday, have different median click-through rates.

Another thing that could be useful to look into is whether there are any differences in medians of click-through rate between the different apps used to access TV 2 Play. There are 14 unique apps in the dataset. Kruskal-Wallis test is used again, with the null hypothesis that there are no differences between the medians of click-through rate among the different apps. As the p-value is, again, much smaller than 0.05, it can be said that there is enough

evidence to conclude that the medians of the groups are not equal. Dunn's test has shown that most of the pairs have equal medians of click-through rate, meaning that the use of different applications and devices might not play such a big role in further analysis. Overall, we have gotten a good overview of what data look like, as well as what multi-lists tend to be the most popular and, thus, can make an appearance among the recommendations produced by Collaborative Filtering and Multi-Armed Bandits. Further, it can be expected to have a lot of multi-lists that represent re-watching content or continuing to watch the content, as well as some of the multi-lists that relay recommended content.

3.2 Existing recommender system

The recommender system used by TV2 Play is not for public disclosure, but some details are of relevance to this thesis, so the TV 2 team has agreed to provide some general information about their recommender. First of all, our focus is only on the model that provides recommendations as multi-lists, as discussed in Chapter 2. Thus, we do not touch upon the methods used to produce the lists themselves. The existing recommender system, also referred to as Feed-CF, is built on the Alternating Least Squares framework and uses the data for the last two weeks to produce recommendations. In this work, Feed-CF will act as a baseline to compare against the performance of other recommender systems.

3.3 Evaluation metrics

To evaluate the performance of the models, a selection of evaluation metrics is used. Here the focus is both on the accuracy of the recommendations - how similar they are to what a user has previously expressed interest in (user-centric metrics) and on the description of the recommendations themselves (recommendation-centric metrics). By using the notion of k or top-k, we refer to the number of the first k recommendations produced by the algorithm (e.g., if k equals 10, then the first 10 recommendations produced by the model are considered). Each model within this work produces a set of multi-lists for each user in the data, for whom the recommendations are placed from "best" to "worst", depending on the score (predicted click-through rate) that each multi-list receives from a model for that user.

The recommendations are produced by calling the respective function within each library, passing the test dataset, a number of recommendations to produce (k) and context features for MAB approaches when applicable. Below is an overview of the proposed evaluation.

Precision@k. The main evaluation metric, as it directly measures how accurate the provided recommendations are for users [9]. Generally, a higher score indicates better performance. Precision@k is calculated as the number of relevant recommendations in the top-k per user ($\sum Relevant\ recommendations$), where relevant recommendations are the multi-lists that a user has previously clicked on, divided by the number of recommendations (k). The number is then summed over all users in the dataset ($\sum_{u=1}^{N_u}$) and divided by the number of users (N_u).

$$precision@k = \frac{1}{N_u} \sum_{u=1}^{N_u} \frac{\sum Relevant\ recommendations}{k}$$

Recall@k. By measuring recall, it is possible to examine how many of the relevant items were returned in the recommendations [9]. As is the case with precision@k, a higher score on recall@k is considered better. The function takes a number of recommendations provided to each user in the top-k that are relevant ($\sum Relevant\ recommendations$) and divides it by the number of all relevant multi-lists for that user ($All\ relevant\ multi-lists$). This is then summed over all users ($\sum_{u=1}^{N_u}$) and divided by the number of users (N_u).

$$recall@k = \frac{1}{N_u} \sum_{u=1}^{N_u} \frac{\sum Relevant\ recommendations}{All\ relevant\ multi-lists}$$

Mean average precision@k. This metric combines both precision@k and recall@k to evaluate the quality of a ranked list of recommendations. In other words, the higher it is, the more relevant recommendations are placed at the top of the recommendations produced. It is calculated as precision@k for each value of k, which is then multiplied by either 0 or 1, depending on whether the recommendation at the kth step is relevant ($\sum_{i=1}^{k} \text{Precision@}i\ *\ r_i$). The numbers are summed and divided by the number of relevant items in the top-k recommendations ($number\ of\ relevant\ recommendations$), which is referred to as average precision@k. Average precision@k is then summed over all users ($\sum_{u=1}^{N_u}$) and divided by the number of users (N_u), which yields the mean average precision@k.

$$mean\ average\ precision@k = \frac{1}{N_u}\sum_{u=1}^{N_u}\frac{\sum_{i=1}^{k}\text{Precision@}i\ *\ r_i}{num\ of\ relevant\ recommendations}$$

Diversity@k. Diversity@k measures how different the lists of recommendations are across all users. The higher the score, the more diverse the recommendations are, which allows to conclude how recommendations differ between users. A low score indicates that users are being suggested almost the same multi-lists. Diversity@k is produced by calculating cosine similarity for every user's recommendation list at top-k ($cosine\ similarity(Recommendations@k)$), where each row in the matrix with recommendations given to a user is treated like a vector. It then computes the average similarity and subtracts that number from 1.

$$diversity@k = 1 - average\ cosine\ similarity(Recommendations@k)$$

Novelty@k. By looking at the novelty of recommendations, it is possible to see how many of the multi-lists at top-k are completely new to users. New multi-lists are defined as multi-lists that have not been previously shown to the user. In some cases, since these multi-lists are not included in the user-feed interaction data, they can also be interpreted as "irrelevant" and, thus, calculated as precision@k subtracted from 1.

$$novelty@k = 1 - precision@k$$

Catalogue coverage@k. Having a recommender that suggests roughly the same multi-lists over and over again can be disadvantageous for both the users and the streaming platform. It is important that most of the multi-lists get some coverage and exposure in order to maintain diversity and new discoveries. A high score on catalogue coverage@k means that almost all multi-lists are being exposed to users. Catalogue coverage@k is calculated by scanning through all of the recommendations for users and keeping track of which multi-lists have been exposed ($number\ of\ all\ exposed\ multi-lists@k$), or recommended, in this run. The number of those is then divided by the number of all multi-lists in the dataset ($number\ of\ all\ multi-lists$).

$$catalogue\ coverage@k = \frac{number\ of\ all\ exposed\ multi-lists@k}{number\ of\ all\ multi-lists}$$

Average recommendation popularity@k. It is evident from previous studies that some recommenders tend to favor popular items and suggest them to users more often [22]. In order to account for that and be able to calculate the popularity of recommended multi-lists, average popularity@k is introduced, as described by Yin et al [49]. Having popular items in the recommendations is not necessarily disadvantageous but it can lead to popularity bias. The frequency of occurrence for each multi-list is calculated, which is then used to sum them for each recommendation in the top-k recommendations ($\sum_{k=1}^{k} multi-list\ popularity$) and divided by k. The number is summed over all users ($\sum_{u=1}^{N_u}$) and divided by the number of them (N_u).

$$average\ recommendation\ popularity@k = \frac{1}{N_u}\sum_{u=1}^{N_u}\frac{\sum_{k=1}^{k} multi-list\ popularity}{k}$$

3.4 Training and testing the recommenders

After having analyzed the data available, we focus on building the models, which was, again, performed in Python. For Collaborative Filtering implicit[11] library has been used, while Multi-Armed Bandit recommenders have been built using mab2rec[12]. To utilize the data properly, it is crucial to understand that implicit data differ greatly from explicit data, where users can explicitly state where they do not like something, either by giving an item a low rating or putting a "dislike" on an item. Implicit data only allow us to look into what items users have been shown and what has been clicked on, which does not necessarily indicate that the users did not like it. The first step is to determine which hyperparameters are the best for each of the models, for which Bayesian Optimization[13] algorithm is used. It works by taking in a function to optimize as an input, which is, in our case, precision@k. Further, it performs n random runs with different values of hyperparameters in order to find

[11] https://github.com/benfred/implicit
[12] https://github.com/fidelity/mab2rec
[13] https://github.com/fmfn/BayesianOptimization

the highest precision@k and continues to look for the best-fitting hyperparameters for m runs in the space around the best values found in the randomized stage. Bayesian Optimization allows to cut down on the optimization time, as well as introduce elements of randomness to an, otherwise, limited approach, e.g., Grid Search [47].

For the Collaborative Filtering approach, all three available algorithms from the implicit library have been used, namely Alternating Least Squares (ALS), Bayesian Personalized Ranking (BPR), and Logistic Matrix Factorization (LMF). Before starting to optimize, however, the hyperparameter ranges are defined. The hyperparameters and their ranges that were optimized are as follows: 1) number of factors - [1, 64], 2) regularization - [0.0, 1.0], 3) number of iterations - [1, 50] and 4) learning rate - [0.0, 1.0]. It is important to notice that BPR and LMF share the same hyperparameters, while ALS does not take in learning rate as a parameter. The choice behind ranges of hyperparameters is based on previous work done by the TV 2 Play team, which revealed that the best results generally yield from the models with fewer than 64 factors and fewer than 50 iterations; regularization and learning rate are defined at their full range.

Another detail is that there are two models for each algorithm, where one predicts feedId and another predicts feedName. Thus, the outcomes of both are treated slightly differently due to the nature of the output recommendations. Using feedName naturally ensures that there are no duplicates in the recommendations since each multi-list has a unique name on that level, feedId-based recommenders occasionally produce an output where all ids of the multi-lists are unique but the names of them are not. To avoid that and imitate the real world, where users do not get the exact same thing recommended twice, such recommendations are filtered in a way where the multi-lists are compared by names, and the second instance of the recommendations is dropped, if it has already appeared in the recommendation list. To ensure that there are enough multi-lists suggested and there are exactly k multi-lists, for feedId-based models the number of output recommendations is larger.

Having optimized the hyperparameters and having found the best values for them, which is done on the validation dataset, the models have to be trained. The training is performed on the training dataset. Overall, ALS takes approximately 28.5 mins to both optimize and train, BPR takes 18.5 mins and LMF takes 23.5 mins. The evaluation part will be discussed in detail in the next section. For Multi-Armed Bandits, the optimization is performed in the exact same way with the exception of several non-parametric approaches that naturally

do not require any optimization. A plotted example of Bayesian Optimization for epsilon in EpsilonGreedy approach can be found in Figure 3.5. The overview of the hyperparameters and the corresponding ranges can be found in Table 3.4.

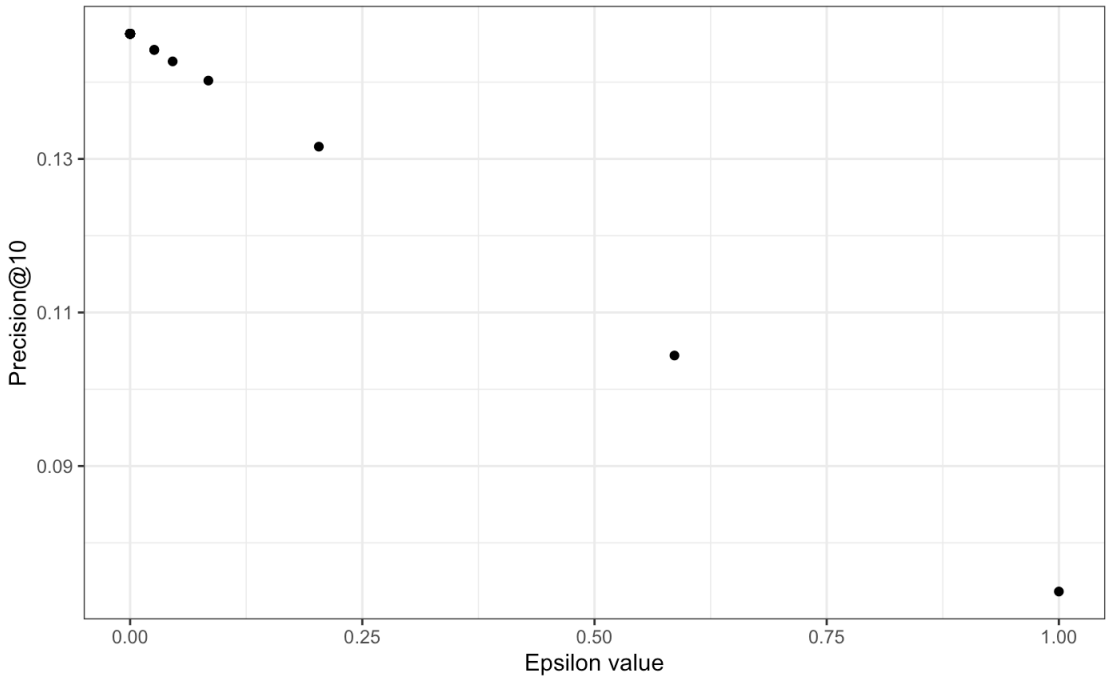

Figure 3.5: A plot showcasing the optimization process for EpsilonGreedy recommender on feedId. The x-axis is the value of the hyperparameter and the y-axis is the value of precision@10 for that epsilon.

Algorithm	epsilon	l2 lambda	alpha	tau
Random				
Thompson Sampling				
EpsilonGreedy	0.0 - 1.0			
UCB1			0 - 10	
LinGreedy	0.001 - 1.0	0.001 - 1.0		
LinUCB		0.0 - 1.0	0 - 10	
Softmax				0.01 - 10

Table 3.4: Multi-Armed Bandits approaches and the corresponding hyperparameter ranges.

A difference between CF and MAB is how some of the Multi-Armed Bandit recommenders make use of context features, specifically, LinGreedy, LinUCB, and Softmax. For building context features, the dates of the interactions have been utilized. Each user has been assigned

either a 0 or 1 based on whether their interactions took place mostly during the weekend or not. For instance, if user A mostly uses TV 2 Play during the weekdays, from Monday to Friday, then that user would have 0 in the 'isWeekend' context feature, while user B, who only uses the platform during the weekend, would have 1. Other than that, the optimization and training processes are identical, with the exception of passing context to the algorithms that are able to make use of it. Another difference lies in the Thompson Sampling and UCB1 approaches that take in click-through rate as a binary variable, as discussed previously. It is worth noting that some of the MAB models have taken significantly less time to optimize and train than CF models, however, the time ranges from several seconds for non-parametric and non-contextual approaches to under 10 minutes for parametric contextual approaches.

3.4.1 Offline testing: first stage

The offline testing stage consisted of evaluating all trained models on the metrics defined in Section 3.3 against each other. To provide a baseline for evaluating the performance, Random and Popularity-based recommenders have also been built, where the former outputs a random list of recommendations to each user and the latter finds the most popular multi-lists in the dataset and recommends the same multi-lists to every user. Given that the optimization and training for all of the models have been identical, the results should be easily comparable to each other. However, there is no straightforward way to tell which model outperforms the other ones, as most of the models with higher precision tend to have higher average recommendation popularity scores and lower diversity scores. The biggest challenge at this stage is to decide which metrics are more important both for the platform and for the users, which ended up being the user-centric metrics like precision@k, recall@k, mean average precision@k, and diversity@k. While recommendation-centric metrics provide valuable input into the quality of recommendations, the trade-off between accuracy and diversity implies sacrificing more precise recommendations to some degree in order to appeal to the users' tastes. To test whether the differences between the results that each recommender provided are statistically significant, a non-parametric Kolmogorov-Smirnov test was used. This test helps to determine whether two samples come from the same distribution [48]. The choice behind the test was made due to the non-normal distribution of precision@k. The null hypothesis for the test is that the values of precision@k, where $k = (1, 3, \ldots, 30)$, for recommender systems, come from the same distribution. The results are described in detail in Chapter 4.

Given these conditions, CF approaches, namely ALS and BPR, have the highest potential when it comes to performance. Additionally, after discussing the way the multi-lists are constructed within the TV 2 Play, it was decided that further training and testing shall focus solemnly on recommending feedId, rather than feedName, even though some models performed better when recommending names of the multi-lists. The decision has been made due to the fact that even if different users are provided with multi-lists that have the same name, those might end up having different content inside, which is more valuable than the seemingly higher accuracy of feedName-based recommenders. Thus, it was decided that further focus should be put on ALS and BPR on feedId, as the two models that have the most potential to compete with Feed-CF.

3.4.2 Offline testing: second stage

As mentioned earlier, two models have made it to the final stage of testing (the overview of model elimination can be found in Figure 3.6). It was decided to add weights to each of the models, increasing the number of competitors up to four and testing them against each other one more time. This stage entails using BM25 weighting scheme, which can be applied to non-textual modalities (e.g., ratings). The BM25 scores are used to re-rank the items in the user-item matrix to reduce the influence of the most popular or frequent items [19]. BM25 was used via transforming the original matrix by adding b and k1 hyperparameters. They are responsible for reducing the impact of popular multi-lists, so, with that in mind, the matrix is recalculated and then summed to the original unweighted matrix. The weighting scheme hyperparameters are optimized for the training dataset only, before optimizing for factors and other hyperparameters. The ranges for k1 and b are from 0.1 to 5 and from 0 to 1, respectively. The tuning is performed in this way to ensure that the weights applied to the matrices afterward are consistent for training, validation, and test sets. Having done that, we define the ranges of the hyperparameters again, identically to the first stage described in Section 3.4.1.

Machine learning models can sometimes struggle to perform well on test data. Thus, understanding the limitations of each algorithm in terms of underfitting and overfitting is crucial. Underfitting refers to a case when a model fails to capture the underlying trends in data and is characterized by having high bias and low variance [34]. On the other hand, overfitting is described as a case when a model has high performance on training data and low

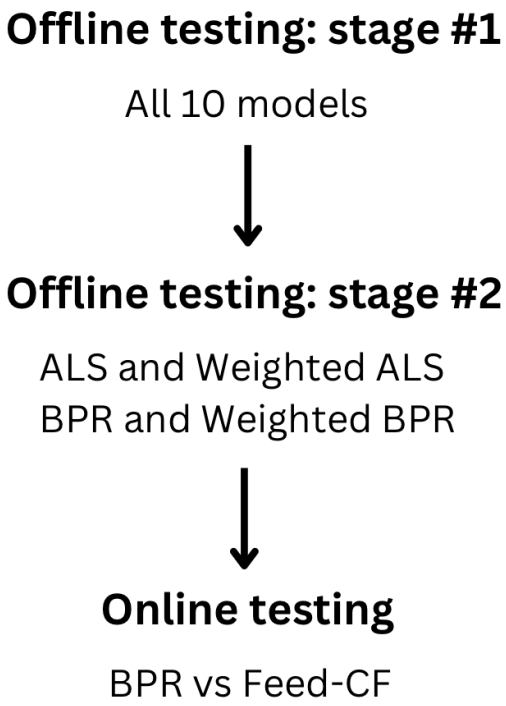

Figure 3.6: A scheme showcasing the model elimination process from offline to online testing.

performance on test data, which happens due to the poor ability to generalize. Such models normally have low bias and high variance [50]. In order to account for potential under- or overfitting, the models have been evaluated both on training and test sets to compare how well the trained models perform on both and check for fit problems.

Model	precision@10		
	Training set	Test set	Difference
ALS	0.48	0.27	0.21
ALS with BM25	0.379	0.262	0.117
BPR	0.439	0.31	0.129
BPR with BM25	0.409	0.271	0.138

Table 3.5: Model comparison to investigate potential underfitting. Alternating Least Squares without bm25 weights might be not generalizing the new data well.

Evaluating recommender systems in terms of bias-variance trade-off can be tricky, especially due to the nature of implicit data, which is rather sparse when converted to a matrix. Generally, the training precision@10 is higher than the test precision@10, which could point

to underfitting. On the other hand, only unweighted ALS seems to have a striking difference in precision@10 (found in Table 3.5), so the results should indeed be treated with care, but it does not pose a direct danger to the output recommendations' quality. The overview of precision@k or training accuracy for different k values can be found in the Appendix, Table A.1. The remedies that could be applied to ML models are not directly applicable to recommender systems (e.g., increasing the model complexity or training for a longer period of time). As there is no more data available to be used for training, it is necessary to treat ALS as a model that potentially generalizes the new data poorly. Ideally, getting more data would be beneficial in terms of increasing test accuracy.

3.4.3 Online testing

After finding the best performing configuration for each model, they have been, once again, compared, and unweighted BPR has outperformed the other ones. Finally, the hypothesis for A/B testing was formulated, which states: on the TV 2 Play kids' page, the Collaborative Filtering model built with Bayesian Personalized Ranking that has 2 factors, with regularization of 0.103, 29 iterations, and learning rate of 0.032 gives better precision at 10 and, thus, outperforms the weighted Collaborative Filtering model built with Alternating Least Squares that has 16 factors, regularization of 0.01, 50 iterations, b of 1 and k1 of 3. The latter model in the hypothesis refers to Feed-CF which is currently used by TV 2 Play for recommending multi-lists.

Thanks to the efforts made by the TV 2 team, they were able to use the findings of this work and implement the unweighted BPR with the hyperparameters described above on their platform, where half of the users are routed towards the existing recommender system, Feed-CF, and another half has recommendations produced by the unweighted BPR. The TV 2 Play team was also courteous enough to set up A/B testing of the unweighted BPR model on the movies page[14] in addition to the kids' page, which allows to compare the results of user engagement between children and adults. It should be noted, however, that the unweighted BPR was not optimized on the movie data specifically for online testing during the course of this work.

Since we were presented with an opportunity to test the unweighted BPR model on the movies page, it is important to understand how the content differs from what is presented on

[14]https://play.tv2.no/film

the kids' page. Focusing on the click-through rate, one can see in Figure 3.7 that the top-10 most clicked-through multi-lists on the movies page consist mainly of personalized content, e.g., "Anbefalt til deg"[15], "Redaksjonen anbefaler"[16] and "De beste filmene nå"[17]. Looking further into the most clicked multi-lists and those that are being shown the most, the two are nearly identical, as depicted in Figure 3.8. It shows the most popular multi-lists by the sum of views and clicks, respectively, from which we can conclude that recommendations based on popularity (e.g., "Populært nå"[18], "Populært fra Nordisk Film+"[19] and "Populært denne uken"[20]) seem to peak users' interests.

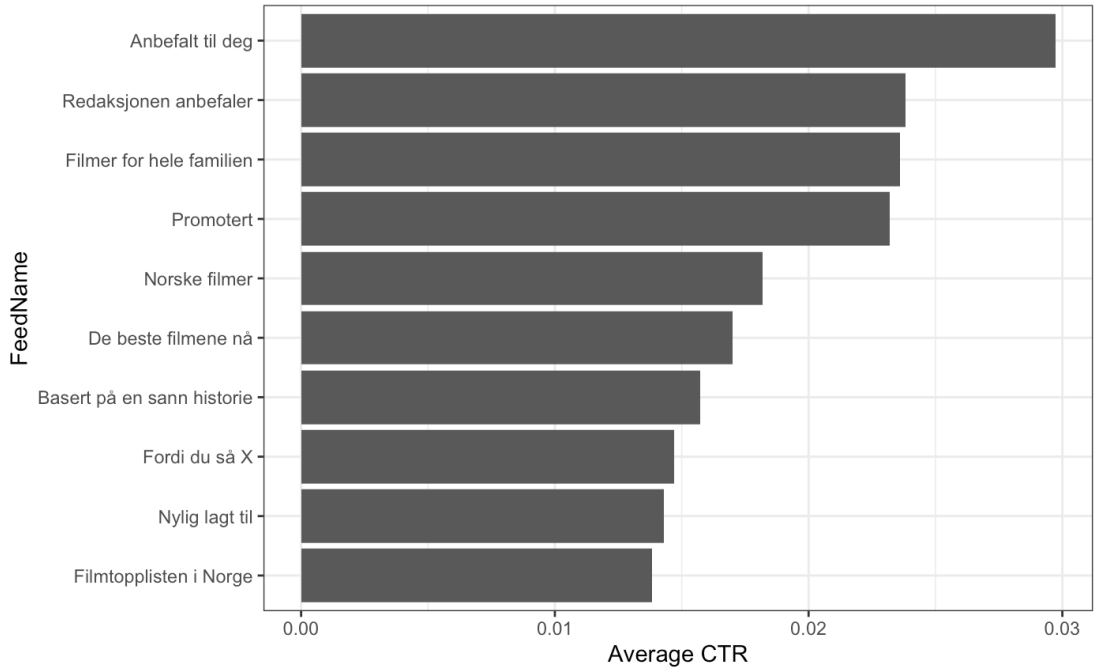

Figure 3.7: A bar chart showcasing the top-10 most popular multi-lists (in terms of click-through rate) on the movies page on TV 2 Play in the period between 08.08.2022 and 31.08.2022.

[15]"Recommended for you"
[16]"The editorial staff recommends"
[17]"The best movies right now"
[18]"Popular now"
[19]"Popular from Nordisk Film+"
[20]"Popular this week"

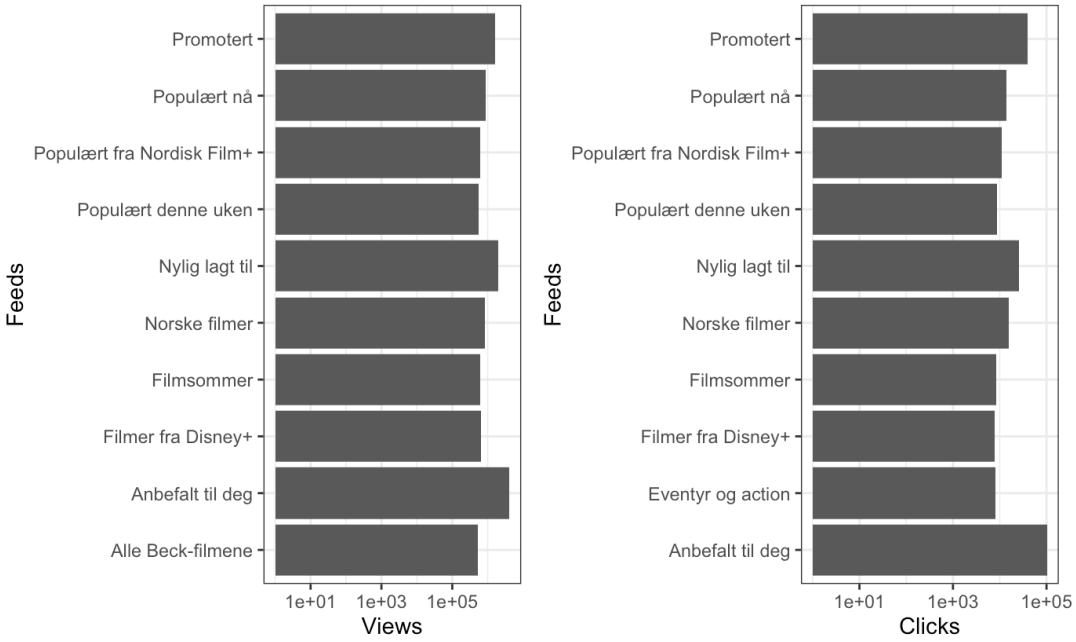

Figure 3.8: A plot showing the top-10 multi-lists shown to the users and clicked by the users on the movies page of TV 2 Play side-by-side. The y-axis is logarithmically transformed and the axes are then flipped. The data come from the movies page on TV 2 Play in the period between 08.08.2022 and 31.08.2022.

In both cases, the A/B testing is set up identically. The online testing stage lasted for two weeks, from 07.05.2023 until 20.05.2023. Both Feed-CF and the unweighted BPR were trained on the data corresponding to the platform page daily. The click-through rate (the sum of clicks divided by the sum of views) for each day for both models was recorded and sent to the author after the complete online testing. In order to investigate the differences between the results, a non-parametric Kolmogorov-Smirnov test was used. The results are further presented in Chapter 4.

Chapter 4

Results

This chapter discusses the results obtained from the models in various settings. First of all, the results from the offline testing are analyzed, yielding the best algorithm, which corresponds to the first research question. Secondly, A/B testing results are presented and analyzed, allowing to answer the second research question.

4.1 Offline testing various recommenders (RQ1)

As discussed in Chapter 3, in the first stage of model training and comparison on the evaluation metrics, the results from all 10 models have been presented. Precision@k is chosen as the main metric to optimize for since it reflects the accuracy of recommendations quite well [44, 17]. In order to capture the fluctuations in performance, various values of k have been used, $k = (1, 3, \ldots, 30)$. However, the focus will be set on $k = 10$, as it is used in similar works. The full overview of metrics at each k can be found in Appendix. Tables 4.1 and 4.2 are found below, which contain the user-centric and recommendation-centric metrics for each algorithm built on feedId in the first stage of offline testing. These also include baseline models (Random and Popularity-based recommenders) and Feed-CF recreated on the data that was used for our models. Since the discussion is centered mainly around the feedId-based models, even though some feedName-based models provide better performance, the focus of this section will be put on the performance of feedId-based models only.

Model	precision@10	recall@10	map@10	diversity@10
Baseline				
Popularity-based	0.31	0.566	0.612	0
Feed-CF	0.25	0.459	0.475	0.568
Random	0.072	0.117	0.152	0.884
Collaborative filtering				
BPR	0.31	0.567	0.735	0.443
ALS	0.27	0.499	0.492	0.503
LMF	0.191	0.332	0.368	0.448
Multi-Armed Bandits				
UCB1	0.187	0.334	0.324	0
Thompson Sampling	0.182	0.33	0.271	0.188
EpsilonGreedy	0.146	0.26	0.196	0
LinGreedy	0.099	0.154	0.172	0.237
LinUCB	0.099	0.154	0.173	0.235
Random	0.075	0.122	0.159	0.878
Softmax	0.074	0.12	0.157	0.878

Table 4.1: Results from the first stage of offline testing on user-centric metrics with $k = 10$. Each model is categorized depending on the approach it belongs to, excluding baseline models. Models within each category are then sorted descendingly by precision@10.

As it is evident from Table 4.1, BPR on feedId performs as well as the Popularity-based model, which yields the best accuracy. Additionally, the BPR model outperforms Feed-CF on precision@10, which means that, potentially, BPR could outperform Feed-CF in online settings as well. ALS comes in second place in terms of precision, also outperforming Feed-CF. When looking at the recall@10 and mean average precision@10 (map@10), BPR shows scores quite higher than other models, including both the Popularity-based model and Feed-CF. In terms of diversity@10, Feed-CF is able to provide recommendations that differ between users better than BPR. Moreover, looking at the recommendation-centric metrics in Table 4.2, one can notice how Feed-CF deals with exposing feeds much better than BPR (catalogue coverage@10), which means that fewer of the feeds are represented in the recommendations produced by BPR. Looking at both user-centric and recommendation-centric metrics, we can see that none of the Multi-Armed Bandits have performed anywhere close to CF or baseline models, thus, MAB will be excluded from the second stage of testing. Considering that BPR and ALS have produced better results than the rest of the models, these two will be taken to the next stage of offline testing.

To test whether the differences between the results that each recommender provided

Model	novelty@10	cc@10	arp@10
Baseline			
Popularity-based	0.69	0.114	0.056
Feed-CF	0.751	0.761	0.028
Random	0.928	1	0.012
Collaborative filtering			
BPR	0.69	0.307	0.037
ALS	0.73	0.92	0.03
LMF	0.809	0.727	0.032
Multi-Armed Bandits			
UCB1	0.813	0.114	0.032
Thompson Sampling	0.818	0.295	0.032
EpsilonGreedy	0.854	0.114	0.025
LinGreedy	0.901	0.909	0.015
LinUCB	0.901	0.216	0.015
Random	0.925	0.932	0.012
Softmax	0.926	0.932	0.012

Table 4.2: Results from the first stage of offline testing on recommendation-centric metrics with $k = 10$. Each model is categorized depending on the approach it belongs to, excluding baseline models. Models within each category are then sorted descendingly by precision@10 from Table 4.1.

are statistically significant, a non-parametric Kolmogorov-Smirnov test was used. The null hypothesis for the test is that the values of precision@k, where $k = (1, 3, \ldots, 30)$, for recommender systems come from the same distribution. According to the test, when performing pairwise comparisons for each recommender, there is not enough evidence to conclude that all of these results for each recommender come from the same distribution and, thus, the differences between them are statistically significant. Presenting each pair would be time-insufficient, so the main results include that CF recommenders provide results that are statistically different from MAB recommenders' results. It was consistent for each pairwise comparison, except for Thompson Sampling and UCB1, which in comparison to some recommenders (namely, Feed-CF and LMF) have shown to provide results that come from the same distribution, meaning that the results are not statistically different. As discussed in Chapter 3, building upon the best models from the first stage, ALS and BPR with BM25 weights are introduced. The results for each model are presented in Table 4.3. The results for each k step can be found in Appendix. Note that ALS and BPR have been trained during the first stage, so the results used in the second stage come from already trained models.

Metric	BPR	BPR with BM25	ALS	ALS with BM25
precision@10	0.31	0.271	0.27	0.262
recall@10	0.567	0.496	0.499	0.487
map@10	0.735	0.719	0.492	0.509
diversity@10	0.443	0.458	0.503	0.516
novelty@10	0.69	0.729	0.73	0.738
cc@10	0.307	0.511	0.92	0.795
arp@10	0.037	0.031	0.03	0.029

Table 4.3: The results from ALS, ALS with BM25, BPR, and BPR with BM25 from the second stage of offline testing. The models are presented in descending order by their precision@10 score.

Again, one can see how BPR performs better than the rest of the models in terms of accuracy, while also having the highest average recommendation popularity@10 score and the lowest catalogue coverage@10. These two metrics can be connected in a way that the more popular multi-lists get recommended to most users, leaving no room for other multi-lists to make an appearance. Overall, it is crucial to be aware of the limitations BPR poses (tends to recommend more popular items), yet it does give the highest scores on user-centric metrics. Thus, this model is compared to Feed-CF in online testing settings. When comparing the models from the second stage of offline testing, we can use the Kolmogorov-Smirnov test once again to check whether the differences in the results are statistically significant. According to the test statistic, there is enough evidence to conclude that the results from ALS, ALS with BM25, BPR, and BPR with BM25 come from the same distribution and, thus, are not statistically different.

Looking at Feed-CF and BPR closer, we can compare the actual recommendations produced between the two of them. Let us assume that user A can be characterized as an inactive user. They have been active once and were shown only one feed before the end of the session, but they have not clicked on it, yielding a click-through rate of 0.0. Figure 4.1 contains the lists of the first 10 recommendations that BPR and Feed-CF give to that user (the names of the multi-lists are translated from Norwegian to English).

It is interesting to notice how different the lists are. Even though a couple of multi-lists can be found in both sets of recommendations, none of the multi-lists are ranked at the same place in BPR and Feed-CF. Thus, the two models have completely different approaches when dealing with users who have not provided enough data for the models to learn. It seems like BPR recommends popular multi-lists, while Feed-CF suggests multi-lists that are less

1. "Recommended"	1. "Series from Oiii"
2. "Continue watching"	2. "Popular with us"
3. "Maybe you will like these?"	3. "Movies for you"
4. "My list"	4. "Series and movies from Disney+"
5. "Because you watched X"	5. "Recently bought and rented"
6. "All kids' movies from A to Z"	6. "Continue watching"
7. "Popular with us"	7. "My list"
8. "Exclusively on TV 2 Play"	8. "Maybe you will like these?"
9. "Popular now"	9. "One more time?"
10. "Catch up before the new season"	10. "Cartoon series for the smallest"

Figure 4.1: The first 10 recommendations provided to user A by the unweighted Bayesian Personalized Ranking (on the left) and Feed-CF (on the right).

popular, yet can yield relevant. Additionally, we consider user B, who has been active on the platform for several weeks, has been shown over 200 multi-lists and has clicked on some of them. The recommendations provided by each model are found in Figure 4.2 (the names of the multi-lists are translated from Norwegian to English). For the user that has provided more feedback, recommendations seem to be more consistent between BPR and Feed-CF. However, both still provide recommendations unique to each approach. Overall, BPR and Feed-CF have some similarities in terms of what is being recommended, yet the tendency of BPR to recommend more popular items opposes Feed-CF, which provides more diverse recommendations.

1. "Recommended"	1. "Continue watching"
2. "Continue watching"	2. "All kids' movies from A to Z"
3. "All kids' movies from A to Z"	3. "Because you watched X"
4. "My list"	4. "Popular with us"
5. "Forest, wilderness and friendship"	5. "Series and movies from Disney+"
6. "Colorful animations from Oiii"	6. "Recommended for you"
7. "Learn and laugh!"	7. "Movies for you"
8. "Popular with us"	8. "Learn and laugh!"
9. "All kids' series from A to Z"	9. "Series from SF Kids"
10. "For both adults and kids"	10. "Honk and drive!"

Figure 4.2: The first 10 recommendations provided to user B by the unweighted Bayesian Personalized Ranking (on the left) and Feed-CF (on the right).

4.2 Online testing Feed-CF and Bayesian Personalized Ranking recommenders (RQ2)

As discussed in Section 3.4.3, the unweighted BPR and Feed-CF have been put on the movies and kids' pages, where they are trained daily on the corresponding data. First of all, let us consider the result from the kids' page. Figure 4.3 showcases the click-through rate for each day over the course of two weeks. It is evident from the plot that Feed-CF outperforms the unweighted BPR, resulting in higher CTR values on the kids' page. To confirm the findings, the non-parametric Kolmogorov-Smirnov test is conducted. The null hypothesis is that the results from both Feed-CF and the unweighted BPR come from the same distribution. Judging by the test statistics, there is not enough evidence to conclude that the samples come from the same distribution, meaning that the results are statistically different.

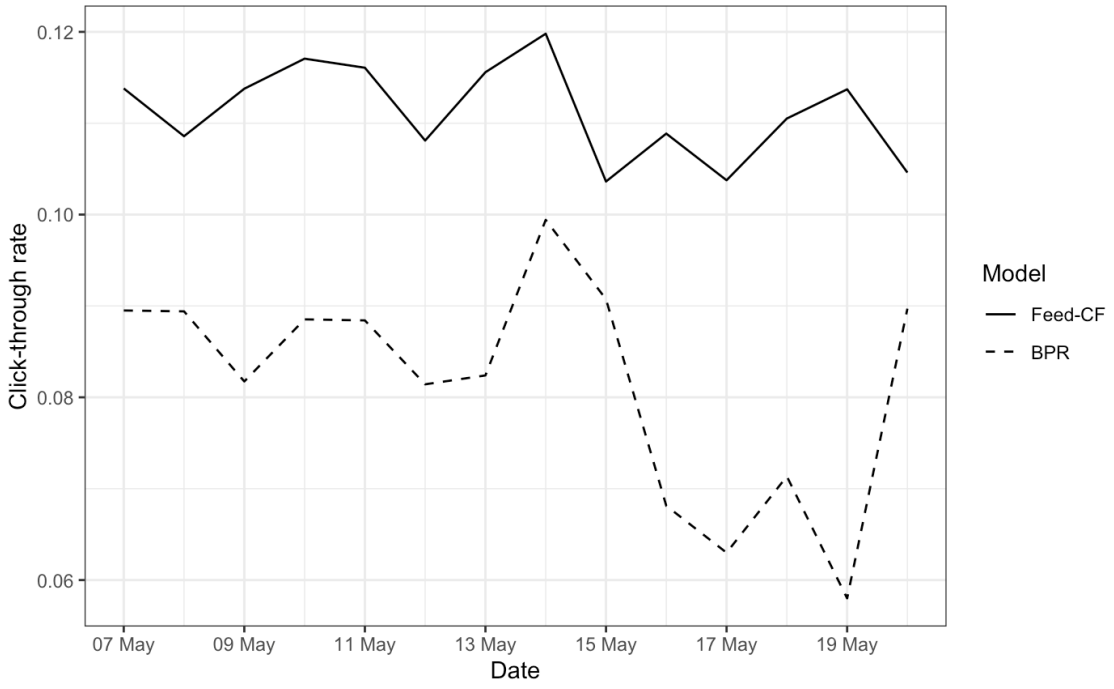

Figure 4.3: A line plot showcasing click-through rate between 07.05.2023 and 20.05.2023 on the kids' page for Feed-CF and BPR.

When it comes to the results from the movies page, Figure 4.4 showcases that the models seem to perform in a similar way. By applying the non-parametric Kolmogorov-Smirnov

test, we can state that there is enough evidence to conclude that the click-through rate results for Feed-CF and the unweighted BPR come from the same distribution and, thus, the differences are statistically insignificant. Overall, the statistical analysis has concluded that the unweighted BPR model has performed worse than Feed-CF on the kids' page and performed as well as Feed-CF on the movies page.

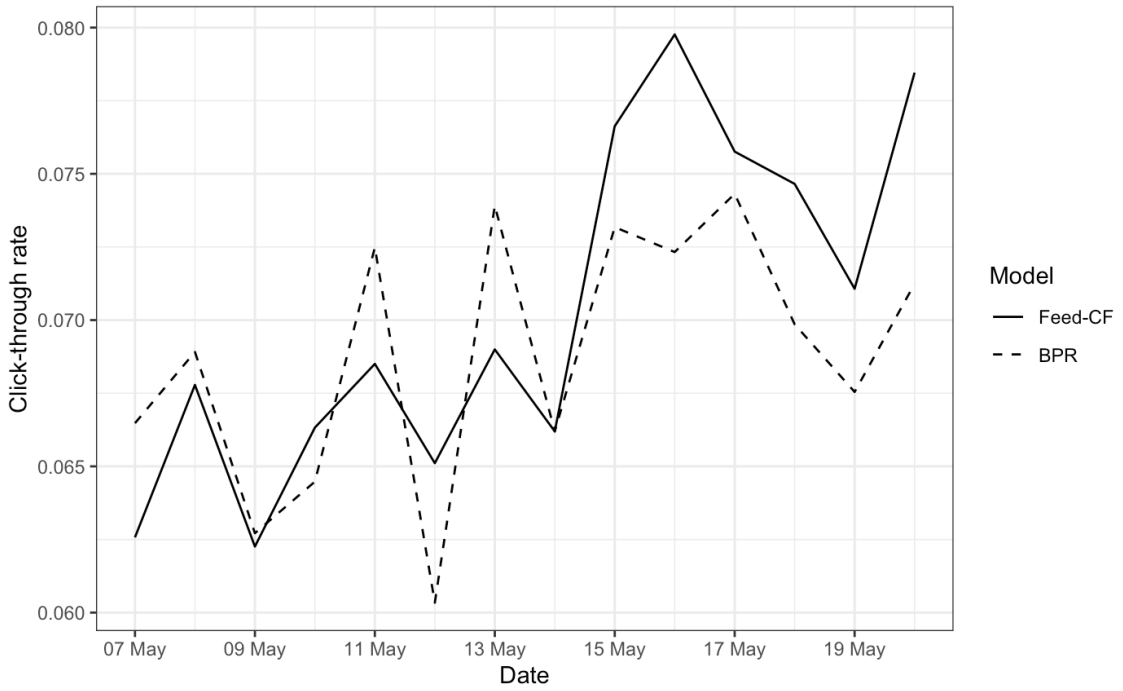

Figure 4.4: A line plot showcasing click-through rate between 07.05.2023 and 20.05.2023 on the movies page for Feed-CF and BPR.

Chapter 5

Discussion and Future Work

The final Chapter provides a summary of the main findings, focuses on the limitations within this work, and directions for future research. Section 5.1 discusses the results and goes into detail to explain them. Further, Section 5.2 brings up ethical considerations that should be considered within this work. Section 5.3 goes on to describe the limitations of the thesis, while Section 5.4 finishes by proposing the directions for future research.

5.1 Discussion

The main goal of this study was to perform a comparison of various recommender systems approaches that are fit to work with implicit feedback and multi-lists and identify whether the algorithm with the highest precision@k can outperform Feed-CF used by TV 2 Play. First of all, the most promising algorithms were chosen according to the task at hand. Having introduced user-centric and recommendation-centric metrics, we have built 13 models in the first stage of offline testing, 3 of which served as a baseline. After training those, the models were compared in performance. ALS and BPR have been chosen as the two models with the most promising results; the two models have been modified by BM25 weighting scheme, which brought the number of competing models to 4. Finally, the unweighted BPR as the best-performing model in the offline testing stage was chosen to be put up against Feed-CF in the A/B testing. Evidently, the unweighted BPR failed to provide better performance than Feed-CF.

5.1.1 Comparing Collaborative Filtering and Multi-Armed Bandit recommenders (RQ1)

The first research question states: "How do different recommender systems compare in the performance of recommending multi-lists on user-centric and recommendation-centric metrics in the offline setting on TV 2 Play data?". The approaches to compare have been narrowed down to Collaborative Filtering and Multi-Armed Bandits, as they show promising results when working with implicit feedback data. However, the use of MAB approaches is under-researched to the best of our knowledge, which is also partially the reason why it was proposed to use in this work.

According to the results from the first offline testing stage (Table 4.1 and Table 4.2), none of the MAB models have come close in performance to ALS and BPR, when comparing them on precision@10, recall@10, and mean average precision@10. While UCB1 and Thompson Sampling did provide the most accurate recommendations among Multi-Armed Bandits, they have still underperformed compared to ALS and BPR. Neither UCB1 nor Thompson Sampling use context features for making predictions, but interestingly, both require the variable for prediction to be binarized, which was previously discussed in Chapter 3. However, while Thompson Sampling shows consistently similar results at each k, UCB1's performance drops drastically for all $k > 1$, which shows unreliability. Otherwise, both models show quite similar results, except when analyzing catalogue coverage and diversity. It appears that while Thompson Sampling provides not very accurate but diverse recommendations and exposes a large part of the feeds during the process, UCB1 tends to have recommendations that are nearly identical for all users. That can be explained by the average recommendation popularity score, where UCB1 seems to recommend more popular feeds at $k = 1$, which gives higher precision since users are often interested in popular multi-lists. Later on, the score on precision for UCB1 drops, as the model seems to recommend items that are very consistent with what a user has previously expressed interest in. In other words, while Thompson Sampling provides diverse and novel recommendations along with some that users are prone to like, UCB1 focuses solemnly on providing feeds that are bound to be perceived well by users, which makes it a safer approach in a sense but strips away any notion of personalization towards users.

Overall, the poor performance provided by MAB approaches can be potentially explained by the data sparsity. While Matrix Factorization-based approaches deal with sparsity rel-

atively well, Multi-Armed Bandits can evidently struggle to produce accurate recommendations from sparse implicit feedback data [11]. Additionally, it is possible that in this particular setting, the MAB models were configured sub-optimally, resulting in poor performance. Section 5.4 explores this topic further. Overall, CF approaches that focus on the trends within the data and exploit user preferences have proven to be more effective when dealing with implicit feedback data and multi-lists than MAB approaches, which rely on exploration-exploitation trade-off to maximize long-term returns. Potentially the MAB approaches are simply unable to capture the changing user preferences from the historical data fast enough and, thus, were not effective. To our best knowledge, there has not been research that directly compares CF against MAB approaches. However, the results of this study cannot be generalized to other platforms beyond TV 2 Play due to the unique approach they take to recommending content to their users and the type of data used. Thus, the first research question can be answered by concluding that in the case of the TV 2 Play platform, when recommending multi-lists on implicit feedback, CF models, namely ALS and BPR, have significantly outperformed MAB models. CF approaches not only have shown higher precision, recall, and mean average precision scores, they have shown to produce relatively diverse recommendations, where a large part of the feeds is exposed.

The next stage consists of comparing ALS and BPR against their variants with BM25 weights. As it can be seen in Table 4.3, the BPR model has shown the highest performance in comparison to the other three models when it comes to user-centric metrics, except diversity. That leads us to the question that was posed before - what goal is considered when developing a recommender system? Having discussed the results from the second stage with the TV 2 Team, it was decided that BPR would be chosen for further online testing. While it does not provide the most diverse recommendations across users, the feeds that are suggested are relevant and can potentially provide increased user engagement. Since it is one of the main goals that any streaming platform seeks, it is also a starting point for further algorithm development, which is discussed in Section 5.4.

5.1.2 Comparing Feed-CF and Bayesian Personalized Ranking recommenders (RQ2)

The second research question states: "How does the chosen model perform on recommending multi-lists in comparison to Feed-CF in the online setting on TV 2 Play data?".

Probably the most interesting part of this work lies in the results that come from the A/B testing performed on TV 2 Play. Having analyzed the click-through rate for Feed-CF and the unweighted BPR models, we can clearly say that the unweighted BPR was not able to outperform the model currently used by TV 2 Play in the context of recommending multi-lists on the kids' page. On the other hand, our model performed as well as Feed-CF when used on the movies page. The findings can be explained by the differences in how users of both pages consume content. Judging by the preliminary exploratory data analysis, it can be noted that users of the kids' page prefer multi-lists that either build upon what they are already watching (e.g., "Fortsett å se"[1]), suggest content they have paid for ("Nylig kjøpt og leid"[2], "Dine kjøpte og leide filmer"[3]) or personalized recommendations ("Filmer til deg"[4]). On the other hand, the users of the movies page are more likely to click on multi-lists that are recommended to them, e.g., "Anbefalt til deg"[5], "Redaksjonen anbefaler"[6] and "Populært denne uken"[7], which are generally popular by themselves and represent a lot of promoted content. These differences can be the reason for the observed performance, meaning that the users of the kids' page do not necessarily pay attention to how popular the multi-lists are, while the users of the movies page do. That leads us to conclude that the unweighted BPR model, which is prone to recommending already popular content, can perform well for the users who consume popular content but not for those with other interests. Thus, the second research question can be answered by saying that when used on the kids' page, the unweighted BPR is not able to provide recommendations that would appeal to the users' diverse preferences. Yet, it does quite well if the users enjoy watching popular content (e.g., on the movies page).

5.2 Ethical considerations

Privacy is one of the biggest concerns when using recommender systems. How is implicit feedback collected, and are users aware that their activity online is monitored to some extent? In order to register on the TV 2 Play streaming platform, a user has to agree to the

[1] "Continue watching"
[2] "Recently bought and rented"
[3] "Movies that you bought and rented"
[4] "Movies for you"
[5] "Recommended for you"
[6] "The editorial staff recommends"
[7] "Popular this week"

terms and conditions, which state that some of the data are used to help improve the user experience by providing personalized tips and recommendations. Thus, a user has to consent to share the data. Looking into it further, none of the demographic data were shared with us during the whole process of collaboration with TV 2 Play team. The only data available about the users are the type of device that was used to access the platform, which makes users practically unidentifiable and, therefore, anonymous. Additionally, the data were handled in accordance with Norwegian Agency for Shared Services in Education and Research[8] regulations. Finally, producing recommendations for children has to be treated with care. Content has to be appropriate, since parental control is not always turned on by the users. Specifically in this work, every multi-list available for recommendations comes from the collection that is specifically tailored towards children. This excludes a possibility of suggesting age-inappropriate, potentially harmful content.

5.3 Limitations

There are several limitations to this study that should be noted. First of all, it only takes into account the users of TV 2 Play streaming platform, thus, the results could differ greatly for other population and should be treated with care before generalizing. This is evident when taking into account the users of the movies and kids' pages, for whom the results differed, even when given by the same model. Additionally, as this thesis is focused primarily on Collaborative Filtering and Multi-Armed Bandits approaches due to the limited computational and time resources, conclusions on performance can only be made on these two approaches solemnly. It is also crucial to remember that, while MAB approaches have not proven to be effective within the scope of this work, it can be attributed to improper model configuration or the type of data used. Thus, further research into how CF and MAB models compare in performance needs to be done in different settings.

The exploratory data analysis has examined the difference in click-through rate during the week via the Kruskal-Willis test, where it was evident that users do have some differences in how they behave on certain days of the week. The possibility of using context for Collaborative Filtering was then considered and implemented via pre-contextual and post-contextual filtering separately but the models were under-performing and, thus, are excluded from the

[8]https://sikt.no/en/home

analysis. However, the information was still used for constructing context features for MAB. It is worth noting that expanding on the results of this work and building a contextual model could improve both the performance in terms of accuracy and the overall user experience.

Another point of this work to be mindful of is popularity bias. It is evident both from the results and the previous research that BPR tends to recommend popular items. That can eventually lead to popularity bias, where a recommender recommends more popular items over and over again, limiting the personalization [22]. Possible solutions for mitigating popularity bias should be considered and implemented in order for the recommender to produce fair recommendations. Moreover, implicit feedback is constructed only from interactions with multi-lists that users have been either shown or clicked on, which does not directly mean that unexposed multi-lists would not have been of interest to the users.

5.4 Future work

One of the most obvious implementations in future work is to expand the use of the algorithm on other pages of TV 2 Play, taking into account the differences in preferences. We have observed how the users of the movies page are more prone to liking popular content and, thus, our model has produced good results. It would be interesting to see what contributes to the users' tastes from further data analysis and use the findings for building models for different pages of the platform. That could help further explain the results that this work came to. Another thing to expand on is the use of MAB approaches. Multi-Armed Bandits have not shown to be a good fit for this particular problem but they hold potential and could be used in other settings, especially when the models get a chance to learn on larger data. Additionally, looking into different context features and acquiring item context features could improve the recommendations produced by the MAB models. Moreover, exploring the ways to mitigate data sparsity problems when using MAB can help to avoid such problems in the future. This work serves as a solid foundation for further research and experimentation within the TV 2 Play platform. Adding to the use of the unweighted BPR model, one could also consider taking a different approach and performing A/B testing to compare Feed-CF to a model with lower accuracy but more diverse recommendations. This could potentially help mitigate popularity bias since most users do tend to click on popular multi-lists more often, as is evident from the data analysis.

Bibliography

[1] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The Unfairness of Popularity Bias in Recommendation. In *13th ACM Conference on Recommender Systems*, 2019. doi: 10.48550/arXiv.1907.13286.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005. doi: 10.1109/TKDE.2005.99.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2):235–256, 2002. doi: 10.1023/A:1013689704352.

[4] Dirk Bollen, Bart P. Knijnenburg, Martijn C. Willemsen, and Mark Graus. Understanding choice overload in recommender systems. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 63–70. Association for Computing Machinery, 2010. doi: 10.1145/1864708.1864724.

[5] Sébastien Bubeck. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012. doi: 10.1561/2200000024.

[6] Ku-Chun Chou, Hsuan-Tien Lin, Chao-Kai Chiang, and Chi-Jen Lu. Pseudo-reward Algorithms for Contextual Bandits with Linear Payoff Functions. In *JMLR: Workshop and Conference Proceedings*, volume 29, pages 1–19, 2014.

[7] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the Fourteenth International Conference on*

Artificial Intelligence and Statistics, volume 15, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[8] Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, and Matthew Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*. ACM, 1999.

[9] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46. Association for Computing Machinery, 2010. doi: 10.1145/1864708.1864721.

[10] Alexis Dinno. Nonparametric Pairwise Multiple Comparisons in Independent Groups using Dunn's Test. *Stata Journal*, 15:292–300, 2015. doi: 10.1177/1536867X1501500117.

[11] Björn H. Eriksson. Contextual Bandits with Sparse Data in Web setting. In *arXiv: Learning*, 2021.

[12] Nicolò Felicioni, Maurizio Ferrari Dacrema, and Paolo Cremonesi. A Methodology for the Offline Evaluation of Recommender Systems in a User Interface with Multiple carousels. In *UMAP '21: Adjunct Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*, pages 10–15, 2021. doi: 10.1145/3450614.3461680.

[13] Carlos A. Gomez-Uribe and Neil Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, 6(4):1–19, 2016. doi: 10.1145/2843948.

[14] Asela Gunawardana and Guy Shani. Evaluating Recommender Systems. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 265–308. Springer US, 2015. doi: 10.1007/978-1-4899-7637-6_8.

[15] Rula A. Hamid, Ahmed S. Albahri, Jwan K. Alwan, Zaidoon T. Al-qaysi, Osamah S. Albahri, Aws A. Zaidan, Alhamzah Alnoor, Abdullah H. Alamoodi, and Bilal B. Zaidan. How smart is e-tourism? A systematic review of smart tourism recommendation system applying data management. *Computer Science Review*, 39:100337, 2021. doi: 10.1016/j.cosrev.2020.100337.

[16] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558, 2016. doi: 10.1145/2911451.2911489.

[17] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004. doi: 10.1145/963770.963772.

[18] 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, 2008. doi: 10.1109/ICDM.2008.22.

[19] Melanie Imhof. BM25 for Non-Textual Modalities in Social Book Search. In *Conference and Labs of the Evaluation Forum*, 2016.

[20] Dietmar Jannach, Mathias Jesse, Michael Jugovac, and Christoph Trattner. Exploring Multi-List User Interfaces for Similar-Item Recommendations. In *Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization*, pages 224–228. ACM, 2021. doi: 10.1145/3450613.3456809.

[21] Christopher C. Johnson. Logistic Matrix Factorization for Implicit Feedback Data. In *Advances in Neural Information Processing Systems*, 2014.

[22] Anastasiia Klimashevskaia, Mehdi Elahi, Dietmar Jannach, Christoph Trattner, and Lars Skjærven. Mitigating Popularity Bias in Recommendation: Potential and Limits of Calibration Approaches. In Ludovico Boratto, Stefano Faralli, Mirko Marras, and Giovanni Stilo, editors, *Advances in Bias and Fairness in Information Retrieval*, Communications in Computer and Information Science, pages 82–90. Springer International Publishing, 2022. doi: 10.1007/978-3-031-09316-6_8.

[23] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434. Association for Computing Machinery, 2008. doi: 10.1145/1401890.1401944.

[24] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 1 edition, 2020. doi: 10.1017/9781108571401.

[25] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 661–670. Association for Computing Machinery, 2010. doi: 10.1145/1772690.1772758.

[26] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003. doi: 10.1109/MIC.2003.1167344.

[27] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based Recommender Systems: State of the Art and Trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer US, 2011. doi: 10.1007/978-0-387-85820-3_3.

[28] Nikos Manouselis, Hendrik Drachsler, Riina Vuorikari, Hans Hummel, and Rob Koper. Recommender Systems in Technology Enhanced Learning. In *Recommender Systems Handbook*. Springer New York, NY, 2011. doi: 10.1007/978-0-387-85820-3_12.

[29] Anita Nanda, Dr. Bibhuti Bhusan Mohapatra, Abikesh Prasada Kumar Mahapatra, Abiresh Prasad Kumar Mahapatra, and Abinash Prasad Kumar Mahapatra. Multiple comparison test by Tukey's honestly significant difference (HSD): Do the confident level control type I error. *International Journal of Statistics and Applied Mathematics*, 6(1): 59–65, 2021. doi: 10.22271/maths.2021.v6.i1a.636.

[30] Eva Ostertagova, Oskar Ostertag, and Jozef Kováč. Methodology and Application of the Kruskal-Wallis Test. *Applied Mechanics and Materials*, 611:115–120, 2014. doi: 10.4028/www.scientific.net/AMM.611.115.

[31] Michael J. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999. doi: 10.1023/A:1006544522159.

[32] Michael J. Pazzani and Daniel Billsus. Content-Based Recommendation Systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, Lecture Notes in Computer Science, pages 325–341. Springer, 2007. doi: 10.1007/978-3-540-72079-9_10.

[33] Ladislav Peska and Peter Vojtáš. Negative implicit feedback in e-commerce recommender systems. In *ACM International Conference Proceeding Series*, 2013. doi: 10.1145/2479787.2479800.

[34] Swathi Pothuganti. Analysis on Solutions for Over-fitting and Under-fitting in Machine Learning Algorithms. *International Journal of Innovative Research in Science Engineering and Technology*, 7:12401–12404, 2018. doi: 10.15680/IJIRSET.2018.0712086.

[35] Behnam Rahdari, Peter Brusilovsky, and Alireza Javadian Sabet. Controlling Personalized Recommendations in Two Dimensions with a Carousel-Based Interface. In *In Proceedings of the Joint Workshop on Interfaces and Human Decision Making for Recommender Systems*, 2021.

[36] Behnam Rahdari, Branislav Kveton, and Peter Brusilovsky. The Magic of Carousels: Single vs. Multi-List Recommender Systems. In *Proceedings of the 33rd ACM Conference on Hypertext and Social Media*, pages 166–174. ACM, 2022. doi: 10.1145/3511095.3531278.

[37] 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*, UAI '09, pages 452–461. AUAI Press, 2009.

[38] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to Recommender Systems Handbook. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 1–35. Springer US, 2011. doi: 10.1007/978-0-387-85820-3_1.

[39] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender Systems: Introduction and Challenges. In Francesco Ricci, Lior Rokach, and Bracha Shapira, editors, *Recommender Systems Handbook*, pages 1–34. Springer US, 2015. doi: 10.1007/978-1-4899-7637-6_1.

[40] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(1):59, 2022. doi: 10.1186/s40537-022-00592-5.

[41] Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, and Zheng Wen. A Tutorial on Thompson Sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

[42] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic Matrix Factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264. Curran Associates Inc., 2007.

[43] 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*, WWW '01, pages 285–295. Association for Computing Machinery, 2001. doi: 10.1145/371920.372071.

[44] Guy Shani and Asela Gunawardana. Evaluating Recommendation Systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011. doi: 10.1007/978-0-387-85820-3_8.

[45] Aleksandrs Slivkins. Introduction to Multi-Armed Bandits. *Foundations and Trends® in Machine Learning*, 12(1):1–286, 2019. doi: 10.1561/2200000068.

[46] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, second edition, 2018.

[47] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. Number arXiv:2104.10201. arXiv, 2021. doi: 10.48550/arXiv.2104.10201.

[48] Benjamin Yakir. Nonparametric Tests: Kolmogorov-Smirnov and Peacock. In *Extremes in Random Fields: A Theory and its Applications*, pages 103–124. 2013. doi: 10.1002/9781118720608.ch6.

[49] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *Proceedings of the VLDB Endowment*, 5(9):896–907, 2012. doi: 10.14778/2311906.2311916.

[50] Xue Ying. An Overview of Overfitting and its Solutions. *Journal of Physics: Conference Series*, 1168(2):022022, 2019. doi: 10.1088/1742-6596/1168/2/022022.

[51] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender

systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010. doi: 10.1073/pnas.1000488107.

[52] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, Berlin, Heidelberg, 2008.

Appendix A

Results from the offline testing

Model	p@1	p@3	p@5	p@10	p@15	p@20	p@25	p@30
Baseline								
Feed-CF	0.812	0.717	0.607	0.414	0.316	0.254	0.209	0.176
Random	0.082	0.083	0.084	0.086	0.086	0.086	0.085	0.085
Popularity-based	0.5	0.469	0.431	0.328	0.267	0.228	0.2	0.182
Collaborative filtering								
ALS	0.93	0.8	0.681	0.48	0.371	0.3	0.249	0.21
ALS with BM25	0.813	0.7	0.568	0.379	0.298	0.252	0.219	0.195
BPR	0.978	0.776	0.64	0.439	0.344	0.292	0.254	0.228
BPR with BM25	0.91	0.725	0.573	0.409	0.338	0.291	0.254	0.223
LMF	0.289	0.26	0.236	0.2	0.178	0.162	0.149	0.138
Multi-Armed Bandits								
Random	0.089	0.089	0.089	0.088	0.088	0.087	0.086	0.085
Thompson Sampling	0.13	0.141	0.138	0.189	0.158	0.148	0.136	0.127
EpsilonGreedy	0.002	0.03	0.13	0.157	0.149	0.138	0.125	0.116
UCB1	0.344	0.116	0.139	0.197	0.152	0.148	0.134	0.126
LinGreedy	0.078	0.106	0.126	0.127	0.114	0.122	0.121	0.11
LinUCB	0.078	0.106	0.126	0.127	0.111	0.121	0.121	0.11
Softmax	0.088	0.088	0.088	0.088	0.087	0.087	0.089	0.094

Table A.1: Precision@k for each feedId-based model presented in this work. The models are tested on the training set.

Model	p@1	p@3	p@5	p@10	p@15	p@20	p@25	p@30
Baseline								
Feed-CF	0.378	0.391	0.316	0.249	0.194	0.161	0.137	0.117
Random	0.071	0.071	0.071	0.072	0.073	0.073	0.072	0.071
Popularity-based	0.499	0.463	0.425	0.31	0.248	0.207	0.179	0.16
Collaborative filtering								
ALS	0.422	0.414	0.34	0.27	0.21	0.174	0.152	0.132
ALS with BM25	0.44	0.437	0.346	0.262	0.202	0.167	0.145	0.126
BPR	0.719	0.587	0.444	0.31	0.225	0.186	0.161	0.146
BPR with BM25	0.694	0.551	0.414	0.271	0.208	0.179	0.154	0.137
LMF	0.258	0.271	0.215	0.191	0.163	0.146	0.129	0.118
Multi-Armed Bandits								
Random	0.076	0.076	0.076	0.075	0.074	0.073	0.072	0.07
Thompson Sampling	0.128	0.137	0.13	0.182	0.145	0.134	0.119	0.108
EpsilonGreedy	0.001	0.024	0.123	0.146	0.134	0.125	0.11	0.101
UCB1	0.34	0.114	0.129	0.187	0.14	0.134	0.118	0.107
LinGreedy	0.078	0.091	0.116	0.099	0.088	0.102	0.104	0.093
LinUCB	0.078	0.091	0.116	0.099	0.087	0.102	0.104	0.093
Softmax	0.074	0.075	0.074	0.074	0.073	0.073	0.074	0.078

Table A.2: Precision@k for each feedId-based model presented in this work. The models are tested on the test set.

Model	r@1	r@3	r@5	r@10	r@15	r@20	r@25	r@30
Baseline								
Feed-CF	0.077	0.229	0.305	0.458	0.522	0.568	0.601	0.611
Random	0.011	0.035	0.058	0.117	0.177	0.234	0.29	0.342
Popularity-based	0.094	0.263	0.402	0.566	0.661	0.721	0.77	0.815
Collaborative filtering								
ALS	0.09	0.251	0.337	0.499	0.567	0.615	0.659	0.683
ALS with BM25	0.092	0.262	0.341	0.487	0.545	0.59	0.63	0.652
BPR	0.145	0.345	0.425	0.567	0.606	0.653	0.698	0.75
BPR with BM25	0.138	0.322	0.392	0.496	0.559	0.633	0.672	0.706
LMF	0.049	0.146	0.194	0.332	0.417	0.49	0.538	0.583
Multi-Armed Bandits								
Random	0.013	0.037	0.062	0.122	0.181	0.237	0.289	0.34
Thompson Sampling	0.024	0.078	0.117	0.33	0.38	0.462	0.505	0.542
EpsilonGreedy	0.0002	0.011	0.111	0.26	0.345	0.434	0.469	0.506
UCB1	0.063	0.064	0.113	0.334	0.367	0.458	0.496	0.533
LinGreedy	0.012	0.04	0.086	0.154	0.203	0.329	0.421	0.445
LinUCB	0.012	0.04	0.086	0.154	0.2	0.328	0.42	0.445
Softmax	0.012	0.037	0.06	0.12	0.177	0.235	0.299	0.375

Table A.3: Recall@k for each feedId-based model presented in this work. The models are tested on the test set.

Model	mp@1	mp@3	mp@5	mp@10	mp@15	mp@20	mp@25	mp@30
Baseline								
Feed-CF	0.378	0.454	0.455	0.466	0.464	0.461	0.451	0.449
Random	0.071	0.12	0.138	0.152	0.152	0.15	0.147	0.143
Popularity-based	0.499	0.674	0.647	0.612	0.598	0.589	0.583	0.578
Collaborative filtering								
ALS	0.422	0.492	0.486	0.492	0.492	0.489	0.476	0.472
ALS with BM25	0.44	0.501	0.499	0.509	0.51	0.507	0.495	0.492
BPR	0.719	0.784	0.773	0.735	0.727	0.708	0.692	0.679
BPR with BM25	0.694	0.77	0.749	0.719	0.697	0.663	0.651	0.642
LMF	0.258	0.4	0.402	0.368	0.354	0.347	0.342	0.338
Multi-Armed Bandits								
Random	0.076	0.128	0.146	0.159	0.159	0.156	0.152	0.149
Thompson Sampling	0.128	0.263	0.3	0.271	0.268	0.256	0.254	0.252
EpsilonGreedy	0.001	0.025	0.132	0.196	0.202	0.192	0.191	0.19
UCB1	0.34	0.342	0.302	0.324	0.328	0.321	0.323	0.324
LinGreedy	0.078	0.136	0.15	0.172	0.167	0.167	0.163	0.162
LinUCB	0.078	0.136	0.151	0.173	0.167	0.166	0.163	0.162
Softmax	0.074	0.126	0.144	0.157	0.156	0.153	0.149	0.145

Table A.4: Mean average precision@k for each feedId-based model presented in this work. The models are tested on the test set.

Model	d@1	d@3	d@5	d@10	d@15	d@20	d@25	d@30
Baseline								
Feed-CF	0.705	0.66	0.629	0.572	0.546	0.506	0.47	0.442
Random	0.989	0.966	0.943	0.884	0.823	0.759	0.693	0.625
Popularity-based	0	0	0	0	0	0	0	0
Collaborative filtering								
ALS	0.715	0.648	0.604	0.503	0.483	0.462	0.433	0.413
ALS with BM25	0.686	0.632	0.569	0.516	0.489	0.472	0.451	0.439
BPR	0.372	0.37	0.476	0.443	0.494	0.482	0.456	0.426
BPR with BM25	0.393	0.388	0.486	0.458	0.457	0.443	0.429	0.41
LMF	0.617	0.563	0.537	0.448	0.366	0.294	0.243	0.202
Multi-Armed Bandits								
Random	0.988	0.963	0.939	0.878	0.816	0.753	0.688	0.621
Thompson Sampling	0.832	0.534	0.374	0.188	0.191	0.192	0.169	0.156
EpsilonGreedy	0	0	0	0	0	0	0	0
UCB1	0	0	0	0	0	0	0	0
LinGreedy	0.264	0.264	0.264	0.237	0.211	0.185	0.159	0.15
LinUCB	0.262	0.262	0.262	0.235	0.209	0.17	0.157	0.139
Softmax	0.988	0.963	0.939	0.878	0.816	0.753	0.683	0.591

Table A.5: Diversity@k for each feedId-based model presented in this work. The models are tested on the test set.

Model	n@1	n@3	n@5	n@10	n@15	n@20	n@25	n@30
Baseline								
Feed-CF	0.622	0.609	0.684	0.751	0.806	0.839	0.863	0.883
Random	0.929	0.929	0.929	0.928	0.927	0.927	0.928	0.929
Popularity-based	0.501	0.537	0.575	0.69	0.752	0.793	0.821	0.84
Collaborative filtering								
ALS	0.578	0.586	0.66	0.73	0.79	0.826	0.848	0.868
ALS with BM25	0.56	0.563	0.654	0.738	0.798	0.833	0.855	0.874
BPR	0.281	0.413	0.556	0.69	0.775	0.814	0.839	0.854
BPR with BM25	0.306	0.449	0.586	0.729	0.792	0.821	0.846	0.863
LMF	0.742	0.729	0.785	0.809	0.837	0.854	0.871	0.882
Multi-Armed Bandits								
Random	0.924	0.924	0.924	0.925	0.926	0.927	0.928	0.93
Thompson Sampling	0.872	0.863	0.87	0.818	0.855	0.866	0.881	0.892
EpsilonGreedy	0.999	0.976	0.877	0.854	0.866	0.875	0.89	0.899
UCB1	0.66	0.886	0.871	0.813	0.86	0.866	0.882	0.893
LinGreedy	0.922	0.909	0.884	0.901	0.912	0.898	0.896	0.907
LinUCB	0.922	0.909	0.884	0.901	0.913	0.898	0.896	0.907
Softmax	0.926	0.925	0.926	0.926	0.927	0.927	0.926	0.922

Table A.6: Novelty@k for each feedId-based model presented in this work. The models are tested on the test set.

Model	cc@1	cc@3	cc@5	cc@10	cc@15	cc@20	cc@25	cc@30
Baseline								
Feed-CF	0.511	0.67	0.727	0.761	0.886	0.932	0.943	0.943
Random	1	1	1	1	1	1	1	1
Popularity-based	0.011	0.034	0.057	0.114	0.17	0.227	0.284	0.341
Collaborative filtering								
ALS	0.727	0.773	0.818	0.92	0.943	0.943	0.943	0.943
ALS with BM25	0.239	0.375	0.58	0.795	0.898	0.92	0.92	0.92
BPR	0.023	0.068	0.148	0.307	0.455	0.602	0.705	0.83
BPR with BM25	0.08	0.181	0.307	0.511	0.67	0.784	0.841	0.886
LMF	0.614	0.693	0.705	0.727	0.739	0.739	0.739	0.739
Multi-Armed Bandits								
Random	0.932	0.932	0.932	0.932	0.932	0.932	0.932	0.932
Thompson Sampling	0.125	0.148	0.182	0.295	0.443	0.455	0.58	0.693
EpsilonGreedy	0.011	0.034	0.057	0.114	0.17	0.227	0.284	0.341
UCB1	0.011	0.034	0.057	0.114	0.17	0.227	0.284	0.341
LinGreedy	0.33	0.625	0.83	0.909	0.932	0.932	0.932	0.932
LinUCB	0.023	0.068	0.114	0.216	0.307	0.375	0.455	0.523
Softmax	0.932	0.932	0.932	0.932	0.932	0.932	0.932	0.932

Table A.7: Catalogue coverage@k for each feedId-based model presented in this work. The models are tested on the test set.

Model	ap@1	ap@3	ap@5	ap@10	ap@15	ap@20	ap@25	ap@30
Baseline								
Feed-CF	0.035	0.039	0.03	0.028	0.022	0.018	0.016	0.014
Random	0.011	0.011	0.011	0.012	0.012	0.012	0.012	0.011
Popularity-based	0.091	0.085	0.078	0.056	0.043	0.036	0.031	0.027
Collaborative filtering								
ALS	0.036	0.041	0.032	0.03	0.024	0.02	0.017	0.015
ALS with BM25	0.041	0.043	0.033	0.029	0.023	0.019	0.016	0.014
BPR	0.091	0.075	0.053	0.037	0.026	0.021	0.018	0.017
BPR with BM25	0.087	0.071	0.05	0.031	0.023	0.02	0.017	0.015
LMF	0.04	0.045	0.035	0.032	0.026	0.024	0.021	0.019
Multi-Armed Bandits								
Random	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.011
Thompson Sampling	0.024	0.025	0.023	0.032	0.025	0.023	0.02	0.018
EpsilonGreedy	0.0002	0.004	0.022	0.025	0.022	0.021	0.018	0.017
UCB1	0.061	0.021	0.022	0.032	0.024	0.022	0.02	0.018
LinGreedy	0.012	0.016	0.018	0.015	0.014	0.016	0.017	0.015
LinUCB	0.012	0.016	0.018	0.015	0.013	0.016	0.017	0.015
Softmax	0.012	0.012	0.012	0.012	0.012	0.012	0.012	0.013

Table A.8: Average recommendation popularity@k for each feedId-based model presented in this work. The models are tested on the test set.