Improving Recommender Systems Algorithms for Personalized Music Video Television by Incorporating User Consumption Behaviour and Multiple Types of User Feedback

Master Thesis

Reza Aditya Permadi

Improving Recommender Systems Algorithms for Personalized Music Video Television by Incorporating User Consumption Behaviour and Multiple Types of User Feedback

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Reza Aditya Permadi

Multimedia Computing Research Group
Faculty EEMCS, Delft University of Technology
Delft, The Netherlands
www.ewi.tudelft.nl

XITE Networks International
Spijkerkade 28, 1021JS
Amsterdam, The Netherlands
www.xite.nl

Improving Recommender Systems Algorithms for Personalized Music Video Television by Incorporating User Consumption Behaviour and Multiple Types of User Feedback

Author: Reza Aditya Permadi
Student id: 4612612
Email: reditya@gmail.com

Abstract

This thesis explores the effects of incorporating user consumption behavior and multiple types of user feedback to improve recommender systems for personalized music video television. An industrial use case is made possible by the availability of anonymized user interaction data on *curation-based* personalized music television system provided by XITE, a music video television broadcasting company in Amsterdam. The characteristics of the curation-based system motivates us to explore the effects of user behavior and feedback on two tasks: session reranking and like prediction task. For the session reranking task, an improvement, in terms of Mean Average Precision (MAP), is achieved by leveraging behavior toward playback of repeated item consumption, together with the implicit user preference which is inferred from personalized average playback ratio for each video. Three types of feedback are used for the 'like' prediction task: explicit feedback when user presses like on a video, and implicit feedback in the form of skipping and watching a video completely. A multi-level sampler within Bayesian Personalized Ranking algorithm is used to exploit those types of feedback, and an improvement is obtained compared to using only one type of explicit feedback. Finally, considering common behavior that people often turns on the television while not actively paying attention to it, we show that performing heuristic cut-off, by only considering few music videos watched completely after an active action is taken by the user on the system as positive implicit feedback, could improve the MAP compared to assuming positive implicit feedback for all videos watched completely by the user.

Thesis Committee:

Chair:	Prof. Dr. Martha Larson, Faculty EEMCS, TU Delft (*supervisor)*
Company supervisor:	Dr. Bouke Huurnink, XITE Networks International
Committee Member:	Dr. Nava Tintarev, Faculty EEMCS, TU Delft

Preface

This thesis marks the end of an amazing 2-years journey to pursue Master of Science degree in Computer Science at TU Delft. Personally, the journey has so many ups and downs, and I will not be able to manage them without the blessings from Allah SWT, which I feel really grateful. I would like to thank my family: my wife, my parents, and parents in-law, for their unconditional support and countless pray for my success. Also to my little daughter, Alya, and my newborn twins, Rafa and Arzan, who always brighten up my day.

Throughout the process of finishing this thesis, I have received support from many people. First, I would like to thank my supervisor, Prof. Martha Larson, for accepting me to conduct this thesis under her supervision. Martha always provides valuable research advice and gives a lot of freedom for me to work on what interests me the most. Her critical yet motivating remarks everytime we discuss the progress of my thesis has helped me shape my research attitude.

I would also like to thank Dr. Bouke Huurnink for the amazing opportunity to do research at XITE. Along the way, Bouke has always been available for discussions and regularly gives thorough critical feedback into the results of my experiment and the writing of this thesis. I would like to thank Dr. Nava Tintarev for introducing me to Bouke at the end of 2017, which opened up a way leading to this thesis, and for accepting to be in my thesis committee.

I also want to thank my colleagues at XITE (Alessandro, Roman, and Natalia) for our insightful discussions everytime I presented my work at the office. To the members of RecSys-EWI meeting for their critical feedback during the initial version of this thesis, and to my fellow friends at TU Delft for all the memorable moments together tackling the ups and downs of this journey. Last but not least, to Lembaga Pengelola Dana Pendidikan (LPDP-RI) for their generous financial support needed to pursue my study.

Reza Aditya Permadi
Delft, The Netherlands
September 2018

Contents

Preface iii

Contents v

List of Figures vii

1 Introduction 1
- 1.1 Curation-based Personalized Music Video Television . . . 2
- 1.2 Challenges . . . 3
- 1.3 Research Questions . . . 5
- 1.4 Contributions . . . 7
- 1.5 Thesis Structure . . . 8
- 1.6 Publication . . . 8

2 Background 9
- 2.1 Recommender Systems . . . 9
- 2.2 User Feedback . . . 10
- 2.3 Algorithms . . . 11
- 2.4 Repeated Item Recommendation . . . 17
- 2.5 User Mode Detection . . . 18
- 2.6 Case study: XITE Personalized Music Television (PMT) . . . 18

3 Approach 21
- 3.1 Evaluation Metrics . . . 21
- 3.2 Session Reranking Task . . . 24
- 3.3 Repeated Items Consumption Behaviour . . . 29
- 3.4 Like Prediction Task . . . 31
- 3.5 User Attention Prediction . . . 34

4 Implementation and Data Analysis 39
- 4.1 Dataset . . . 39

4.2 Implementations of the Recommender Algorithms 43
4.3 Experimental Setup 44
4.4 Empirical Analysis of Repeated Item Consumption 46

5 Results **53**
5.1 Session Reranking Evaluation Benchmark 53
5.2 Incorporating Satiation 56
5.3 Incorporating Implicit and Explicit Feedback 59
5.4 Incorporating User Attention Prediction 62
5.5 Recommendation List Analysis 66

6 Conclusion and Future Work **73**
6.1 Conclusion 73
6.2 Recommendations 75
6.3 Future work 76

Bibliography **79**

A Confidential Information **85**
A.1 Dataset 85

B Paper Contribution **87**

List of Figures

2.1 A screenshot of PMT channels 19
2.2 A screenshot when the user presses like 19

3.1 Illustration on the session assignment from the original dataset 24
3.2 Offline evaluation strategy 27
3.3 Time-based split of the dataset into folds of training and test set 27
3.4 Illustration on music videos candidates selection and evaluation 28
3.5 Proposed sampling strategy 33
3.6 Illustration of using different play cut-off threshold 36

4.1 Cumulative distribution function (CDF) for videos and users. In (a), only few users watch a large number of videos, and in (b), only few videos are watched by lots of users. 41
4.2 Distribution of average playback percentage. Most videos are either skipped in the beginning of their playback or completely played. 42
4.3 Frequency and cumulative density function (CDF) of duration for skipped videos. For clarity of the graph, the duration is capped on 300 seconds. 43
4.4 Effects of different ratio of consumption on empirical probability of playing an item again 48
4.5 Ratio between different number of repeated item consumption 49
4.6 Correlation heatmap of recency exponential factor and empirical probability of consuming the item again 50

5.1 MAP performance across different consumption ratios. The arrows and numbers located at consumption ratio of 0.1 and 1.0 indicate the absolute and relative difference between the best model and random. 58
5.2 MAP performance per iteration for different values of sampling ratio for 'like' prediction task, performed on all items which have been seen by the user in the test-set, but have not been liked by the user in the training set. 60

5.3 MAP performance per iteration for different values of sampling ratio for 'like' prediction task, performed on all items which have been presented to the user during the test set period, but *have not been presented* to the user during the training period. 61
5.4 MAP performance considering different play cut-off threshold for the 'like' prediction task using multi-level sampler where the positive items are taken from items with average playback ratio more than 0.5. The evaluation is performed on all items that have not been liked yet by the user. 63
5.5 MAP performance considering different play cut-off threshold for the 'like' prediction task using multi-level sampler where the positive items are taken from items with average playback ratio more than 0.5. The evaluation is performed on all *unseen* items that have not been liked yet by the user. 63
5.6 MAP performance considering different play cut-off threshold for the 'like' prediction task. The evaluation is performed on all items that have not been liked yet by the user. 65
5.7 MAP performance considering different play cut-off threshold for the 'like' prediction task. The evaluation is performed on all *unseen* items that have not been liked yet by the user. 65
5.8 Percentile distribution of popularity metrics over different models and folds . . 68
5.9 Mean absolute and normalized popularity for various number of minimum 'like' across two models : BPR and ML-BPR. The analysis is performed on the first fold of our dataset. 69
5.10 Percentile distribution of popularity metrics over different models and folds . . 70

Chapter 1

Introduction

Nowadays, people are given abundant choices when they explore items in online digital services, such as in the domain of books, music, and movies. The variety of items presented to them might be overwhelming at some points, which motivates the need for the service provider to give a convenient mechanism for the customer to find the products that they want. Typically, the service provider offers a way for customers to search contents. While it might be a viable option, the search process carries an additional step to view or enjoy contents. For instance, in the domain of video and TV, it is even reported that consumers spend almost an hour searching for contents everyday [12]. For both business and consumer perspective, this time spent on searching might be better off used by customers to enjoy contents on the platform altogether. This brings an additional challenge, how does the service provider understand apriori what the user wants to watch?

Recommender systems have emerged in the past years as a way to address the information overload problem in various domains [31]. By understanding customer past behavior and analyzing complex user and item interactions with the system, the service provider might be able to recommend relevant products to their customers. From the user's perspective, successful recommender engine might increase customer's satisfaction and engagement. Moreover, from the business perspective, it is reported in [13] that personalized recommender systems can effectively reduce customer's churn, which reduces the cost needed for Netflix to acquire new customers to replace the churned ones. Over the years, this reduction saves Netflix more than $1B every year [13].

In general, there are two popular families of recommender system algorithms: collaborative filtering and content-based [1]. Collaborative filtering algorithm exploits vast amounts of user and item interaction data and makes it possible to learn from the wisdom of the crowd. The recommendation list produced by this algorithm is often explained to the users by using sentences like: *"others also watched or enjoyed these items"*. Content-based methods, on the other hand, are performed by analyzing features derived from user or item properties, often described with sentences like: *"similar items for the products that you are currently watching are.."*.

As our study focuses on the recommender systems for music video television, we highlight several notable applications of recommender systems in similar services. Spotify has

developed products called Discover Weekly[1], a personalized weekly-released playlist tailored to each user. Pandora has recently released sets of personalized themed playlists[2], giving options for users to enjoy a wide variety of playlists under different themes such as 'Focus' and 'Your Energy' soundtrack. Last.fm has contributed to the advancement of studies in music recommender system by providing a publicly available API, enabling researchers to analyze user and item interactions on their platform. All of this is evidence that now and in the future, digital companies will compete to provide better recommendation and personalization products to their customers.

1.1 Curation-based Personalized Music Video Television

In the 1990s, MTV, a popular US-based music television channel, became an important part of pop and youth culture by releasing periodic charts of the most popular music videos, and streaming them on the television. On a personal note, I watched and was inspired by these channels in my teens period. Music videos offer a different perspective of enjoying music because we can see what the artists look like, and they are also visually pleasing and entertaining.

With the rise of the internet and mobile device usage, linear channels like MTV encounter new sets of challenges compared to their competitors. For example, a platform like YouTube offers easier access for users to watch music video online without the constraint of time and location. Users can watch music videos on their mobile phones, computers, and televisions. Nevertheless, a TV channel has a competitive advantage over platform like YouTube. The steps taken to enjoy contents are typically fewer on television. The TV is just turned on, a couple of steps pressing the remote to select a channel and contents are directly played on the screen without the fuss of selecting and browsing items, or typing some queries. While it is faster, it does not necessarily mean it is pleasing. With linear channels, the users can only watch what the current channel is playing. Often, since these channels mostly stream popular music videos, users enjoy them. However, would it be better if the channels could also play personalized music videos for the user? To answer this question, personalized recommender systems should be developed.

There have been many studies on how to develop recommender systems for television. However, most investigations of recommender systems for television perform research for TV program recommendation [43]. One key difference between recommending TV programs and music videos is due to the length of the content. Typical music videos span 3-5 minutes, while TV programs generally span half an hour or more. If we assume that a user wants to spend the next 30 minutes watching television, one good recommendation of a TV program that is selected by the user perhaps is already enough to keep them engaged, while for music videos, we need to provide more items due to the short duration of the videos. Thus, choosing the right items to recommend might be more challenging for music videos than for TV programs.

[1] https://www.fastcompany.com/3054176/why-spotifys-discover-weekly-playlists-are-such-a-hit
[2] http://blog.pandora.com/us/personalized-soundtracks/

The central topic of this thesis revolves around the development of personalized recommender systems for curation-based music videos television. Our use case is taken at XITE Networks International (XITE)[3], a music television company active in The Netherlands, Germany, Belgium, Qatar, Canada, and the United States. XITE has developed an interactive television product called Personalized Music Television (PMT). The PMT is designed to stream music videos continuously and has multiple methods of serving music videos to its viewers. Most relevant for our study is that it allows viewers to select from a variety of curated *channels* such as "Hip-Hop Essentials", "All Out Dance", and "80's Flashback", or to curate their own channel by selecting specific genres, decades, moods, and visual aspects. Once a viewer selects a channel, the order of the streamed videos is controlled algorithmically, rather than by the viewer or by a curator as is typically the case with playlists. At any time the viewer can interact with the application by giving a *like* to indicate a preference, or making *skip* to go to the next track.

As a business, XITE has some key metrics to measure its success and growth. One of them is to increase user engagement with the PMT. In order to do so, XITE has identified and chosen to monitor the session duration. Generally, any form of improvement of the product needs to have a positive impact to increase the session duration, including any recommender systems algorithms developed for use in the product. The overarching goal of this thesis is to contribute on developing techniques to understand users' behavior, find patterns, and propose recommendation models that are aligned with XITE objective to increase session duration and users' engagement.

We begin this thesis by understanding specific challenges related to the domain. Understanding feedback is one of the key aspects of our study, thus special attention is given to how to exploit different types of feedback. We explore two different ways of framing the recommender system task: as a session re-ranker and 'like' prediction task.

There are three types of evaluation protocol which can be applied to compare different models of recommender system algorithms: user studies, offline, and online evaluations [35]. For this thesis, conducting online experiment is not feasible because of limited access to the production system. Thus, it is decided that an offline experiment be performed to compare and evaluate various recommender systems scenario.

1.2 Challenges

Based on our preliminary understanding of the nature of curation-based music video television service and the literature study that has been performed for this thesis, some key challenges are identified as follows. These challenges eventually form as strong motivations toward formulating the research questions and the experimental studies.

1. **Minimal control to choose contents**

   In typical online music services, users have substantial control over what items they consume. Spotify's users can choose which songs they want to listen to, be it in the form of searching for specific songs, browsing albums or discovering new songs via

[3] http://www.xitenetworks.com

playlists. Users can also manually curate songs and create their own personalized playlists. Music radios, on the other hand, work differently. Typically, pre-selected songs are already chosen for different playlists. One of the most popular internet radio services is Pandora. The free service of Pandora does not allow users to choose songs[4]. Instead, the user can give a seed artist, and the system will play songs that would reflect the styles of the seed artist. Such online radio service offers limited types of user feedback. Typically, a user can either like or skip the songs. Nevertheless, these are already important user feedback.

XITE PMT also offers users limited control to choose which music videos to watch. Once a user selects a channel, streams of curated music videos are continuously played while giving freedom to users to express their preferences by giving different types of feedback. At the moment, the user cannot foresee what videos to be played next.

2. **Repeated item consumption behavior**

   The number of music videos in XITE's catalog is far less than the number of songs, since typically, not all songs released by artists in an album will be offered in the music video version. On the other hand, the average length of music videos is similar to songs which typically falls between 3-5 minutes. Moreover, unlike many scenarios in movie or product recommendation in e-commerce, the user might enjoy listening to the same song or watching the same music video again from time to time [40]. This phenomenon is present in our use case because often, when a user opens the same channel multiple times over different time of the day or week, popular music videos are played repeatedly. Naturally, users feedback toward exposure to the same videos might be different.

   Understanding the user behavior toward repeated item consumption is important to predict the likelihood that the user will enjoy the videos that they have already been presented before. For example, a video A which is skipped once by the user does not necessarily mean that the user will never want to watch A again. However, if a video B is skipped five times and only watched completely once, B might be less preferable by the user. Thus, intuitively, if the curation-based systems need to choose between playing A or B to the user, perhaps an option to choose A over B is better. This is merely a simple illustration. Thorough empirical study on the user behavior toward repeated item consumption is described later in this thesis.

3. **Leveraging multiple types of feedback**

   Despite the existence of a large volume of research on music recommender system domain, there are few papers that touch on the concept of exploiting explicit and implicit feedback for the recommender system [18, 24, 44]. The reason is probably due to the lack of public datasets including these types of feedback. Currently, the most popular publicly available dataset that contains information on music playback events is last.fm dataset [5]. This dataset contains information on the user, item, and

[4] at the time of writing this thesis, September 2018

timestamps level. Larger and newer similar dataset called LFM-1b is also available [38]. The 30Music dataset [41], which was released in 2015, offers a similar type of data with additional explicit 'like' information given by users on last.fm platform. Nevertheless, our use case at XITE offers an opportunity to explore multiple types of feedback. The opportunity of course brings additional challenge since we need to design an algorithm to leverage these different types of feedback.

4. **Understanding user attention**

   When watching television at home, it is relatively common for the viewer to turn the TV on while engaging in other activities. The TV is just played in the background as they are performing other task around the house. From the perspective of the service provider, it is difficult, if not impossible, to detect whether the user is paying full attention to the music video played on the television. We think that this understanding is crucial because if we want to incorporate a complete watched event as positive implicit feedback, the level of certainty whether the user is *actually* watching the music video or not will affect the integration of this feedback into the recommender systems algorithms. For example, imagine a situation that the user is watching the television, and then falling asleep without turning off their TV. Then, it is probably not useful to include the music videos played when they are sleeping as positive implicit feedback.

1.3 Research Questions

Based upon the challenges that have been identified in Section 1.2, we formulate four research questions which we will address by conducting thorough analysis on the dataset that XITE provides.

- **RQ1** *Can we successfully rerank sessions in an offline experiment of recommender systems evaluation for curation-based music video television?*

  Related to the first challenge we have described earlier, we begin our exploration by defining the task that we would like to solve. In particular, we focus on the challenge of reranking items in the curated channels, which are streamed to users in sessions of various length and contents. We think that session reranking is a good proxy to emulate how the online recommendation works in practice. Note that in an online setting, music videos which have been selected before by human curators for a specific channel needs to be presented to the user in a particular order. More detailed explanation about session reranking task is described in Section 3.2.

  It is important to highlight that the reranking term used in our study is different with reranking strategy often described in many recommender systems research [45, 28, 42], which focuses on reranking the result of recommendation list given by certain algorithms to produce a more diverse list. Ours focuses on reranking the items in a session, with respect to the original rank of items based on the ground truth. We also

investigate and propose a new metric which reflect the objective of the recommender system at XITE.

- **RQ2** *Can we improve the recommender system performance on sessions containing items that have been presented before to the user by considering the sequence of repeated item consumption behavior?*

  Related to the challenge of repeated item consumption behavior, given past interactions of a user for a particular item, we propose an algorithm which leverages this information. In particular, we are interested in checking whether time-ordered sequence of user feedback toward the same video matters or not to predict the likelihood that the user will enjoy the music video again. We perform analysis on the session reranking task and compare the performance of the model with and without incorporating the repeated item consumption behavior factor.

- **RQ3** *Can we improve the performance of the recommender system by leveraging multiple types of user feedback?*

  Concerning the challenge of leveraging multiple types of feedback, we explore the different possible ways to incorporate the feedback into the recommender system algorithm. Interpreting different user feedback while understanding the context in which that feedback is given is needed to infer how does one feel about the music videos that are presented to them. We are particularly interested in exploring the use of playback percentage and explicit 'like' feedback. We show that considering the use of learning-to-rank based algorithm called Bayesian Personalized Ranking [33], better performance can be achieved by performing multi-level sampling strategy that takes into account multiple types of feedback.

- **RQ4** *Can we improve the performance of the recommender system by considering predicted user attention in a session?*

  Related to the challenge of understanding user attention, we tested out various scenarios in which we predict the user attention when watching music videos in a session. In particular, we are interested in inferring the state of user attention when the user is watching a music video completely, because it is unclear whether the user is *actually* watching, or paying attention, to the video or not.

  It is important to note that we are not going to evaluate the accuracy of the user attention prediction. Our focus is to verify whether predicting user attention when inferring the meaning when a user watches a video completely improves the final performance of our recommender system or not. We validate the performance with and without taking into account the prediction of user attention in the algorithm.

1.4 Contributions

Based on our study, we have identified the following contributions.

1. Our first contribution is to show that we can leverage the user feedback in the form of average playback ratio to perform session reranking task and achieve significant improvement as compared to random baseline. We have also experimented with model-based recommender systems (SVD), and albeit its small absolute incremental difference, we can achieve statistically significant improvement as compared to the popularity baseline.

2. We propose a score to represent the likelihood that the user will enjoy the same video which have been presented to them before, by using empirical probability approach conditioning on the previous sequence of user feedback toward the same item. The probability is calculated empirically based on the available training data. Furthermore, a weighted average is calculated based on the new score with the average playback ratio. Albeit its simplicity, we can achieve statistically significant improvement in our session reranking task by using the weighted average score as compared to only using average playback ratio.

3. While the work related to graded and multi-level sampling strategy in BPR is not new [30, 29], to the best of our knowledge, we are the first to incorporate multiple types of user feedback and use them in BPR framework for music videos streaming domain. On the 'like' prediction task, we show that by incorporating sampling from multiple types of feedback, the multi-sampler BPR performance significantly improved as compared to the baseline BPR. The contribution is on the use of average playback ratio of items to infer positive and negative implicit feedback. Albeit with simpler approach, we have verified the findings in [30] that noticeable improvement is also achieved by incorporating multiple types of feedback in the domain of personalized music video television.

4. To the best of our knowledge, we are the first to perform empirical study to incorporate user attention prediction in continuous uninterrupted complete consumption of items on the recommender system algorithm. We show that by using multi-level BPR framework in 'like' prediction task, an improvement is achieved when only considering few music videos played completely after an active action is taken by the user as positive implicit feedback.

   We use heuristic method to choose the number of videos N that needs to be considered. This method is naive yet it works well in our experiments. However, we acknowledge that in our experiments, the optimal choices of N cannot be generalized over different splits of dataset that we have evaluated. The performance is also not affected when predicting 'like' on items which have not been presented to the users. While our study is quite rudimentary, we hope that it opens up further research directions on the importance of understanding user attention in music recommender systems.

1.5 Thesis Structure

In Chapter 2, we discuss the recommender system in general, including the user feedback, algorithms, and evaluation metrics. We mention some previous work which is related to repeated item recommendation and considering user state in music recommender system. In Chapter 3, we describe our approach to answer the research questions. Chapter 4 describes the details of our implementation, including preliminary data analysis on the offline dataset. The results of our experiments are described thoroughly in Chapter 5. Finally, we describe the conclusions, recommendations, and future work in Chapter 6.

1.6 Publication

Parts of the study in this thesis lead to one accepted contribution as a poster paper at ACM RecSys Workshop on Offline Evaluation for Recommender Systems (REVEAL 2018), which will be held in October 2018 at Vancouver, Canada. The contribution was written together with the company supervisor, Dr. Bouke Huurnink, and the thesis supervisor, Prof. Martha Larson.

Chapter 2

Background

In this chapter, the broad definition of recommender systems along with various specific techniques and algorithms are discussed. First, in Section 2.1, categories of recommender system defined with respect to what information that is utilized are explained. Section 2.2 covers different perspectives on obtaining feedback from the user as an input to train recommender systems model. Section 2.3 discusses the state-of-the-art algorithms in recommender systems. Special attention is given to learning-to-rank based models called Bayesian Personalized Ranking framework, which we are going to use extensively in our approaches later. Section 2.3.4 explains different evaluation metrics that are used to test and compare different recommender systems model. Related work on the exploration of repeated items and user attention for recommender system is described in Section 2.4 and 2.5. Finally, we close off this chapter by describing our use case, XITE Personalized Music Television (PMT).

2.1 Recommender Systems

Depending on which features or interaction information is exploited, recommender system techniques, in general, can be divided into three broad categories.

1. Content-based filtering.

   Systems that use this method exploit what the user has liked in the past and recommend similar items [35]. Each item, or user, can be represented by a sets of features. For example, a song can be characterized by its audio signals, lyric, title, or genres, and a user can have several properties like age, gender, or occupation. Content-based technique leverages these attributes to recommend items by finding similar items with respect to the characteristics of the target users. Some similarity measures are usually used such as cosine similarity, Euclidean, or Manhattan distance, depending on the type of the features.

2. Collaborative filtering.

   In collaborative filtering (CF), user and item interaction data is leveraged to recommend items to the target users. Using previously rated items by the target user and

other like-minded users, we can predict the rating of unobserved items for a target user [37]. More recent developments of CF also work for unary rating, which is typically found in implicit feedback dataset. Furthermore, in many domains, the size of the user and item is quite large, and the user might not interact with lots of items. This presents a particular challenge because the user-item interaction information is very sparse. Moreover, since collaborative filtering exploits users' past interactions, another challenge might arise to recommend items to a new user, because the system is unable to determine the user preference. A similar problem also arises for a new item. This problem is referred to in the literature as the *cold-start* problem.

3. Hybrid recommender.

   Hybrid-based recommenders combine information from various sources of information, both content-based or user-item interaction data. This is especially relevant in domains where substantial information of the user or item features, as well as user-item interactions data, are available. There are various techniques to combine the recommender model. In [4], some techniques are mentioned: weighted, switching, mixed, feature combination, cascade, feature augmentation, and meta-level. For example, in a weighted based hybrid model, we combine the score obtained from different models multiplied by their corresponding weight. The weights then can be tuned depending on the importance of each model.

2.2 User Feedback

There are two types of user feedback which are commonly used in recommender systems: explicit and implicit feedback.

1. Explicit feedback.

   The Netflix prize popularizes the exploitation of explicit feedback data in recommender system research. This type of feedback is typically in the form of ratings that the user explicitly gives to items [25]. Numerical ratings within a certain range are typically used, such as discrete rating between 1 until 5. Another range of number is also possible. While explicit feedback gives more confidence to infer a user preference towards an item compared to implicit feedback, it is commonly more difficult to obtain since it adds an additional step to the user to rate the item. The service provider needs to make it easy for the user to explicitly rate an item, otherwise the user might be reluctant to rate the items.

2. Implicit feedback.

   Since explicit feedback is typically found to be quite sparse, there is growing attention to exploiting user behavior when using the system and infer user preferences from the behavior [16]. User behavior such as browse, click, view, or listen, can be indirectly mapped into positive user preference towards particular items. These forms of user behavior are called implicit feedback and often are represented by using a binary value of 0 and 1. Some forms of implicit feedback are represented as arbitrary

number, such as listening count to a song. It is expected that the amount of implicit feedback would be bigger than explicit feedback. Nevertheless, special attention must be given, as with implicit feedback, we have less confidence on the user preference compared to explicit feedback.

In the music domain, both types of feedback described are present with different representation in various applications. For example, listening to a song is categorized as positive implicit feedback [39]. Other forms of feedback are available, such as giving a 'like' or thumb-up, which is considered as explicit feedback. The explicit and implicit feedback can be considered together in the recommender system [18]. The user can also add songs to a playlist. While listening to a song, a user can click skip in the middle of it, perhaps indicating an implicit dislike, or the user just does not want to listen to this song under the current context, but prefer to listen to it in another context.

There are several public datasets which have helped to move the research on music recommender system forward. The Last.fm dataset [5] is probably the most popular ones among those used in the literature. It contains information on user listening events of almost 1000 users across millions of songs. The dataset contains implicit feedback data in the form of timestamped events for each listening session. Because the user might listen to songs multiple times, the play count of the songs for every user can be used as a confidence measure of user preferences toward a particular song. Intuitively, if the user listens to a song repeatedly, the user might like it better than those songs that are only listened once or twice.

What is generally missing from publicly available datasets is information on the playback percentage, although some research has derived a way to calculate the duration based on the difference of two consecutive songs to measure the skip ratio [44]. Moreover, songs that are skipped are not included in the dataset of [5]. The skipped song might actually carry a strong signal that will help the model to learn user preferences better.

2.3 Algorithms

In this thesis, we are particularly interested in using collaborative filtering framework, because compared to content-based methods, it is typically considered to provide better and more diverse recommendations. Since we are going to use collaborative filtering model in our research, more focus is given to explain the background and related work in this field.

2.3.1 Neighborhood-based method

The idea of the neighborhood-based model is to exploit the similarity of other users and items in the UI-matrix to recommend relevant items to the target user or vice versa. There are two types of neighborhood-based method: user-based and item-based collaborative filtering.

1. User-based neighborhood [34]

The idea of the user-based neighborhoods is to find users who give similar ratings as the target user to sets of items to predict the rating given by the target user on the target item. Note that the target item needs to be rated by the users in the target user's neighborhood. The final predicted rating is the weighted contribution among the users based on the similarity of their ratings to the target user.

2. Item-based neighborhood [37]

   In the item-based neighborhood, the idea is to find similarity across items first and based on this information, we can predict the rating given by a target user by exploiting ratings that she has rated on similar items with respect to the target item.

2.3.2 Latent factor models

One of the drawbacks of neighborhood-based methods is that they only operate on a user or item basis. Interactions between user and item might be more complex than that. Latent factor models can exploit the user and item factors together to create a prediction. The state-of-the-art latent-factor model is based on matrix factorization (MF).

The basic idea of matrix factorization is similar to dimensionality reduction, which tries to reconstruct a full-size matrix with a low-rank matrix representation. Earlier work explores such dimensionality reduction technique in recommender system [36]. Given a rating matrix r for i number of users and j number of items, MF tries to find a low-rank matrix representation for both user and item factor, typically with the factor f far smaller than i or j, so that the dot product of both of them, $\hat{r}_{ui} = q_i^T p_u$, will approximate the full matrix r. Singular Value Decomposition (SVD) is a dimensionality reduction technique that can be used to find these low-rank representations [25].

Let χ defined as all pairs of user and item, the vector q_i and p^u can be learned by minimizing the following objective function over all sets of known ratings in χ. Furthermore, regularization factor is added to avoid overfitting of the model.

$$\min_{q,p} \sum_{(u,i)\in\chi} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{2.1}$$

A variant of SVD called FunkSVD is popularized by Simon Funk on the Netflix prize competition in 2009. He uses a stochastic gradient descent (SGD) method to solve Eq. 2.1.

Another optimization method is Alternating Least Square (ALS). ALS is used in [16] to solve the problem of deriving latent factor models for implicit feedback data. ALS is preferred over SGD because it can be easily parallelized. This is beneficial in implicit feedback data because typically we have lots of data compared to explicit feedback data. This is also the reason why ALS is built-in in Apache Spark, highly scalable and parallelized software to run computation with a huge dataset.

2.3.3 Learning-to-rank based model

In rating prediction task, the evaluation is performed by minimizing the difference between predicted and ground truth rating, typically measured by using Root Mean Square Error

(RMSE). However, in the actual implementation of recommender systems online, the users often are presented with list of items, not the predicted ratings of the item. For example, when we click an item in Amazon website, at the bottom of the page, we are presented with some items that we might be interested with.

In recommender systems research, the term 'Top-N recommendations' is used to refer to the task where the algorithms generate a list of items that are predicted to be most relevant to the user [8]. In this task, it is found empirically that the algorithms which are optimized for RMSE might not work well in terms of ranking-based accuracy metrics, such as precision and recall [8]. In recent years, many algorithms have been proposed to optimize directly for ranking-based metrics.

Learning-to-rank (LTR) based models learn the relative ordering of the relevant and irrelevant items. There are three types of such models: pointwise, pairwise, and listwise method. For this thesis, we specifically focus on the pairwise method. We chose one family of pairwise methods based on Bayesian Personalized Ranking framework, which we describe extensively as follows.

Bayesian Personalized Ranking (BPR)

BPR [33] falls into the category of pairwise comparison approach and has emerged as the state-of-the-art for Top-N recommendation task. It is often a strong baseline that is difficult to beat.

For implicit feedback data, we only have positive feedback values, which indicate a preference towards an item. The implicit rating consists of 1 for observed item and 0 otherwise. BPR tries to learn the relative ranking by using the assumption that the user prefers items that have been observed compared to the unobserved ones. This is an assumption because we do not know precisely whether the unobserved items are preferred or not by the user. It can be the case that the user has not been exposed yet to the items, or the user is indeed not interested with the item (and thus decided to not interact with it).

To train a BPR-based recommender model, three crucial concepts need to be understood: the sampling strategy, the BPR optimization criterion, and the model.

1. Sampling strategy.

   To learn the relative ranking of items for each user, BPR uses triplets of (*user, observed item, unobserved item*) to learn the relative preference. Formally, the triplets are sampled according to Eq. 2.2.

$$D_S := (u,i,j)|i \in I_u^+ \wedge j \in I \setminus I_u^+ \tag{2.2}$$

   Let I and I_u^+ defined as all items and the items which a user u gives positive feedback value respectively. D_S contains all triplets (u,i,j) such that a user u prefers item i over item j. Moreover, the positive item i is taken from I_u^+ and the negative item j is sampled from random unobserved items.

2. BPR optimization criterion

The original BPR implementation derives an optimization criterion called BPR-OPT, illustrated as follows. Note that BPR is an optimization framework, and we have to use a separate model to produce the recommendation for users. Let Θ defined as the parameter of a model optimized by BPR. BPR-OPT optimizes the posterior probability $p(\Theta| >_u)$, where $>_u$ indicates the latent structure for user u. A logistic sigmoid function is used to calculate the probability that user u prefers item i over j. The input to the function is defined as $\hat{x}_{uij}$. $\hat{x}_{uij}$ is calculated based on the model that we use i.e. matrix factorization.

$$\begin{aligned}
\text{BPR-OPT} &:= \ln p(\Theta| >_u) \\
&= \ln p(>_u |\Theta) p(\Theta) \\
&= \ln \prod_{(u,i,j)\in D_S} \sigma(\hat{x}_{uij}) p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2
\end{aligned} \tag{2.3}$$

To optimize BPR-OPT, stochastic gradient descent technique is used, and the overall learning procedure called LearnBPR, is illustrated as follows [33].

Algorithm 1 LearnBPR procedure

procedure LEARNBPR(D_S, Θ)
initialize Θ
  repeat
    draw (u,i,j) from D_S
    $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial \hat{x}_{uij}}{\partial x} + \lambda_\Theta \cdot \Theta \right)$
  until convergence
return $\hat{\Theta}$
end procedure

3. Model.

Note that in Algorithm 1, the term $\hat{x}_{uij}$ is used to represent a value that explains the relationship between user u, item i, and j. The choice of $\hat{x}_{uij}$ is dependent on the model used. Typically, BPR based on Matrix Factorization is used. In MF, $\hat{x}_{uij}$ is decomposed into :

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj} \tag{2.4}$$

In MF, a dot product of two low-rank matrix is used to calculate the predicted rating. Hence, for user-factor matrix U and item-factor matrix I containing n factors, we can calculate $\hat{x}_{ui}$ as follows:

$$\hat{x}_{ui} = \langle U_u, I_i \rangle = \sum_{f=1}^{n} U_{uf} \cdot I_{if} \tag{2.5}$$

Having defined how to calculate $\hat{x}_{ui}$, based on Algorithm 1, we need to calculate the partial derivative with respect to the parameter of the model Θ as follows :

$$\frac{\partial}{\partial x}\hat{x}_{uij} = \begin{cases} I_{if} - I_{jf} & \text{if } \Theta = U_{uf}, \\ U_{uf}, & \text{if } \Theta = I_{if}, \\ U_{uf}, & \text{if } \Theta = I_{jf}, \\ 0 & otherwise \end{cases} \tag{2.6}$$

2.3.4 Evaluation metrics for offline evaluation

Since our study performs offline evaluation, we highlight several evaluation metrics which are commonly used. In general, there are two types of evaluation metrics: ones that focus on rating prediction task, thus measuring the difference between predicted and ground truth rating, and ones that focus on evaluating the quality of the ranked list of items, also with respect to a ground truth. These evaluation metrics are explained as follows.

1. Rating prediction accuracy metrics

   There are two metrics which fall into this category and often used in the literature: Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). Both of them measure the difference between predicted and ground truth rating but in a different way. Because the difference is raised to the power of two, RMSE penalizes large error more than MAE.

   In both Eq. 2.7 and 2.8, r_{ui} and $\hat{r}_{ui}$ represent the ground truth and predicted rating for user u to item i, while N represents the total number of available ratings that are evaluated.

$$MAE = \frac{1}{N}\sum_{u,i}|\hat{r}_{ui} - r_{ui}| \tag{2.7}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{u,i}(\hat{r}_{ui} - r_{ui})^2} \tag{2.8}$$

2. Top-N recommendation accuracy metrics

   For Top-N recommendation task, rather than predicting the score correctly, it is the correct order of the relevant items that matters. The evaluation metrics to measure

rank order are primarily borrowed from the information retrieval domain, and they are dependent on the relevance value of each item. Typically, the relevance value can be represented either with binary or continuous value. Evaluation metrics that are suitable for binary relevances are: precision, recall, F1-score, and Mean Average Precision (MAP). For graded relevance value, normalized Discounted Cumulative Gain (nDCG) is often used.

Mean Average Precision (MAP). The Average Precision (AP) of a list is calculated by taking the Precision at K for every relevant items in the list. Considering m relevant items for a list with n elements, the AP can be defined as :

$$AP = \frac{1}{m} \sum_{k=1}^{n} P(k) \cdot rel(k) \tag{2.9}$$

Here, $P(k)$ represents the precision at position k. $rel(k)$ is 1 if the item at k is relevant and 0 otherwise. The MAP is simply calculated by taking the mean of all AP across all available queries, or lists N, in the test set:

$$MAP = \frac{1}{N} \sum_{q=1}^{N} AP(q) \tag{2.10}$$

Normalized Discounted Cumulative Gain (NDCG). For a list containing items with graded relevance, typically represented by real or continuous numbers, items with high relevance is considered to be more valuable and the larger the ranking of the item in the list, it is regarded as being less valuable to the user [17]. nDCG captures those two important characteristics. To calculate nDCG, first, Cumulative Gain (CG) needs to be determined [7]

$$CG[k] = \sum_{j=1}^{k} G[j] \tag{2.11}$$

CG is measured by summing all relevance value G for every item in the list of length k. In order to penalize item located lower in the list, the Discounted Cumulative Gain (DCG) is calculated as follows [7] :

$$DCG[k] = \sum_{j=1}^{k} \frac{G[j]}{log_2(1+j)} \tag{2.12}$$

Finally, to calculate nDCG, we divide the $DCG[k]$ by the ideal DCG $DCG'[k]$, which is measured from the perfectly ranked list based on relevance ground truth [7].

$$nDCG = \frac{DCG[k]}{DCG'[k]} \tag{2.13}$$

2.4 Repeated Item Recommendation

The utility of recommending the same items that have been consumed before by the user are different depending on the domain. For bookstore websites, it might not be preferable because typically people buy one copy per one book title. However, for the music domain which is relevant to our study, it might be desirable since people enjoy listening to the same songs at different times and in different contexts [40].

To explore ways to answer RQ2 about repeated item consumption behavior, literature study is performed to find an empirical way to analyze the behavior of users when consuming repeated items. The dynamics of repeated item consumption is studied in [2] across different domains. Based on the study, there are three aspects of an item that the authors argue might affect the decision for a user to choose which items from the past that they would like to consume next: popularity, satiation, and recency.

1. **Popularity** measures the underlying quality of the item, as indicated by how often the user consumes them in the past. Items that are consumed more are considered to be more popular and of better quality. It is found empirically that across different datasets, people tend to consume items with better quality again, rather than those with lower quality. For example, rather than visiting the same restaurant that has been visited once during the past month, people tend to visit the same restaurant which is visited five times in the past month.

2. **Satiation** relates to the level of boredom. To check for the presence of boredom in [2], the behavior of people when consuming the same items in a row is analyzed. If satiation occurs, the authors argue that the probability of consuming an item again after repeatedly consuming it for a certain number of k times will decrease at some point as k increases.

3. **Recency**. Generally, the more recently an item has been consumed, the higher the probability that it will be reconsumed compared to the probability of those items which have been consumed a while ago.

Empirically checking whether such phenomena exist in our dataset is essential to get an initial outlook as to whether there are certain patterns of repeated item consumption. We use this framework with a small modification that suits our dataset in order to measure the dynamics of repeated item consumption empirically. The modification is related to the different characteristics between the domains studied in [2] with ours. In particular, note that exposure to items in the PMT is regulated by an internal system at XITE, and while the users in general have very little control on what items they can see, they are free to give *multiple* types of feedback on the items. In [2], the domains which are analyzed only contains *one* type of positive feedback, and the items are assumed to be independently chosen by the users.

Furthermore, recent research also highlights the importance of modeling repeated item recommendation [22, 10, 6]. Of particular importance is to answer the question of when and

what items that need to be recommended again. Semi Hidden Markov Model [22] and low-rank Hawkes Process [10] have been used to answer the question of when and what repeated items should be recommended. While their study found that such techniques are useful in recommending the right recurrent items recommendation at the right time, the applicability to our use case might be limited due to the characteristics of our domain. Although many items are repeatedly consumed in the PMT application, it is important to note that these repeated consumptions are not due to the intention of the user. Instead, it is because how the curation-based system works which sometimes stream popular items multiple times over different period.

2.5 User Mode Detection

People listen to music under different conditions. Sometimes, the music is played in the background while they are performing other activities such as studying, driving, or exercising. Under such situations, user interactions with the music application are limited. On other occasions, the people actively interact with the music app to choose which songs that they want to listen. Such interactions might be in the form of searching for a song, selecting a song from a playlist, or skipping a song occasionally.

Due to the scarcity of datasets with rich information in terms of feedback types, there is little peer-reviewed research on how to understand the context when users are enjoying music, in terms of active or passive exploration. [24] performs a study on understanding different behavior of people when listening to music and characterize different sessions depending on the activity level of a user when using the app. An active session is defined as a session that contains a lot of active user interactions, while a passive session is defined as a a session with sparse user interactions [24]. User interactions consist of choosing a song, skipping a song, or giving a rating to items.

Making a distinction between these sessions is important because depending on whether an action is taken on an active or passive session, a skip might reflect a different level of preference. A skip performed during a passive session might imply that the user probably dislikes the items more strongly than if the skip is performed during an active session [24]. In an active session, the user tries to explore the item and find songs that they would like to listen to. It is common that during such session, people skip songs occasionally.

While [24] pointed to some directions on inferring implicit feedback on explicit feedback data, extensive evaluation of recommender systems algorithms taking into account those types of feedback has not been explored. This opens up a research question similar to our RQ4 which tries to explore the effect of inferring user mode on the performance of our recommender system.

2.6 Case study: XITE Personalized Music Television (PMT)

The PMT consists primarily of various channels covering different styles of music videos. Other than that, it also offers viewers the possibility to search for music videos from the catalog, generate custom playlist from pre-defined filters, and also interact with previously

'liked' music videos. These channels are created based on seasonal themes and specific popular genres such as rock, R&B, and hip-hop. The most popular channel is the 'Now' channel which provides the latest and trending music videos. Apart from curated channels, there is a 'Mixer' channel where a user can select from 4 pre-defined filters to create their own channel. While watching the music videos, the user can also press 'like' and 'skip' as a feedback to the system. This feedback is valuable for XITE because based on the user feedback, an internal algorithm can be developed to provide personalized recommendation for each user. Fig. 2.1 shows the user interface of the channel selection with the 'Now' channel being the currently selected channel.

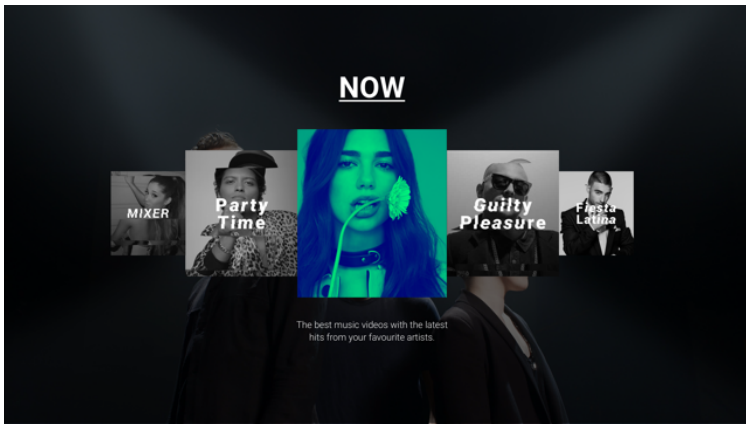


Figure 2.1: A screenshot of PMT channels

Figure 2.2: A screenshot when the user presses like

To curate music videos in each of those channels, XITE has teams of music video experts. The team analyzes, among others, different behavioral metrics and popular charts to choose the best music videos to present to the user within a certain period. The music team is also responsible for annotating and classifying music videos into different taxonomies, such as mood and genre, which will potentially turn into valuable features later on. As the

user base is steadily growing, and people start engaging frequently with the service, more mature and advanced recommender systems thus become feasible to be deployed to make better recommendations to users. Combining rich users interaction data with content-based features would also be an interesting direction onward for XITE.

Unlike typical music streaming applications, there are only a few actions that user can perform on the PMT. When the music video is playing, a user can press 'like' to express positive preference and skip to move into the next video. The user cannot perform fast-forward or pause on the music video while it is playing. The 'liked' music videos are saved in the user's Favorites channel. In this setting, 'like' can be categorized as explicit feedbacks. When the user watches a music video completely, to some extent, we can also implicitly infer that the user like it, albeit with less likeness power compared to pressing 'like' button. This way, a complete *play* can also be categorized as an implicit positive feedback to the system.

Chapter 3

Approach

This chapter discusses the approaches used to answer the research questions identified in Chapter 1 of this thesis. First, we describe considerations on choosing the right evaluation measures as a proxy that is aligned with the business objectives of XITE. Then, two tasks are described: session reranking and 'like' prediction task. The session reranking task explores how RQ1 and RQ2 can be answered, while the 'like' prediction task is more focused towards answering RQ3 and RQ4.

3.1 Evaluation Metrics

The proposed recommender systems in this thesis needs to be aligned with the business objectives of XITE, which technically are represented via certain online evaluation metrics. After rounds of discussion with the representatives from the company, it emerged that one of the most important business objectives is to increase user engagement and satisfaction. The user engagement is translated into increased average viewing time per user, which eventually leads to increased overall session duration. YouTube, for example, also reports in their study that session length is among one of the metrics that they use to evaluate the performance of their online recommender systems via A/B testing [9].

A successful recommender system is determined by its performance in an online setting. However, performing online study is not possible in the context of this thesis. Instead, an offline evaluation study is extensively performed. Considering this reason, we need to find offline evaluation metrics which might be a good proxy for the online performance.

Note that our offline data consists of sessions containing time-ordered music videos presented to the user along with the associated user feedback. The user feedback, e.g. a skip or a complete consumption, is mapped into relevance judgment. Practically, we want the recommender system to be able to put on top of the list the music videos that are liked, or consumed completely, by the user while putting the skipped music videos at the bottom. Intuitively, to connect with the original business objective to optimize for user engagement, we have an assumption that if the users are presented with the videos that they will likely consume completely at the beginning of the channel, it will keep them engaged with that channel and spend more times on it, compared to those users who are presented

with irrelevant music videos early in the channel.

This framing of the problem basically turns our problem into how to find a good model or algorithm to re-rank the order of music videos in a session, so that an optimized ranked list is achieved with respect to the relevance judgment. Evaluation metrics which are suitable for evaluating a ranked list are MAP and nDCG, which have been described in Section 2.3.4.

Furthermore, we highlight one assumption that we deem essential so that we can move forward with the offline evaluation method. First, we assume that within the same session, user action toward each music video stays the same regardless of the order of the music videos presented to them. In another words, the actions in a session is independent of each other. This assumption enables us to use ranking-based evaluation measures and frame a session reranking task.

New metric: Weighted Streak Recall (WSR)

For XITE, it is also important that the user enjoys a lean-back experience while using the PMT, meaning that when the user selects a channel, great contents are just presented to them directly, encouraging the user just to enjoy watching the video *continuously* without needing to press skip frequently at the beginning of the channel. Based on internal discussions at XITE, we have identified two key requirements that XITE believes likely underlie relevance from a user perspective.

1. After opening a channel, the user should be presented with the best, or most relevant, personalized content *as soon as possible*. By giving users the items that they like the most first, they will be drawn into the viewing experience, and thus be more likely to stay with and to explore the service.

2. When enjoying streams of music videos on the PMT, it is normal for a user to regularly switch between *lean-forward* mode of active interaction and a *lean-back* mode with passive consumption of music. In this lean-back mode, it could be helpful to maximize streaks of consecutive relevant music videos, minimizing the interruptions to the experience that occurs when a user feels the need to press skip during a session. In other words, videos should be presented so that relevant and non-relevant items are clustered together, rather than interleaved with one another.

Considering the requirements, we propose an evaluation metric called Weighted Streak Recall (WSR). In general, WSR identifies continuous streaks of relevant items among the stream and calculates the recall score by dividing the length of the streak with the total number of relevant items in the stream. Then, to take into account the position of the streak, each streak is weighted by its inverse position in the stream. By this definition, we appreciate longer streak appearing at the beginning of the streak rather than those appearing later.

The pseudo-code to calculate WSR is defined as in Algorithm 2. Note that for generalizability, we use an input L of the ground truth duration of each music videos along with a threshold t seconds to binarize the relevance into 0 or 1.

Algorithm 2 Weighted Streak Recall

procedure WEIGHTEDSTREAKRECALL(L, t)
Form a modified list L, composed of binary relevance with a threshold of t seconds
Calculate length of the list len_l
Calculate number of relevant items n_rel from L
Initialize $group$,$group\_temp$ as empty lists
Initialize $find = False$, a boolean variable
Initialize weighted streak recall WSR to zero
Calculate list l_rel which contains the position of each relevant items in L
    for $i = 0; i < len\_l; i++$ **do**
        if $i \in l\_rel$ **then**
            $find = True$
            Add i to $group\_temp$
            if $i == (len\_l - 1)$ **then**
                if $length(group\_temp) \geq 2$ **then**
                    Add $group\_temp$ to $group$
                end if
            end if
        else
            $find = False$
            if $length(group\_temp) \geq 2$ **then**
                Add $group\_temp$ to $group$
            end if
            Assign $group\_temp$ as empty list
        end if
    end for
    for $l \in group$ **do**
        $WSR = WSR + \frac{1}{(l[0]+1)} \frac{length(l)}{n\_rel}$
    end for
return WSR
end procedure

While MAP and nDCG are the go-to metrics used to compare ranked lists produced by different models in information retrieval and recommender system domain, we think that WSR reflects the requirements that XITE thinks are likely relevant from the user perspective, which have not been captured yet by existing metrics like MAP or nDCG. However, because we perform offline evaluation, we cannot verify the usefulness of both WSR or MAP in explaining online performance. It is therefore important to evaluate both measures in online recommender systems to find which one correlates better with the online performance, which is out of scope for this thesis.

3.2 Session Reranking Task

This thesis proposes recommender system algorithms and evaluation for XITE from two different aspects of the use case. The first task is to develop a recommender system which works as a re-ranker on a per-session basis. The second one is to develop a model which is trained on explicit feedback signal, e.g., 'like', with the objective to predict the most likely music videos to be liked by the user in the future. For both of these tasks, a different evaluation protocol and algorithms are used to evaluate the proposed recommender system model. In this section, we describe the formulation of the first task: session reranking, starting by describing the definition of a session.

3.2.1 Session definition

While the dataset contains information on session ID, the system assigns the same session ID as long as the device is actively connected to the system's backend within a certain period. It means that the same session ID is assigned when a user turns on the PMT up until the PMT is disconnected, either when the user leaves the application or there is a problem in the connection between the device and the system's backend.

We propose a new definition of a session which is used for the rest of this thesis. The new definition is proposed merely as a way to distinguish the session term used in this thesis, with the original session field in the offline dataset.

Session ID	Order	Channel category	Video ID	Action	New Session ID
567151	1	Now	111	skip	1
567151	2	Now	222	play	1
567151	3	Now	333	skip	1
567151	**4**	**Now**	**444**	**leavechannel**	**1**
567151	5	Throwback Thursday	555	play	2
567151	6	Throwback Thursday	666	skip	2
567151	7	Throwback Thursday	777	skip	2
567151	8	Throwback Thursday	888	skip	2

Session ID	Order	Channel category	Video ID	Action	New Session ID
567151	1	Now	111	skip	1
567151	2	Now	222	play	1
567151	3	Now	333	skip	1
567151	**4**	**Now**	**444**	**skip**	**1**
567151	5	Throwback Thursday	555	play	2
567151	6	Throwback Thursday	666	skip	2
567151	7	Throwback Thursday	777	skip	2
567151	8	Throwback Thursday	888	skip	2

Figure 3.1: Illustration on the session assignment from the original dataset

A session is defined as the collection of music videos and the associated actions from the users between the start of the channel until the user explicitly leaves the channel, or the subsequent music video is started in a different channel than the existing one. This definition breaks up the original session ID assigned by the system into multiple smaller sessions. Fig.

3.1 illustrates how the new session ID is determined for the original sequences of the music videos.

Furthermore, to formalize, let $s_{u,i}$ designates session i for a user u. A session is a list of pairs consisting of a music video v and a user action a that are ordered by time. A session of length N is defined as :

$$s_{u,i} = [(v_1, a_1), (v_2, a_2), ..., (v_N, a_N)] \tag{3.1}$$

Originally, the user action is represented as one of the following actions : 'play', 'like', 'skip', and 'leavechannel'. A method is used to map the user actions into numerical values and form a list of relevance judgments for music videos within the same session.

For each video played in a session, we assign a binary relevance value to it. The binary relevance informs that a value of 1 is assigned to actions of which reflect strong evidence of user preference, and a value of 0 is assigned for those which do not reflect evidence of user preference. All completely watched and liked videos are assigned a value of 1. A value of 0 is assigned when the user presses 'skip' within less than 30 seconds while watching the video, and 1 otherwise.

Note that for the rest of this thesis, for clarity, we refer to an event of playing a video for more than 30 seconds as *consumption* event e.g. the user *consumes* the item. We assign a value of 1 for videos that are liked and played completely. Hence, by this definition, we have $a_i \in (0, 1)$.

3.2.2 From User Feedback to Feature

One of the objective of this thesis is to exploit multiple types of user feedback for music videos recommender systems. Different types of feedback are mapped to different value of playback ratio, e.g. when a user consumes a video completely, naturally the playback ratio is 1. On the other hand, if a user 'skip' a video, the playback ratio is less than 1. We propose to use the average playback ratio as a score to represent the user preference towards an item.

- For a particular user u and music video v, let N equals to the number of times that v is presented to u. We define a time-ordered sequence $D_{ui} = (d_1, d_2, ..., d_N)$ where d_k represents the playback ratio given by u at k-th times of being presented the video v.

- A score for a user u to an item v is defined with the following formula. The score ranges naturally from 0 until 1.

$$R_{u,v} = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{3.2}$$

This score implicitly indicates the user preference toward a particular music video. For example, when the PMT presents a video N times to a user, a score of 1 means that the user always consumes the video completely. On the other hand, a score close to 0 means that the user almost always skips the music video. While typical dataset containing explicit feedback, such as in Movielens [15] and Netflix dataset, deals with an ordinal Likert-scale

rating from 1-5, the nature of our score is continuous from 0 until 1. This decision to have real value as the score is also used in Jester dataset [14], where the rating data spans continuously from -10 to 10.

3.2.3 Base Algorithms

Considering that our study is the first attempt to perform offline evaluation of recommender systems at XITE, we use relatively straightforward base algorithms which are commonly used in many domains: popularity and matrix factorization with SVD. The base algorithms are described as follows.

1. **Random**. A random recommender is typically used as a sanity check for a recommender system evaluation. The premise is that the proposed algorithm must perform better than random. In practice, given an unordered list of music videos in a session, the random recommender returns a randomly ordered list of music videos.

2. **Popularity (Pop)**. This is a non-personalized algorithm and has been found to be a difficult baseline to beat [8]. Since we are focused on session reranking for curated channels, the user cannot choose which music videos they want to watch on these types of channels. The definition of popularity that we are going to use is slightly different than in the literature. In much of the literature, the popularity model uses the count of how many times the item is observed, or rated, by the user. Our definition of popularity might be closer to what is called the Item Average in [8].

   In our popularity recommender, for each music video, the global popularity rating is determined by taking the average of playback ratio given by all users to that particular video. During inference, this recommender returns an ordered list of music videos in a session in descending order of the global popularity rating.

3. **Matrix Factorization with Singular Value Decomposition (SVD)**. From the derived User-Item matrix, which is based on mapping the average playback ratio to rating as described in Section 3.2.2, a model is trained to predict the rating of the unobserved items. More specifically, the implementation of FunkSVD, which works very well in Netflix prize competition, is used.

Note that due to the repeated item consumption, some music videos in the test sets are already seen by the user during the training period. Due to this reason, a modified version of the algorithm is introduced to take into account directly the score given to items in the test sets by assigning the score obtained from the training sets. For example, by using SVD, the scores for unseen items are assigned based on the model's prediction. However, for already seen items in the training sets, it is more logical to use the training set's score rather than the model's predicted score. Hence, for Popularity and SVD, other variants of the model are added: **Pop_User** and **SVD_User**, to indicate that the user's predicted ratings for seen items on these models are replaced by the rating given by the associated user on the item in the training sets.

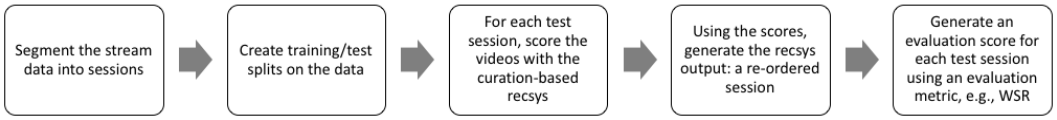


Figure 3.2: Offline evaluation strategy

3.2.4 Offline Evaluation Framework

Having defined the features and the algorithms, the full pipeline of the experiment can now be described in Fig. 3.2.

1. In the dataset, the user's watching history is aggregated on a per-session basis. Each session is represented as described in Equation 3.1

2. The dataset is split into training and test set.

3. The curation-based recommender system is trained on the training set. This model is then applied to each video in the test set. It produces a score that is specific to the user and the video.

4. Based on the predicted score of music videos, a new predicted session is produced. High-scoring videos occur at the beginning of the session. Effectively, session re-ordering is a re-ranking process.

5. Using the ground truth of binary relevance values from each music video in the test set, calculate the evaluation score for each session. The scores can then be averaged for all sessions in the test set.

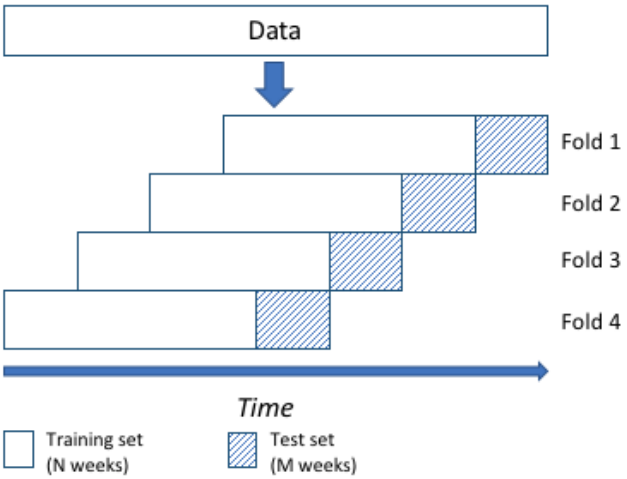


Figure 3.3: Time-based split of the dataset into folds of training and test set

Note that to test the generalizability of the results, a time-based split validation strategy is used [27]. Four folds are created from the whole dataset, as depicted in Figure 3.3. First, for each fold, the dataset is split into eight weeks of training set, followed by one week of test set. This method allows us to simulate the actual potential use-case of the implementation where a model is trained on certain periods of data from the past, and to be used to recommend items in the future.

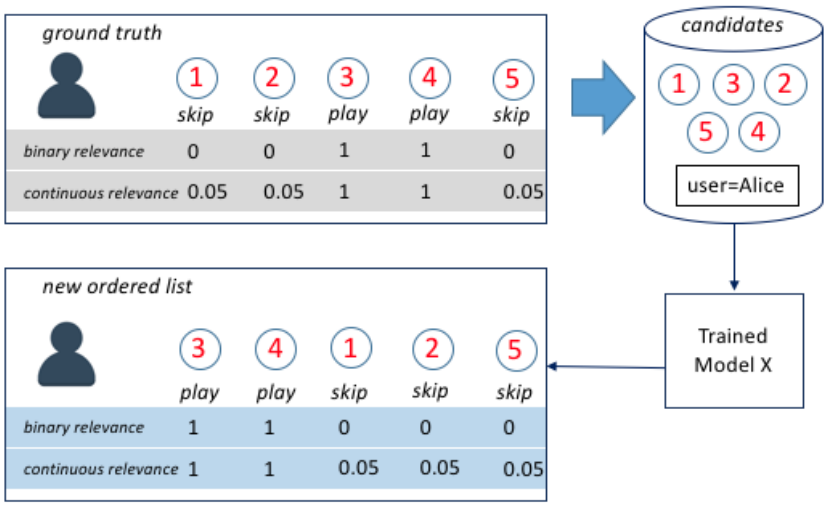


Figure 3.4: Illustration on music videos candidates selection and evaluation

Fig. 3.4 illustrates an example of how the evaluation is performed offline for one session. Suppose that Alice watched five music videos in a session. These music videos have all the same full duration of 200 seconds. While watching these videos, the following actions are performed sequentially by her: skip the first two music videos after 10 seconds, play the third and fourth music videos, and skip the last music videos after 10 seconds. Hence, Alice actions can be represented as sequence of [0, 0, 1, 1, 0], or sequence of [0.05, 0.05, 1, 1, 0.05]. Both sequences define the ground truth of the session.

Now, we have five music video candidates for Alice that needs to be ranked. Our hypothetical trained model X will rank these five items in descending order based on their predicted relevance to Alice. For example, after feeding the candidates to our model, the model produces a predicted order of [3, 4, 1, 2, 5]. Referring to the original ground truth, the predicted order has a relevance value of [1, 1, 0, 0, 0] and [1, 1, 0.05, 0.05, 0.05]. For this particular example, all the MAP, and WSR is equal to 1, and it is the best possible re-ordering of the items.

Note that the evaluation is performed with respect to a known set of item candidates to be scored for each user, and it is done on a per session basis. We contrast our approach with one of the most popular protocols for offline evaluation of recommender systems: one-plus-random [8]. In this approach, for each user in the test set, one rating (or item) is selected as the held-out item. Then, randomly unrated items (1000 are used in [8]) are also picked, and the items together with the held-out item are ranked based on the scores

produced by the algorithm. Based on this order, various types of ranking metrics can be measured. In contrast, in our use-case, we argue that randomly picking from the universe of unobserved items is not fair, since there is a known control on what item is presented: the items presented are drawn from the curated item set for the channel. A more realistic assumption is to assume that the items are drawn from items in the session. In order to implement this assumption, we calculate our evaluation score at the session-level, and not at the item level.

3.3 Repeated Items Consumption Behaviour

Inspired by the approach in [2], repeated item consumption behavior for the PMT is analyzed by using an empirical probability approach that we can directly perform on the data. For the rest of this thesis, given a user's past historical feedback to an item, **the empirical probability of repeated item consumption** is defined as the probability of a user to consume the item again if presented by the system. Note that this term are used a lot in later part of this thesis, and sometimes it is referred only with the term 'empirical probability'.

To give more specific illustration in our use case, we give an example. Imagine that during the past one month, Alice has been presented a music video five times. The sequential order of her consumption can be represented as a sequence of binary value of [0, 0, 1, 1, 1], meaning that Alice skipped the first two occurrences of the video, and consumed the rests of the video for more than 30 seconds (refer to Section 3.2.1 for the definition of consumption and skip). By grouping all actions on unique user and item pairs, we can derive this sequences from the whole dataset. In the next section, we describe the logic behind generating the empirical probability among all the available sequences in our training data.

3.3.1 Deriving the Empirical Probability of Repeat Consumption

To formalize, let $S_{ui} = (s_1, s_2, ..., s_n)$ indicate a time-ordered binary relevance value for item i by a user u with a length n. The empirical probability of repeated item consumption is calculated with the following equation:

$$\Pr(\text{next} = 1|S) = \frac{N(Seq = [S,1])}{N(Seq = [S,0]) + N(Seq = [S,1])} \tag{3.3}$$

Note that the S variable in Eq. 3.3 represents combinatorial patterns of binary value with various lengths. With a length of 2, S can be represented as one of the sequences of [0,0], [0,1], [1,0], or [1,1]. For example, if S equals to [0,0], Eq. 3.3 can be explained as finding the empirical probability that the next action after [0,0] is 1. The empirical probability is calculated by dividing the number of occurrences of the sequence [0,0,1] from the data with a total number of sequences with length 3, which is due to the nature of the binary feedback, can be calculated by the sum of the number of sequences of [0,0,0] and [0,0,1].

The formulation of the empirical probability is related to the concept of n-grams in natural language processing. In n-grams, given a corpus of text, we can generate the probability of occurance of a string w_n with length n with the following equation [3] :

$$\Pr(w_n|w_1^{n-1}) = \frac{C(w_1^{n-1}w_n)}{\sum_w C(w_1^{n-1}w)} \tag{3.4}$$

Note that the probability is calculated by measuring all the occurrences of words after $n-1$ sequences of string w_1^{n-1}. $C(w)$ simply denotes the number of times that string w occurs in the corpus. For our use case, we only have two possible 'words' of w, either it is 0 or 1.

For each pair of user and video, we have a sequence $S_{ui} = (s_1, s_2, ..., s_n)$. Based on all combinations of user and video pairs, we can calculate the empirical probability as depicted in Eq. 3.3 for various n-grams (in our use case, we limit the maximum n of 9, meaning that we only consider the last 9 occurences of the video), considering the sequences of S_{ui} as sentences. Our goal is to derive a table, similar to what is depicted in Table 3.1, for all possible combinations of 0 and 1 with a maximum length of 9. We note that the choice of 9 is quite arbitrary, but we show later in our empirical validation that small n gives already a reasonable performance gain in our model.

Sequence (S)	**Pr**$(next = 1\|S)$
11	0.7972
00	0.202
10	0.4445
01	0.5247

Table 3.1: Example of empirical probability of repeated item consumption for all items which have been presented twice to the users. The empirical probability is calculated based on the first fold of our dataset.

3.3.2 Incoporating the Empirical Probability of Repeated Consumption

When we describe our base algorithms in Section 3.2.3, we consider the fact that many users in the test sets have already watched the same music videos during the training period by predicting the user's score on the test set on that particular items with the average of their scores on the same items in the training set. This consideration is implemented in both of our base algorithms, **Pop_User** and **SVD_User**.

By merely taking the average playback ratio to predict the user preference for repeated items, we are practically omitting the order of the user's actions on that item historically. We hypothesize that this omission does not capture well the sequential nature of perception of an individual user when being presented the same music video multiple times and thus might affect the recommender system performance. It also does not capture the recency nature of the consumption. A simple example depicted in Table 3.1 illustrates that although the number of consumption and skip is the same, the items with a sequence of '01' leads to higher empirical probability to consume the item again than those with '10'. Note that the simple example presented here also illustrates the effect of recency, the empirical probability is higher when the user consumes the item more recently.

We propose a simple hybrid model that would take into account both the long-term user preferences, as indicated by the average playback ratio, and the effect of recency, which can be derived from the sequences of the actions. Weighted scoring of multiple personalization scores is also used in [21] to combine the score of the baseline algorithm with additional parameters, e.g., track repetition, artists, topic similarity, extended track co-occurrences, and social friends. Formally, now we can express the predicted rating using the following equations :

$$\hat{r_{ui}} = \alpha \cdot \overline{r_{ui}} + \beta \cdot p_{ui} \tag{3.5}$$

$\overline{r_{ui}}$ indicates the average rating on item i by user u obtained from the training period and p_{ui} measures the empirical probability of playing the item again, considering the sequences of previous user action on item i. These two measures are weighted by factors of α and β which sums up to one. Having defined the new proposed rating, we can compare the original model, which does not take into account the sequential nature of the actions, with the model that incorporates the sequential nature by considering the empirical probability derived using Eq. 3.3.

Note that in Eq. 3.5, setting the α to 1 means that we only consider the average playback ratio, omitting the empirical probability of repeated item consumption while setting the β equals to 1 simply omits the average playback ratio and only takes into account the empirical probability as the predicted score.

3.4 Like Prediction Task

So far, the session reranking task has not considered explicit feedback in the form of a 'like'. A 'like' is considered as an explicit positive feedback which indicates a stronger preference over an item, compared to a 'skip' or a complete consumption. We propose a 'like' prediction task to explore further the use of multiple types of feedback, as well as incorporating the user attention prediction into the recommender systems, which are related to answering RQ3 and RQ4 respectively. We show that within this task, by performing multi-level sampling method, we can improve the performance of one of the most well-known framework in pairwise learning to rank method for recommender systems, Bayesian Personalized Ranking (BPR), which highlights the importance of considering multiple types of feedback and user attention prediction.

3.4.1 Training Data Generation

Within the period of our training data, pairs of unique user and video are first identified. For each pair, we assign a rating of 1 if the video is liked by the user, and 0 otherwise. Note that this framing of the problem looks like an unary-rating implicit feedback recommender scenario. However, it is different because, in most implicit feedback scenarios, there is no negative feedback. In our case, we can infer negative feedback based on the music videos that are not liked by the user but have already been presented to them by the system. This implicit negative feedback can be considered as auxiliary feedback for our model. Although

we call the feedback as negative feedback, a music video which has not been liked still actually has the potential to be liked by the user in the future.

Several training matrices are derived based on the interactions data between users and items with multiple types of feedback. In all of the derived matrices, the value of the matrix is either 0 or 1, depending on the definition. Each matrix is a matrix which contains the same number of rows and columns. The rows represent the users while the columns represent the items. These matrices are described as follows:

1. Positive matrix P : The value is equal to 1 if a user, at least once, gives a 'like' to an item.

2. Watch matrix W: Using the formula described in Section 3.2.2, for each pair of user and item, the average playback ratio is calculated. If the average playback ratio is more than a certain threshold of T_{watch}, it is categorized as a *watch* event, and we set the value of item i for user u as 1 in W.

3. Skip matrix S: In contrast to the definition of watch matrix W, the value of item i for user u is 1 in S if the average playback ratio is less than T_{skip}.

4. Unobserved item matrix U_o: All items i which have not been observed by user u are assigned a value of 1.

Now that we have a finer grained definition of feedback, the better way of framing our problem is by treating the items as having graded relevance, instead of a binary value. For example, in an e-commerce website, a user can view, save, or purchase a product. Purchasing a product gives the strongest positive signal, while viewing or saving can be considered as a weaker positive signal. In the literature, this problem is considered as Heterogeneous Implicit Feedback (HIF) scenario [32]. If we want to recommend products that would be most likely purchased by the user, the hypothesis is by exploiting the fact that the user also views or save another item, maybe it might help to improve the predictive power of the model.

Some studies have modified the sampling strategy of the original BPR to be more adaptive and sensitive to the graded nature of the feedback [29, 30]. In [29], a variant of BPR called BPR++ is introduced. In BPR++, a function $pweight(u,i)$ for user u and i is used to measure the relative preference towards all positive items. In the original BPR, we cannot create triplets of user and items from two items with positive feedback, since we cannot infer the relative preference of the items. However, with the introduction of *pweight* function, two positive items i and j can be mapped into triplets if $pweight(u,i) > pweight(u,j)$. The authors propose to generate a new set of triplets candidate $D_S^{++} := \{(u,i,j)|pweight(u,i) > pweight(u,j), i \in I, j \in I\}$ where I is the list of positive items for each users. The sampling strategy to obtain the triplets is then biased with a probability β to be taken from D_S^{++} instead of taken from the original BPR sampling set.

[30] tries to exploit the fact that multiple types of feedback might be available, and using them might improve the performance of baseline BPR model. The authors split the feedback category into a target and auxiliary feedback. For example, in an e-commerce

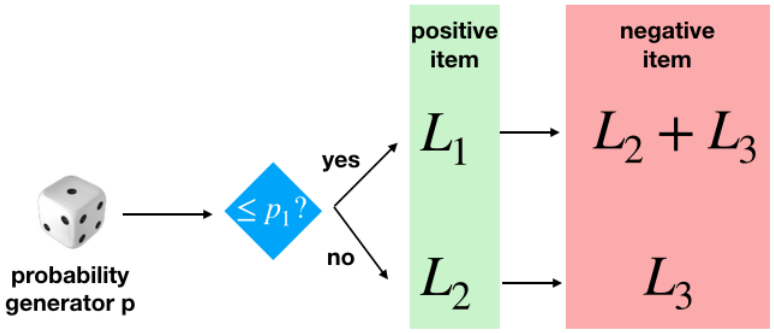


Figure 3.5: Proposed sampling strategy

setting, a purchase by the user is our target feedback, while a view, like, add-to-cart, might be considered as auxiliary feedback. They also suggest that these types of feedback have different levels of preference. Naturally, a purchase is located at a higher level than a view. By considering these different levels, the negative items sampler is derived such that it can also pick negative items from items in the lower level, instead of picking randomly from unobserved items.

3.4.2 Proposed Sampling Procedure

To realize the multi-level sampling strategy, multiple levels are defined based on the types of feedback that it represents. Each matrix defined in Section 3.4.1 is mapped into a level L_i. The assignments are : L_1 to P, L_2 to W, L_3 to S, and L_4 to U_o. Inspired by the approach in [29, 30], instead of sampling negative items randomly from items that have not been liked by the user, we can sample them from the items that have been presented to them.

Because many items are consumed more than once, not all presented items themselves carry the same level of preference. For instance, an item that is always skipped by the user might have less potential to be liked in the future than an item that is always watched completely, thus, is a more suitable candidate for negative item preference. This adds a little bit more complexity to the overall sampling strategy but gives more potential to learn better preference.

In practice, we propose a modified sampling strategy as illustrated in Fig. 3.5. While it is quite rudimentary, the decision to use the proposed sampling strategy is informed by the actual intuition that underlies our hypothesis. We want to test out whether using items which have been presented as negative feedback instead of random would produce better representation of user preference, which will eventually lead to higher performance of the recommender system model.

3.4.3 Evaluation Protocols

For the session reranking task, the evaluation is performed on a per-session basis. As has been described in Section 3.2.4, the underlying motivation is to approximate how the curation-based system works online. For the 'like' prediction task, we propose a slightly different evaluation protocol which extends beyond session-level evaluation.

Note that people rarely give 'like' feedback compared to another types of implicit feedback e.g. a 'skip' and a complete consumption. Thus, naturally, fewer sessions contain 'likes'. Moreover, we think that when a user presses 'like' on a video, it spans beyond session level preference. Although the video has been liked by the user once, the user might occasionally skip or consume the video completely on multiple sessions. Thus, we argue that it is not suitable to perform evaluation on a per-session basis for the 'like' prediction task.

Instead on a per-session basis, we propose to evaluate the task across all videos which have been presented to the user during the test period. The videos that have been liked by the user during the training period are excluded, however, those that have been presented before but have not been liked yet remain. The users who do not give 'like' to any video in the test set are removed from the test set. MAP is used as the evaluation metrics. In the test set, a video which is 'liked' at least once by the user is assigned a relevance value of 1, and 0 otherwise, regardless whether the video is skipped or consumed completely.

3.4.4 Algorithms

For the 'like' prediction task, Bayesian Personalized Ranking framework is used. We refer to the technical detail on how BPR operates in Section 2.3.3. The original sampling-based strategy for BPR is modified to take into account the fact that we also have auxiliary feedback, e.g. the implicit feedback represented by level L_2 and L_3, apart from the positive feedback, as described in Section 3.4.2. Note that we do not modify how the optimization is performed.

To formalize, compared to the original triplets (u,i,j) candidates to sample from BPR in Eq. 2.2, in the proposed sampling approach, there are two new sets of triplets candidates that are possible to sample from, as depicted in Eq. 3.6. In practice, every time a sample is taken, there is a probability p_1 that the sample is taken from D_{L_1}. Graphically, the sampling procedure is shown in Fig. 3.5.

$$\begin{aligned} D_{L_1} &:= [(u,i,j)|i \in L_1^u \wedge j \in (L_2^u \cup L_3^u)] \\ D_{L_2} &:= [(u,i,j)|i \in L_2^u \wedge j \in (L_3^u)] \end{aligned} \tag{3.6}$$

3.5 User Attention Prediction

> **RQ4** *Can we improve the performance of the recommender system by considering predicted user attention in a session?*

The hypothesis implied in RQ4 is that there is an effect on considering predicted user attention to the performance of the recommender system. If we can discard, or assign less

weight, the music videos in sessions during which the user does not pay attention to the television, we hypothesize that the model will be able to learn the preference of the user better. This hypothesis is quite intuitive because the user preference is affected by the music videos in their viewing history that they *actually* see and are aware of. Note that the prediction of user attention is implied since there is no ground truth as to whether the user is actually watching the television or not.

In [20], the user attention behavior on real-world dataset from Spotify is studied. The author proposes a simple time decaying factor as a measure of user attention, which exponentially decreases after an active action on the platform has been performed by the user. The idea is that an active action excites user attention. This is intuitive because a user is always paying attention on the application when performing active actions such as pressing like, skipping a song, or changing playlists. They perform extensive exploration on their study, however, the impact of user attention modeling to a recommender system task has not been explored yet.

To evaluate the effects of user mode prediction, an end-to-end evaluation protocol is proposed. Because we do not have ground truth, the user attention prediction cannot be evaluated on its own. Instead, we want to evaluate the effects on the final performance of our recommender system. This raises a question. What kind of task is suitable to check the benefit of incorporating user attention prediction?

We propose to use the 'like' prediction task to evaluate the impact of user attention prediction. Hypothetically, if a model *knows* when a user is actually watching a music video among continuous streams of a completely consumed videos, it should be able to derive a better representation of the user preference than a model which assumes that all completely consumed videos are watched by the user. We have not commented yet on the empirical validation of this hypothetical statement, but it is quite intuitive for us because if the user does not watch a music video, and the model *assumes* that s/he watches it, it will inevitably fail to infer the right user preference.

Next, we describe some of our thinking on how to perform user attention prediction. In [24], they propose that there are two modes when a user is listening to music, it is either they are in active or passive mode. We also think that user attention when watching the PMT can be represented by such binary status, either they watch and pay attention to the television, or not. In practice, we can also consider that the user enjoys listening to the music video only.

By observing the user behavior, we can already check whether a user is actively paying attention to the television or not from the presence of the user feedback. It is intuitive to say that when a user performs an active action by pressing a button on the remote control, they are indeed paying attention to the television. Thus, a moment after pressing the button, we assume that the user *still* pays attention to the television. Nevertheless, such assumption cannot be empirically checked since we do not have the ground truth of the user prediction when watching the television. Furthermore, we assume that when a user watches the videos completely in a continuous manner, it means that they transform into a passive mode, and the risk of the music videos not actually watched by them increases as the length of the continuous stream becomes larger.

3.5.1 Using Play Cut-off Threshold

The 'like' prediction task is used to check the benefit of predicting user attention on the performance of the recommender systems. The BPR framework is still used with multi-level sampling strategy as described in Section 3.4. The heuristics method to infer user attention considers only N videos consumed completely after the last active action taken by the user as positive implicit feedback. We assign N to a variable called *play cut-off*. For example, referring to Fig. 3.6, if N equals to 1, it means that only the first video consumed completely after any active action is performed by the user (e.g., pressing skip or like) is considered, and the rest of the videos are discarded as if the user does not watch those videos.

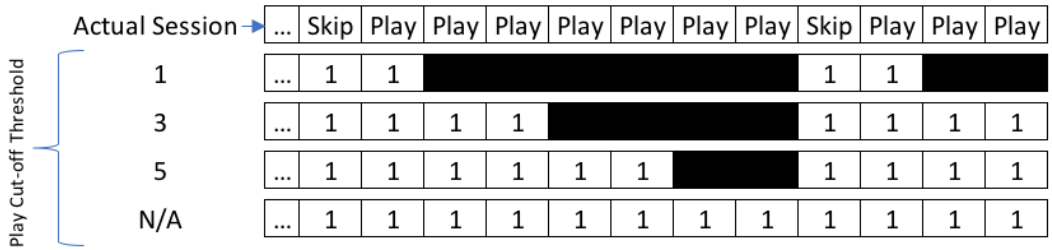


Figure 3.6: Illustration of using different play cut-off threshold

The BPR multi-level sampler framework also enables us to regulate from which collections of data that the positive and negative items are sampled. To show the utility of inferring user attention, we propose to evaluate the model on the scenario where multi-level sampling strategy is used with the probability of sampling positive items from items that have been liked by the user, equals to *zero*. When the probability equals to zero, it means that we only rely on the 'skip' and completely consumed videos as implicit feedback to pick the positive and negative items sample from each user. The formulation of the problem now enables us to really focus on the importance of inferring liked items from the implicit feedback.

Imagine a use-case that a user u' completely watches a video v' 5 times, and skips the video three times all with the same playback ratio of 0.05. Now, let the play-cutoff threshold is set to 1. The baseline method which incorporates all played events will assign a preference score of: $\frac{1}{(5+3)}(1 \cdot 5 + 0.05 \cdot 3) = 0.6438$. However, after observing the position of the completely consumed videos, only one of them is watched right after any active action is taken by u'. The rest of the video is consumed during inferred passive mode. Using the play cut-off threshold, we only consider 1 among 5 occurrences of completely consumed videos to be considered in the calculation of the user preference score. Now, the preference score becomes : $\frac{1}{(1+3)}(1 \cdot 1 + 0.05 \cdot 3) = 0.2875$.

Furthermore, because we define items having average playback ratio score of 0.5 as positive items candidates, using the baseline method, video v' will be considered as positive for user u'. In contrast, using the play cut-off threshold, v' will be assigned to the negative items candidates. Based on this simple example, we highlight the effect of inferring user

attention which can change the implicitly positive rated items to negative and vice versa.

Chapter 4

Implementation and Data Analysis

This chapter describes the experimental setup to answer the research questions by using approaches described in Chapter 3. Some preliminary data analysis is also described, which is not directly related to answering the research questions but serves as an empirical verification as to why repeated item consumption behavior needs to be taken into account.

4.1 Dataset

In this section, some characteristics of the dataset are explained, including necessary statistical information, to describe some intuitions on the distributions of the data and the relevance to our research questions.

4.1.1 Ethical aspects on the dataset collection

With the enforcement of General Data Protection Regulation starting on May 25th, 2018, company which processes European citizen's personal information needs to take extra measures to manage the users data. Careful consideration on how to provide the dataset for this study was made by the company. The data given by the company does not include any private and sensitive information whatsoever that could potentially lead us to determine the personal identity of real users, such as the name, address, gender, or email. The fields that contain information about the user identity are thus anonymized. During the period of the internship to finish the thesis, we are not allowed to access the production environment, which by large limit ourself to fully develop a system that can be readily used. However, based on the frequent discussions with XITE technology team, we can understand and think about how to implement the solution if access is given to the real environment. For this study, the dataset is provided as text files which can only be processed offline.

Moreover, regarding the ethical processing of the historical user behaviour, referring to the Terms of Service of the company[1], when the user consent to it, the company are allowed to process information provided by the users to give better service and personalization. This gives legal background for the company to perform studies and research based on the

[1] https://xite.nl/pdf/PRIVACY_XITE_PMT.pdf, accessed on July 20th, 2018

historical user data. A Non Disclosure Agreement (NDA) was signed in the beginning of the thesis to make sure that I will only use the data provided by the company during the duration of the internship.

4.1.2 Descriptions

The dataset consists of anonymized user actions on the XITE PMT platform within several months in 2017. Each entries contains information on : *userId*, *videoId*, *sessionId*, *channelId*, *timestamp*, *actions*, *durations (in seconds)*, and *playback percentage*. Explanations on each columns are described in Table 4.1. Furthermore, the statistics of the dataset are described in Table A.1a and A.1b (in the Appendix). For confidentiality reason, this table is not shown in the public version of this thesis. The size of the dataset is large enough and deemed suitable to be used for this thesis.

Field	**Description**
userID	The anonymized ID of the user
videoID	The ID of the music video
sessionID	The ID of the session in which the music video is played
channelID	The ID of the channel. The channels can be mapped to another mapping from channelID to channel's name. In this thesis, we divide the channel into three categories of channel : Now, Mixer, and Others.
timestamp	The timestamp, represented in YYYY-MM-DD HH:MM:SS e.g. 2017-01-01 10:20:30
action	The action given by user for the music video. There are different types of actions : play, skip, like, and leavechannel.
durations	The duration of the music video played in seconds
playbackPercentage	The playback percentage is defined as the ratio between the duration and the length of the music video.

Table 4.1: Description of the fields in the dataset

4.1.3 Preprocessing

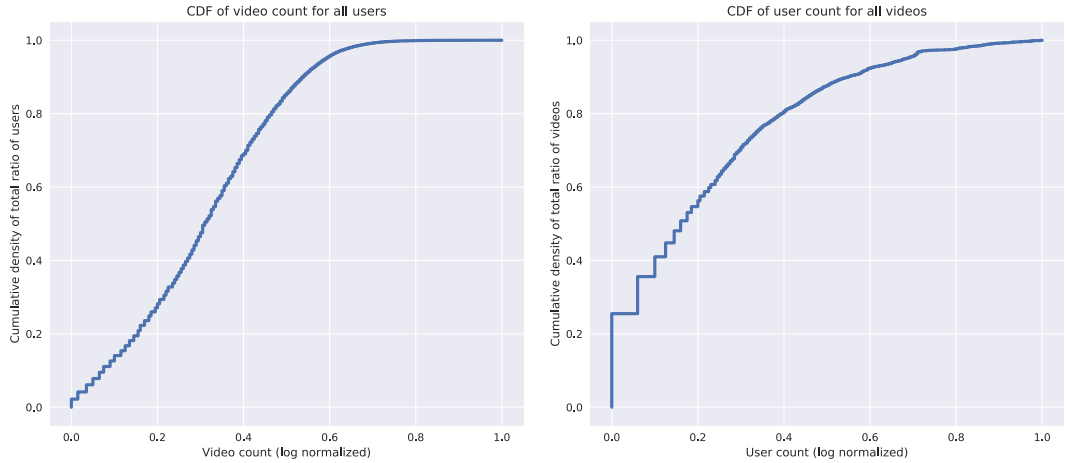
The users in the training sets are filtered so that those who watch less than ten music videos are removed. Such data filtering strategy is common in several recommender system research to remove the effects of cold-start issues for inactive users. User and music videos in the test set that are not present in the training sets are also removed. Table A.1a and A.1b illustrates the general statistics of the dataset. In general, in terms of size, we are confident that the dataset is large and representative enough to be used in our study.

4.1.4 Distribution of data

Visualizing distribution of our dataset with respect to various statistics help us to understand the characteristics of user behavior. In the following points, we will report such visualization and describe intuitions that can be drawn from them. The following exploratory data analysis is performed on the first fold only of our dataset. However, we can expect similar trends over different folds as well.

User and video count

Fig. 4.1 illustrates the distributions of total unique video played per user and vice versa. A small number of videos are very popular and played by a lot of users. As for the user's statistics on the left graph, a small number of users watch a large number of videos. It can be the case that they are a very active user, but it is unlikely that those users actively watch all of these videos completely. It can be the case also that these users turn on the television without paying attention to it. Note that the x-axis is log normalized to obscure the absolute number of users and videos.



(a) CDF of video count for all users

(b) CDF of user count for all videos

Figure 4.1: Cumulative distribution function (CDF) for videos and users. In (a), only few users watch a large number of videos, and in (b), only few videos are watched by lots of users.

Score distribution

Fig. 4.2 illustrates the distribution of the average playback percentage per user on all unique tuples of user and video. The distribution forms a bimodal distribution where lots of videos are either completely played or completely skipped.

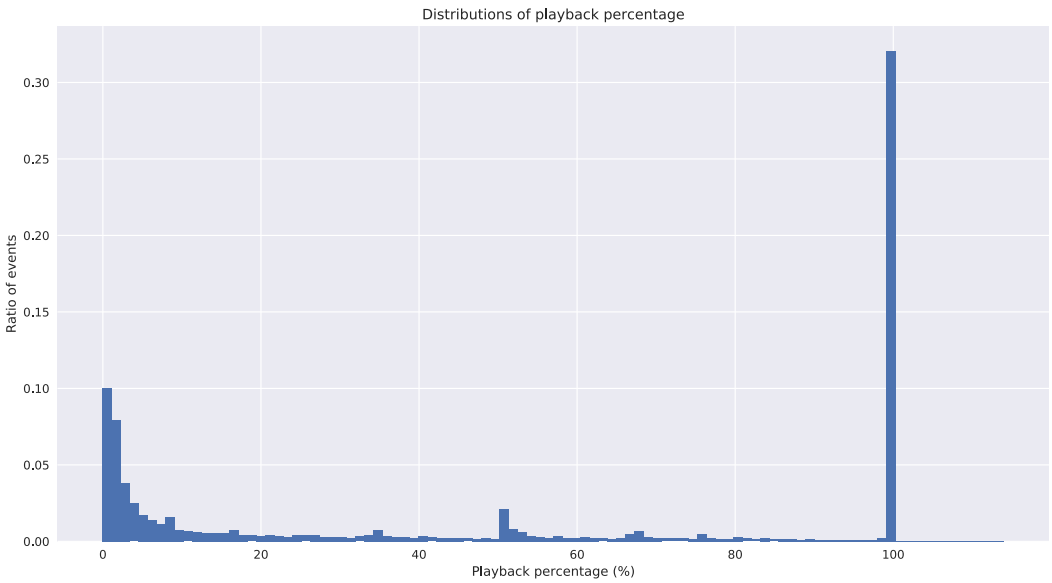


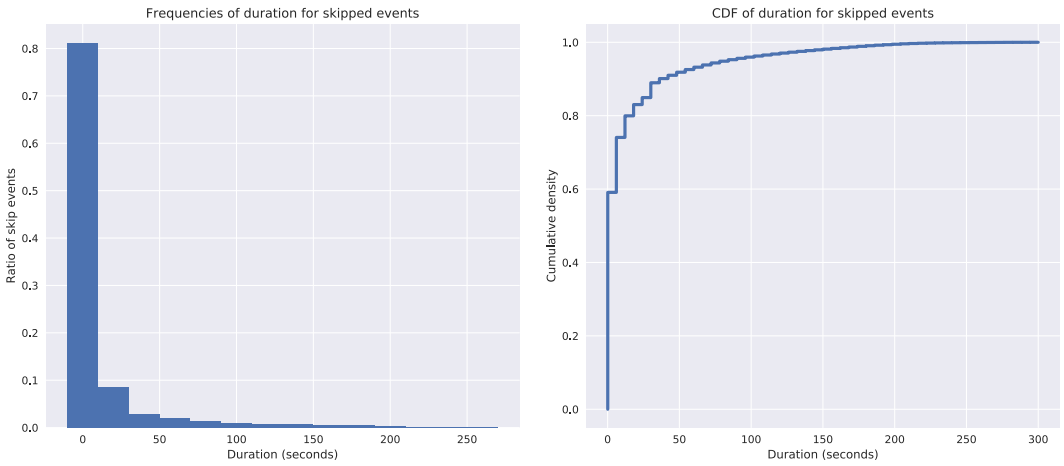
Figure 4.2: Distribution of average playback percentage. Most videos are either skipped in the beginning of their playback or completely played.

Skipping behaviour

People intentions when skipping a music video can be ambiguous. A user might skip the video because she does not like it, and prefer not to be shown the video again, or simply because they do not want to watch it completely now, but does not have a strong dislike toward the item. Thus, we need to find a way how to interpret such signals from the skipping behavior. Since we do not have the ground truth whether the user skips because of dislike or not, what we can do is to infer from the data. Duration is one signal that we can use, particularly because skipping music video after 2 or 3 seconds likely carries different intention than skipping signal after 50 seconds, for example. Fig. 4.3 illustrates the duration of all the 'skip' events in our training dataset. Based on the cumulative density function (CDF), we notice that 80% percentile of the duration data is within less than 30 seconds.

Distribution of channels

As discussed in Section 2.6, there are various channels that the user can choose when opening the XITE PMT application. Those channels differ in characteristics such as the genre, popularity, or the era of the music videos. This distinction might be interesting because for example, a recommender system model that favors popularity might work well in one channel while working poorly in another channels. Table 4.2 illustrates the distribution of sessions in each channel. The 'Now' channel has a particularly high percentage because it is the default channel presented to the user when opening the application.



(a) Frequency of duration for skipped videos (b) Cumulative density function

Figure 4.3: Frequency and cumulative density function (CDF) of duration for skipped videos. For clarity of the graph, the duration is capped on 300 seconds.

Channel	Percentage
Now	84.69
Others	12.59
Mixer	2.20
Favorites	0.53

Table 4.2: Distribution of channels across sessions

4.1.5 Implicit and Explicit Feedback

For the 'like' prediction task, we exploit the presence of implicit and explicit feedback to provide a recommendation list that is optimized to rank the items candidates in the test set so that the liked items are put earlier in the list. Similar to what has been described in the previous section, experiments are run over four folds. For each fold, a new form of the dataset is generated as described in Section 3.4.1.

The statistics of the dataset used in 'like' prediction task are described in Table A.2. For confidentiality reason, this table is not displayed in the public version of this thesis. Apart from the number of 'like', we also report the number of items with an average playback ratio greater or less than 0.5. As previously mentioned in Section 3.4.1, the division of items based on the average playback ratio is used to create levels across different types of feedback.

4.2 Implementations of the Recommender Algorithms

Based on the choices of algorithms that are described in Chapter 3, two types of baseline algorithms are implemented, namely matrix factorization with FunkSVD and Bayesian

Personalized Ranking (BPR). Due to the practicality nature of this thesis, a unified programming language to perform all necessary steps, from data analysis to the implementation of the algorithms, is needed. It is decided that Python is used because it is currently the most popular language for data science and machine learning. The FunkSVD implementation is taken from Surpriselib[2] while the baseline BPR is taken from Implicit[3] library.

Surpriselib offers various baseline algorithms based on neighborhood and matrix factorization methods. It also offers convenient methods to perform standard evaluation strategy such as cross-validation and various evaluation metrics. This library is mainly used to perform experiments on the session reranking task. It is used to train the FunkSVD model and assign predicted scores based on the chosen algorithms.

On the other hand, Implicit provides recommender systems algorithms for implicit feedback datasets. While it provides API for Python, the underlying algorithms are implemented in Cython, which makes the performance of this library very fast. Currently, this library implements Alternating Least Squares [16], Bayesian Personalized Ranking [33], and Item-Item neighborhood model. We use the BPR implementation of Implicit for 'like' prediction task. Note that the original BPR implementation does not offer a way to sample from multiple types of feedback. Due to this reason, some modifications are performed.

For both libraries, only the model training part is used, which basically implements algorithms to train the model and assign predicted scores on the item. The data preprocessing and evaluation part is implemented by ourself. To perform evaluations, a standalone Python script is used. However, for most of the data pipelines, it is decided to implement in Jupyter notebooks [23] to maintain good reproducibility of the experimental setup.

4.3 Experimental Setup

This section describes the specific setup for each task that is conducted in order to answer the research questions.

4.3.1 Session Reranking Offline Evaluation Benchmark

It is important to design an experimental setup which resembles actual hypothetical use case of the implemented recommender system in the production environment at XITE. As described in Section 3.2.4, a time-based split of training and test data is chosen. This is quite logical since eventually, the recommender system should be trained on some selected data within a training period, and implemented live to provide recommendations for the upcoming interactions.

For both session reranking and 'like' prediction task, we are focused on evaluating recommender system performance by using ranking-based metrics. Therefore, the actual predicted score by itself is not that important, instead, the ordering of the items per session, for the session reranking task, and per user, for 'like' prediction task, that matters.

[2] http://surpriselib.com/
[3] https://github.com/benfred/implicit

For session reranking task, the average playback ratio is used as features to fill the user-item matrix. The predicted score therefore naturally translates to the predicted average playback ratio of the item in the test set. This is however not the case in practice since the predicted score is merely used to order the item in each session in the test set. The practical implication is that rather than viewing the average playback ratio value *as-is*, it is therefore better to view it as a representation of the user preference level. The higher the score is, the more the user prefers to watch it.

4.3.2 Incorporating Satiation

In experiments related to incorporating boredom by considering repeated consumption item behavior, repetition even might occur in the test set. To take into account the number of times that an item has been played by the user, we store the sequential actions for each item in a variable, which is updated everytime an action is performed to that item in the test set. Thus, the repetition is not calculated based only on the occurrences of the item in the training set, but rather updated every time the item is presented to the user in the test set as well.

By using Equation 3.3, we can find the empirical probability of consuming the item again given certain sequences of past user actions on the same items from the training data. Since a music video can be repeated multiple times, it is therefore important to think about the length of the historical actions on the same item that must be considered. For example, using only a length of two perhaps is too short. A maximum length of five is considered, meaning that if a music video is seen more than five times, we only consider the last five actions on that item. While this choice seems arbitrary, it is observed that in the dataset, more than 90% percentile of repeated items are less than 5 in length.

Moreover, we want to explore the effects of using different values of α and β to weight the importance of the empirical probability of consuming the item again as has been previously described in Eq. 3.5 at Section 3.3.2. We perform a search for optimal β within the range of [0, 0.3, 0.5, 0.7, 1]. Note that if the β is equal to 0, the model effectively does not use the empirical probability score, while in contrast, if β is equal to 1, it means that only the empirical probability score is used.

4.3.3 Incorporating Implicit and Explicit Feedback

In this experiment, various values of probability of sampling positive items from liked items are used to realize the multi-level sampling strategy. Specifically, we conduct the experiment using the following values of p_1 : [0, 0.3, 0.5, 0.7, 1.0].

4.3.4 Incorporating User Attention Prediction

For this experiment, we compare the performance using various play cut-off threshold of 1,3, 5, and none. Furthermore, we run the experiment using multi-level sampling strategy on the like prediction task when only sampling positive items from music videos having average playback percentage of 0.5 and negative items otherwise. We compare the difference in performances both in predicting the likes for all and unseen items.

4.3.5 Hyperparameter Tuning

Hyperparameter tuning is important in any machine learning model, but we found that not all hyperparameter are equally important in our use case. For example, we found that the number of factors does not affect a lot the performance of the FunkSVD and BPR algorithm. For BPR, we use the default factors implemented in Implicit library, while for FunkSVD, we chose a factor of 10. Learning rate and regularization, however, turns out to be quite important in our BPR model. We systematically evaluate the performance across different values of learning rate and regularization parameter (0.1, 0.01, 0.001), and chose the best one for each split of the dataset as the reported performance. Generally, using learning rate and regularization parameter of 0.01 yield the best performing model across all folds in our dataset. Since we are not focused on reporting the best performing hyper-parameter configurations from FunkSVD and BPR, we do not report the performance difference across different hyperparameter with these models in this thesis.

Across all experiments, statistical significance check is used to evaluate whether the results from various recommender system models are significantly different from each other. Wilcoxon signed rank test is used, and the result is reported as significant if we achieve $p < 0.01$ as compared to the base algorithms.

4.4 Empirical Analysis of Repeated Item Consumption

In this section, some illustrations are given to provide intuition on why different sequences of actions might lead to different probability to consume the item again. We refer to the derivation of the empirical probability of repeated item consumption which has been described in Section 3.3.

Table 4.3 shows two examples where sequences with the same number of consumption and skip can have different values of empirical probability of repeated item consumption. We can draw two interesting insights based on the table. First, if we look at the case where the number of consumption and skip equals to 2, we can observe noticeable differences in the empirical probability among the first and last three of the sequences. The lowest one (0.42) reflects the case where the user skips twice during the last two occurrences of the item, while the one with the highest score (0.58), on the contrary, consumes the last two occurrences of the items. This finding shows that recent actions toward items might carry more influence on the probability of consuming the item again.

Secondly, if we look at the case where the number of skip is zero, it shows that as the user consumes more often of the same video, the empirical probability of consuming it again also increases. This finding shows that when a user consumes items repeatedly without giving any skips, higher of exposure to the same items increases the probability that they would consume the item again.

Both of these findings are only drawn from the statistics in Table 4.3. Next, the generalizability of the findings are discussed based on the study performed in [2]. We have described the motivation of using this framework in Section 2.4.

The domain on which the repeated item consumption study is performed in [2] is quite different compared to the characteristics of the PMT. In particular, the capability for the user

#Consumption	#Skip	Sequence	Probability to Consume Again
2	2	0011	0.58
2	2	1001	0.53
2	2	0101	0.54
2	2	1010	0.45
2	2	0110	0.45
2	2	1100	0.42
4	0	1111	0.87
3	0	111	0.84
2	0	11	0.80
1	0	1	0.73

Table 4.3: Example of empirical probability of consuming the item again considering past sequences of user actions on Fold-1 of our dataset.

to select a specific item intentionally is quite limited in the PMT. Nevertheless, with some modification suitable to our use case, the framework can still be used to empirically check the presence of popularity, satiation, and recency, in our dataset. The following analysis is performed on the first fold of our dataset, but we expect that in general it also applies to the other folds as well.

1. **User Appreciation**

   In [2], popularity is the first factor which affects the user behavior to consume item repeatedly. The term popularity refers to an 'individual' level popularity. We rephrase the term popularity with user appreciation because in our use case, the user can only react to items which are presented to them. Thus, user appreciation indicate how the user appreciates the items that are presented to them. So far, we have defined binary judgement to indicate appreciation towards an item, which relates to the behavior whether the user consumes or skip an item. Our hypothesis is that people will more likely consume the items with a higher ratio of consumption in the past.

   For example, imagine that we have two groups of users who have been presented an item 10 times. The first group consumes six times (and skips four times) while the second one only consumes twice (and skips eight times). If we give the same item again for both groups and our hypothesis is true, the first group will have a higher chance of consuming the item again, because they appreciate the item more than the second group in the past.

   Referring to Section 3.3.1, a session S_{ui} is defined as time-ordered binary relevance value for item i to user u with length n. In practice, S_{ui} consists of all possible permutations of 0 and 1 in a sequence of length n. For each permutation S_{ui}, the consumption ratio is calculated by taking the average of the binary relevance value. For example, if $S_{ui} = (0,1,1)$, then the consumption ratio equals to 0.67. Furthermore, for each

permutation S_{ui}, the empirical probability of repeated item consumption can be calculated using Eq. 3.3. Finally, we have tuples of sequences S_{ui}, the consumption ratio and the empirical probability for repeated item consumption. The consumption ratio and the empirical probability are plotted in Fig. 4.4. Note that for the same value of consumption ratio, there might be different sequences having different empirical probability, for example, sequences of (0,0,1) and (1,0,0). This is why we plot the graph as a boxplot.

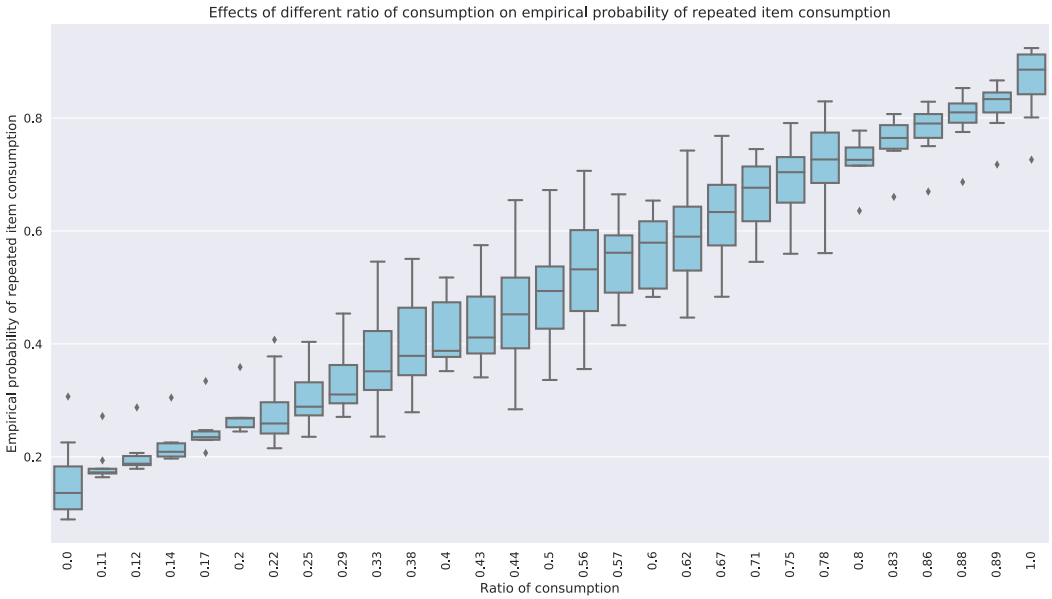


Figure 4.4: Effects of different ratio of consumption on empirical probability of playing an item again

Based on the Fig. 4.4, we can see that higher consumption ratio leads to a higher empirical probability of consuming the item again. This fact supports our hypothesis that indeed better user appreciation in the past towards an item leads to a higher probability of consuming the item again.

2. **Satiation**

There can be various perspectives to check for the existence of satiation in our use case. First, we can observe the empirical probability as the function of an increasing number of repeated stream. If boredom exists, we expect that as the number of repeated stream (with no skips) grows, the empirical probability of playing the item again will decrease. The underlying motivation is that people will then get bored after streaming the same item again for a certain number of times k. This definition is similar to how the satiation is defined in [2].

Another perspective is that the presence of satiation can be defined as the changing of user preferences under repeated exposure to items. Given the same amounts of

exposure to items, if there is no satiation, then the user preference would be fixed, either the user always skips (because s/he does not like it), or always consumes the items because s/he enjoys it. This definition is more suitable for our use case because it captures both actions, consumption and skip, towards items, as opposed to the first perspective, where we only focus on the user who always consumes the item.

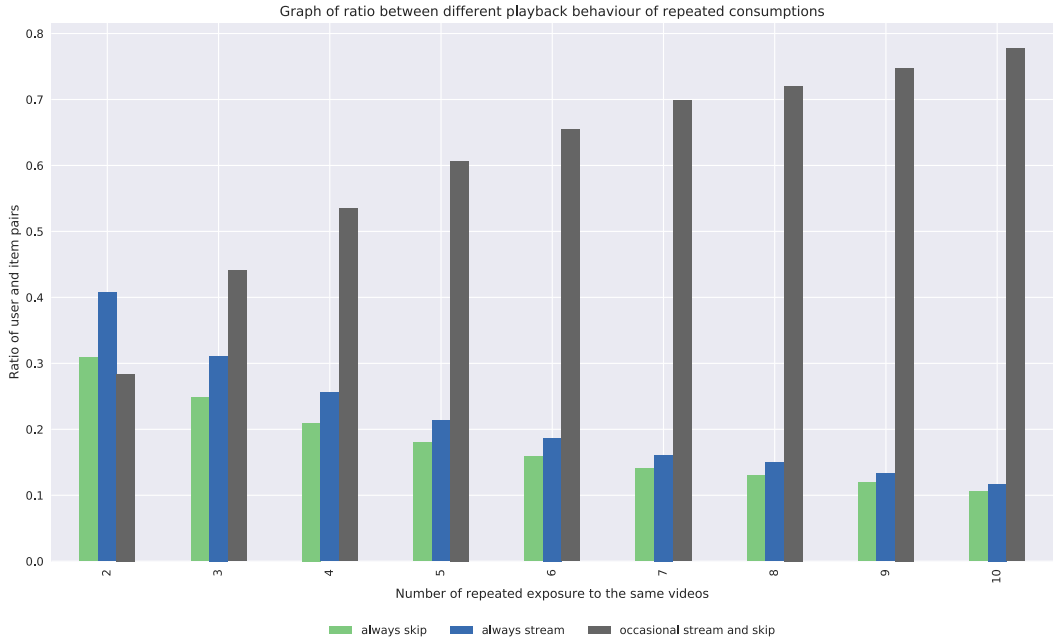


Figure 4.5: Ratio between different number of repeated item consumption

Fig. 4.5 shows the ratio of three behaviors: always consume, always skip, and occasionally consume and skip, across different numbers of repeated exposure to the same item. The ratio is calculated by dividing the total occurrences of each sequence of actions with n length by the total occurrences of sequences having n length. The sequence which only contains 0 and 1 is mapped into 'always skip' and 'always consume' category respectively. Otherwise, it is mapped into 'occasionally consume and skip'.

Based on Fig. 4.5, it can be observed that as the number of repeated exposure grows, the proportion of the user and item pairs falling into the category of 'occasionally consume and skip' increases at the expense of decreasing number of 'always skip' and 'always consume'. The 'occasionally stream and skip' ratio is also high even when the number of exposures is two. It means that the user's appreciation towards the same item is not fixed and has more fluctuation as the frequency of exposure increases, which is intuitive since the possible permutation of 0 and 1 in a sequence for longer sequence is also higher. Nevertheless, analysis on smaller number of exposure, e.g. two or three, already confirms that indeed satiation exists because there are some reasonable number of users who react differently during different times towards the

same items.

Special care needs to be taken when interpreting this finding. In particular, when observing the consumption behavior, because it might be originated from lean-back behavior from the user where they just turn on the TV while not paying full attention to it. Due to time-constraint, further exploration is not performed on this matter, and we leave it as potential future work.

3. **Recency**

Recency effects occur when, given that the user is presented with the item that s/he has seen before, s/he tends to consume the items when it is consumed recently, rather than in the past (and skips them recently). To measure the recency effect, a definition of recency need to be derived first. Let the recency definition described as *recency factor*, which is calculated based on the sequence S_{ui}. The recency effects indeed occur if there is a high positive correlation between the recency factor and the empirical probability of consuming the item again.

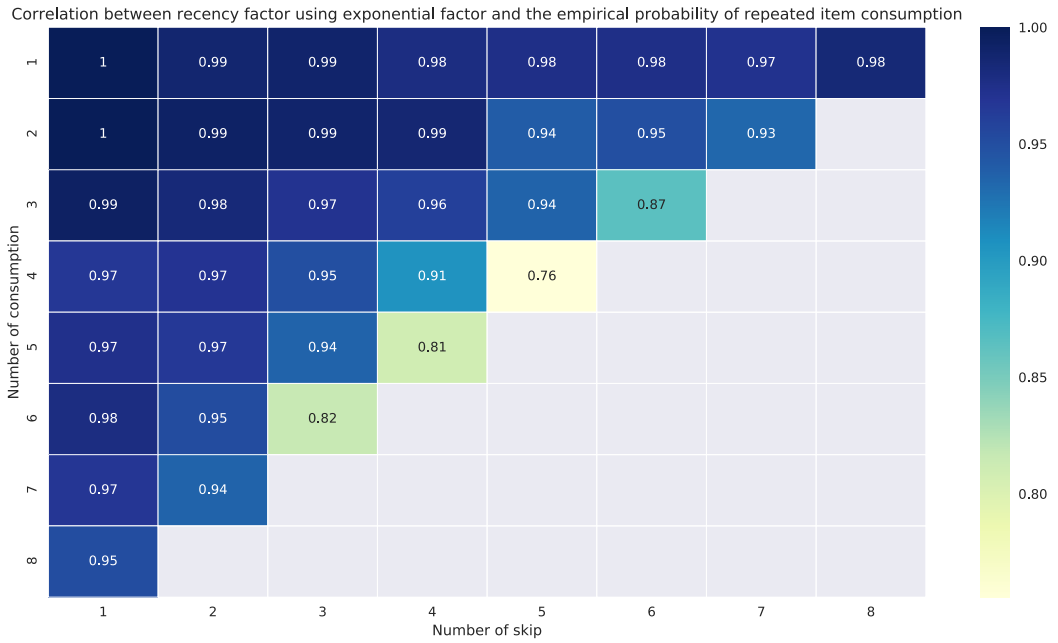


Figure 4.6: Correlation heatmap of recency exponential factor and empirical probability of consuming the item again

We refer to a study conducted in [26] to pick few metrics that are suitable to be used to explain the recency factor. In [26], some decay functions are used to model recency of user interactions, which eventually are used in a k-NN based model as a pre and post-filtering strategy to weigh either the user factor or the predicted recommendation score. We use the exponential and power decaying factor with a factor λ of 1. In their study, the optimum value of λ needs to be empirically validated based

Algorithm 3 Pseudocode to calculate recency factor

procedure RECENCYFACTOR(L)
Calculate length of the list L as len_L
Reverse the order of list L as L_rev
Calculate number of relevant items n_rel from L
Calculate list L_pos_rel which contains the position of each relevant items in L_rev

$Factor_{exponential} = \frac{1}{n_{rel}} \sum_{i \in L\_pos\_rel} e^{-i}$
$Factor_{power} = \frac{1}{n_{rel}} \sum_{i \in L\_pos\_rel} \frac{1}{i^{\lambda}}$
$Factor_{linear} = \sum_{i \in L\_pos\_rel} \frac{(len\_L+1)-i}{n_{rel} \cdot (len\_L+1)-(2 \cdot n_{rel}-1)}$

return $Factor_{exponential}, Factor_{power}, Factor_{linear}$
end procedure

Sequence	Emp. Prob	Recency Linear	Recency Exponential	Recency Power
0011	0.58	1.00	0.25	0.75
0101	0.54	0.86	0.21	0.67
1001	0.53	0.71	0.19	0.63
0110	0.45	0.71	0.09	0.42
1010	0.45	0.57	0.08	0.38
1100	0.42	0.43	0.03	0.29

Table 4.4: The recency factor for all possible permutation of S containing 2 consumption and 2 skip

on the dataset, however in our case, tuning the λ is not needed because the decaying function is merely used to explain the recency effect, rather than being used in the recommendation model.

Note that when measuring user appreciation earlier, tuples of sequences S_{ui}, consumption ratio, and the empirical probability are derived for all possible permutations of S. For each tuples, the recency factor is calculated based on Algorithm 3. To give a more concrete example, Table 4.4 shows the recency factor for all possible permutations of sequences containing 2 consumption and 2 skip.

Based on the recency factor calculated for each sequence, for various combinations of number of consumption and skip, a heat map can be drawn to help aid understanding the presence of high correlation between the recency factor and empirical probability of consuming the item again. Fig. 4.6 shows the correlation coefficients of the recency factor using exponential factor and the empirical probability of consuming the item again. We can see high correlation presents across all combination of consumption and skip, which support the argument that more recent consumption correlates well with a higher chance of the user to consume the item again.

Chapter 5

Results

This chapter presents the results of the experiments conducted on the methods described in Chapter 3. First, the experimental results on the session reranking task are described. Building on empirical findings about repeated item consumption behavior, we present an experimental evaluation on an approach that includes the empirical probability for repeated item consumption into a session reranking model.

Next, we move to a different task; we evaluate various models for 'like' prediction task. We test various assumptions on inferring positive and negative feedback from historical data and modify the sampling strategy of BPR to take into account these assumptions. Considering that many sessions are composed of passive stretches where there is long continuous stream of complete consumption without active user interruption, we test out the effect of predicting user attention on the recommender systems performance. Finally, we close off this chapter by performing analysis on the popularity aspects and the genre incoherence of the recommendation lists, based on the 'like' prediction scenario.

5.1 Session Reranking Evaluation Benchmark

Our first set of experiments is conducted over all sessions in the test set and related to answer RQ1: *Can we successfully rerank sessions in an offline experiment for recommender systems evaluation for curation-based music video television?* The result is presented in Table 5.1. For this set of experiments, it is worth noting that all models outperform the random baselines. Furthermore, the overall best performing model is SVD_User, which outperforms the random baseline by 0.0946 in terms of MAP, and 0.0694 in terms of WSR, for the 'All' channels category. Although we notice that the difference with the second best performing model, Pop_User, is statistically significant with $p < 0.01$, the difference between the MAP and WSR value is relatively small. For the MAP, the difference between SVD_User and Pop_User is 0.0052 while for WSR, the difference is 0.0043, both for the 'All' channels.

We also reported the performance across different types of channels. Note that each channels contain different flavor of videos such as the genre and the popularity, which are curated by the music experts at XITE. The performance ranking of the models across dif-

ferent channels is consistent, as well as the trend between MAP and WSR. Due to the consistency across different channels, in later analysis, the performance is only reported for the 'All' channel.

	MAP				WSR			
Model	**All**	**Now**	**Others**	**Mixer**	**All**	**Now**	**Others**	**Mixer**
Random	0.5584	0.5606	0.5561	0.5027	0.1756	0.1768	0.1778	0.1257
Pop	0.5945	0.594	0.6062	0.565	0.198	0.1978	0.2077	0.1661
SVD	0.6036	0.6034	0.6147	0.5699	0.2031	0.2021	0.2163	0.1809
Pop_User	0.6478	0.6515	0.637	0.585	0.2407	0.244	0.2305	0.1833
SVD_User	**0.653**	**0.6568**	**0.6421**	**0.5872**	**0.245**	**0.2477**	**0.2379**	**0.192**

Table 5.1: Evaluation for various recommender models considering all sessions, including those sessions containing repeated item consumption, averaged across all folds. The results are reported for each types of channels: 'All' indicates that all types of channels are included. The best performance for each channel is written in bold.

5.1.1 Discussion

Based on the findings described in Table 5.1, the proposed evaluation setting is validated. We tested a relatively simple models (SVD and Popularity), and it turned out that the result is quite reasonable, in a way that there is noticeable performance difference using various models, especially when compared to the random baseline. Note that it is not our intention to find the *best* performing algorithms. Such a study, which would focus to compare various recommender systems models, can be conducted on top of the evaluation framework that we have proposed in this thesis. Rather, it is our intention to validate the necessary evaluation framework needed to explore further challenges, related to repeated item consumption behavior, that are described in the next section.

Furthermore, careful attention must be given to interpret the result with respect to the characteristics of the sessions. One might have the following two questions: *1) Why is the score quite high, even for random?*, and *2) Why is the difference between random and the best performing model quite small?* We consider the answers to these questions in the following discussion.

In the evaluation phase, sessions might have a different number of length and relevant items, and the average of the evaluation metrics is taken across all sessions. We notice that there is a large number of sessions where the user just mostly plays the videos and only skips very occasionally. This behavior will affect the value of the evaluation metric because, with a high percentage of relevant items in a list, the MAP would naturally be high.

Typically, in most offline recommender system evaluations, a fixed length of the recommendation list is defined, and the metrics are calculated with respect to the fixed-length list for all users. Such method is driven by an assumption that if an actual recommendation list is to be provided to the user, there are only limited number of items which are reasonable to be displayed to the user, for example, because of the visual constraint on the application

itself, e.g.,, the size of the browser and the display size of the mobile telephone. A small number is usually chosen such as 5 or 10, depending on the domain of the applications.

Our evaluation framework, on the other hand, is based on different perspective. We evaluate the capability of different models to perform session reranking task. We argue that using cut-off in our evaluation metrics e.g., MAP@K would give a different interpretation. Cutting off the list into a fixed-length K means that we add another assumption that hypothetically the user only enjoys the first K videos produced by our model in a session. Instead, we keep the session length fixed and we propose to perform more in-depth analysis on different values of consumption ratio across sessions. This analysis is related to our experiments to answer RQ2 as well, so it is presented in Section 5.2.2.

For this experiment, there is no prior baseline to compare to, since there is no prior work at the company to evaluate the recommender system model offline. As a sanity check, it is crucial that the model outperforms the random baseline, which they all do. Furthermore, considering the lack of baseline, we set out to propose our first experiment as a baseline for more elaborate methods that we describe in the next section, which is based on repeated item behavior.

The result that we have obtained also serves as an empirical validation that the proposed experimental setup does indeed make sense, and eventually, it helps us to answer RQ1 that: yes, by exploiting playback ratio as feature, and using SVD algorithm combined with the individual average playback ratio of each user for each item, we are able to perform session reranking which is better than the random baseline.

5.1.2 Intuitions on the MAP Difference

Looking at the small difference in terms of MAP between SVD_User and Pop_User, a question might arise whether this small difference will bring noticeable effect on the final perception of the user. In particular, when this result must be communicated to non-technical persons, an intuitive explanation is needed to justify the performance difference across different models. This might also be relevant in our work because eventually, the recommendation setup would need to be explained to the music curator team.

No	List	Average Precision
1	[1,0,0,0,0,1,1,1,1,0,1,0,1]	0.5573
2	[1,0,0,1,1,0,1,1,1,0,1,0,0]	0.6571
3	[1,0,0,1,1,0,1,1,1,0,0,1,0]	0.6495

Table 5.2: Average precision for three different lists

An illustration is given to help give some intuition as follows. Imagine that we have three lists as depicted in Table 5.2, which hypothetically originate from the result of session reranking produced by three different models. These lists have an equal number of relevant items and length. Note that the average precision is almost similar to the result presented in Table 5.1, in particular for the random, SVD_User, and Pop_User respectively.

The second list has higher AP compared to the first list because some relevant items are put in higher position, where the position is quite early in the list (the fourth and fifth

position among the list of length 13). The third and second list only has small AP difference because the second list ranks one item higher in the latter part of the list, where the effect of increasing by one position is much less than increasing position earlier in the list.

Considering assumption that users prefers to get relevant music videos earlier in the list, based on this simple illustration, we can say that the utility of increasing MAP by small number does not bring many benefits to the user since the increase is due to putting items higher in the latter part of the list. On the other hand, a large difference between MAP indicates the ability of the model to put items higher earlier in the list. This illustration implies that although the results between SVD_User and Pop_User is statistically significant, due to the small difference between the MAP value, the actual 'offline' user perception when being exposed to both algorithms might be similar.

5.2 Incorporating Satiation

In this section, we describe some experimental findings to answer RQ2: *Can we improve the recommender system performance on session containing items which have been played before by the user by considering the sequence of repeated item consumption behavior?*.

After performing an empirical analysis of repeated item consumption behavior that is described in Section 4.4, empirical validation of the results on including the empirical probability into the recommender model is presented. The empirical probability is used to modify the final predicted score for already seen items. In a session, there are three different cases with respect to the occurrences of repeated items in it: (1) no repeated item, all items are new to the user (2) some of the items are new, the rest are repeated items, and (3) all of the items are repeated items.

Case	Model	Type	MAP	WSR
All session	Pop_User	Original	0.6488	0.2407
	SVD_User		0.653	0.245
	Pop_User	RepProb	0.6649	0.2512
	SVD_User		**0.6669**	**0.2547**
Seen items only	Pop_User	Original	0.6947	0.2599
	SVD_User			
	Pop_User	RepProb	**0.7144**	**0.2712**
	SVD_User			

Table 5.3: MAP and WSR performance comparison of using empirical probability for repeated items consumption, indicated by RepProb, with the original model, indicated by Original, for case (2) and (3). The best performance for each case is written in bold.

Clearly, since there is no repeated item, we can not use the empirical probability for repeated items on case (1). The inclusion of the empirical probability of repeated item consumption can only be evaluated on case (2) and (3). As previously reported in Table 5.1, the best performing model is those which use the historical average playback ratio of seen items as the predicted score, i.e. SVD_User and Pop_User.

Table 5.3 shows the result of incorporating empirical probability into the model for case (2) and (3). For case (2), improvement can be seen for both the SVD_User and Pop_User where inclusion of the empirical probability factor into the model increases the MAP performance by 0.0139 and 0.0161 respectively compared to the original model. In terms of WSR, the difference is 0.0097 and 0.0105. These results are compared by using Wilcoxon signed ranked test and a statistical significance for all pairs of models is obtained with $p < 0.01$.

	Value of beta				
Fold	**0**	**0.3**	**0.5**	**0.7**	**1.0**
1	0.7045	0.7174	**0.7222**	0.7196	0.7079
2	0.6946	0.7098	**0.7144**	0.7116	0.7006
3	0.7056	0.7199	**0.7260**	0.7223	0.7126
4	0.7255	0.7422	**0.7472**	0.7450	0.7346

Table 5.4: MAP performance for case (3) across various values of β as indicated in Eq. 3.5. The SVD_User is used. Note that for case (3), SVD_User and Pop_User basically use the individual average playback ratio for each user as a score to rank items in a session. The optimal value of β is 0.5, which is written in bold.

Regarding the choice of the weight α and β as in the Eq. 3.5, it is found that using 0.5 as the weight for both of them produce the best performance, as indicated in Table 5.4. It is also quite interesting that by setting the α to zero (and β to one), meaning that we consider only the empirical probability of repeated consumption as the predicted final score, better performance is obtained compared to the original model when the value of β is zero, albeit worse than the model which consider both the original and empirical probability score.

5.2.1 Discussion

The finding presented earlier shows the potential of using the empirical probability of repeated item consumption as the score to rank the video in a session. We have shown that considering the average playback ratio for seen items is already a good predictor of future user preference. For unseen items, SVD performs better than using the global popularity of the videos, albeit with a small improvement. Combining them both yields the best performing model, and we can even improve the performance by considering the empirical probability of repeated items into the model.

The empirical probability of repeated items simply illustrates the relative frequency of events when the user consumes an item again, given past historical consumption sequence for that particular item by the user. Albeit its simplicity, it can practically be used to decide, among few music videos that have been played recently by the user, which one must be presented earlier. Nevertheless, we acknowledge that it is quite straightforward, because the empirical probability for each possible consumption sequences is calculated across all users, and does not take into account the potential that perhaps some groups of users and items have different repeated consumption patterns. In our case, regardless of the user and item, if the sequence of consumption towards an item is the same, e.g. user A consumes item X with the consumption sequence of [1,0,0] and user B consumes item Y also with the

same consumption sequence, then the empirical probability that A and B will consume the item X and Y again is the same. Considering the time constraint in this thesis, we leave it as future work on how to explore a more personalized empirical probability estimation for repeated item consumption.

5.2.2 Impact of session's consumption ratio on recommendation performance

In this section, we highlight the importance of considering different consumption ratios in the evaluation of our model. The consumption ratio of a session is defined as the ratio of items with a relevance value of 1 to the length of the session. Note that the previously reported results are calculated on all sessions with various length. In a session, a user can either stream zero or all music videos. We have excluded these cases in our evaluation, which basically translates to sessions containing zero relevant and irrelevant items respectively. It is meaningless to perform reordering on these sessions, especially if we consider the session as a binary list since any reordering will yield the same score in terms of the MAP.

Practically, for sessions with length N, all sessions having at least one until $N-1$ relevant items are considered in the evaluation. We notice that incremental performance across different models might vary depending on the ratio of relevant items in the list. While it does not affect the training of the model itself, plotting the performance across different consumption ratio might help to interpret how good the model is compared to each other.

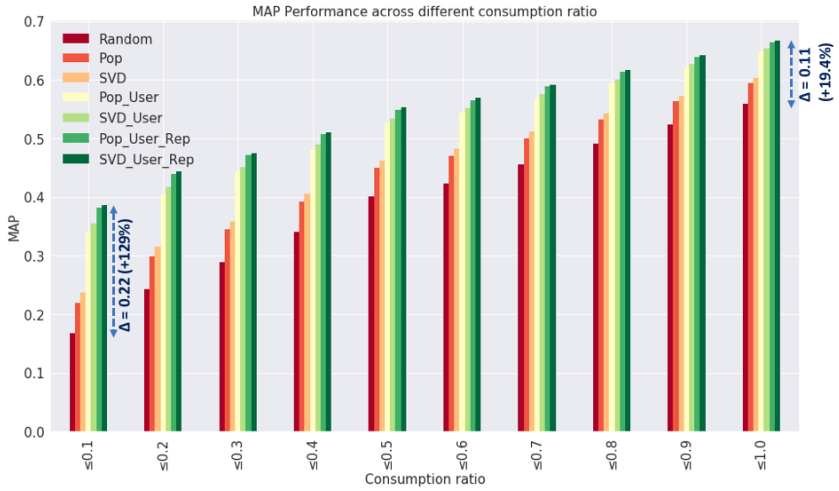


Figure 5.1: MAP performance across different consumption ratios. The arrows and numbers located at consumption ratio of 0.1 and 1.0 indicate the absolute and relative difference between the best model and random.

Fig. 5.1 shows the MAP performance of various models across different consumption ratios in our session reranking dataset. Note that each group of bars in the graph represents the performance measured on all sessions having at most a certain consumption ratio, in-

dicated in the x-axis. It is interesting to note that the incremental difference between the random and the best performing model decreases when considering sessions with higher consumption ratios. For sessions having consumption ratio less than 0.1, the incremental absolute difference between the SVD_User_Rep and random is 0.22 (+129% in relative), while the difference is only 0.11 (+19.4% in relative) for sessions having less than 1.0 of consumption ratio However, this finding is expected since the capability of the random model to correctly rank the items decreases a lot when it has to order sessions which only have at most 10% of relevant items. Imagine for example a session having ten videos containing only one relevant video. The impact of failing to rank the items higher will have severe performance in the MAP score, as opposed to sessions containing 8 or 9 relevant videos.

5.3 Incorporating Implicit and Explicit Feedback

In this section, we describe our experimental findings on the 'like' prediction task. The framing is completely different from what has been described in the session reranking experiments. We highlight the importance of exploiting different types of feedback and considering them in the training procedure of BPR algorithm. In particular, special attention is given to find the effects of using multi-level sampling strategy with BPR as described in Section 3.4.2. Finally, we highlight the importance of predicting user attention mode when interpreting implicit positive feedback preference. We set our experiments to answer RQ3: *Can we improve the performance of the recommender system by leveraging multiple types of user feedback?*

5.3.1 Multi-level Sampling Strategy with BPR

Recall that in multi-level sampling strategy, a sampling probability p_1 is used to determine from which levels the sampler should pick the positive and negative items candidates accordingly. In another perspective, this probability also indicates the probability threshold that the positive items are sampled from the items which have been liked by the user. The multi-level sampling strategy is illustrated in Fig. 3.5 at Section 3.4.2. In this set of experiments, the BPR baseline algorithm (BL-sampler) is compared with multi-level BPR (ML-sampler) across different number of sampling probability.

Fig. 5.2 illustrates the MAP performance as a function of iterations or epochs for different values of probability of sampling from liked items. Across all folds, the multi-level sampler with a ratio of 1.0 produces the best performance, except for the fourth fold where the best performance is achieved with a ratio of 0.7. A ratio of 1 means that the positive items are all taken from the items that have been liked by the user. The baseline sampler in general yield second lowest performance, only better compared to the sampler with a ratio of 0. A ratio of 0 means that the positive items are only sampled from the video which has a high average playback ratio.

Table 5.5 shows the absolute and incremental difference of performance across different probability of sampling from liked items. The gain is different for each fold. The highest incremental gain can be observed from the second fold where the ML-sampler strategy

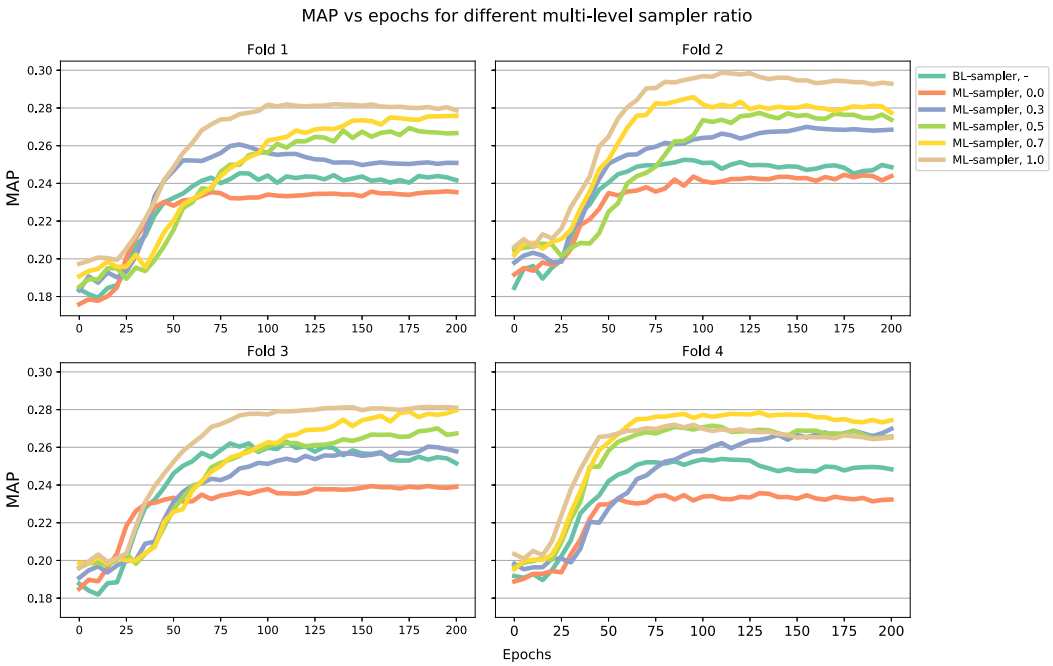


Figure 5.2: MAP performance per iteration for different values of sampling ratio for 'like' prediction task, performed on all items which have been seen by the user in the test-set, but have not been liked by the user in the training set.

increases the MAP by 18.29% relative, or 5% absolute difference, compared to the BPR baseline.

	Probability Sampling Ratio from Liked Items				
Fold	**0**	**0.3**	**0.5**	**0.7**	**1**
1	-0.01 (-3.9%)	0.02 (+6.23%)	0.02 (+9.79%)	0.03 (+12.41%)	**0.04 (+14.93%)**
2	-0.01 (-3.15%)	0.02 (+6.91%)	0.02 (+9.85%)	0.03 (+13.16%)	**0.05 (+18.29%)**
3	-0.02 (-8.95%)	0 (-0.99%)	0.01 (+2.68%)	0.02 (+6.33%)	**0.02 (+7.00%)**
4	-0.02 (-7.15%)	0.02 (+6.3%)	0.02 (+6.97%)	**0.02 (+9.71%)**	0.02 (+7.16%)

Table 5.5: Absolute and relative incremental performance (in parentheses) of the multi-level sampler strategy compared to the BPR baseline performance. The performance is taken from the maximum MAP among all epochs from Fig. 5.2. Best improvement is written in bold.

5.3.2 Multi-level Sampler Performance on Unseen Items

Streaming unseen items, which turns out to be enjoyable by the users, are more interesting both for the company and the users. For XITE, exposure to new contents will increase the coverage of items that it has. For the users, they can discover new great music videos that

are enjoyable to them. Thus, the power of the recommender system to provide a better prediction of videos which might be liked by the user from the items which have not been presented before is very crucial. To compare the performance of baseline BPR with multi-level BPR, the test set is modified so that only items which have not been presented to the user in the training set remain. We refer to this sets of items as 'unseen items' for the rest of our analysis. Fig. 5.3 shows the MAP performance of the multi-level sampling strategy tested on unseen items. In general, similar trends can be observed compared to Fig. 5.2.

Figure 5.3: MAP performance per iteration for different values of sampling ratio for 'like' prediction task, performed on all items which have been presented to the user during the test set period, but *have not been presented* to the user during the training period.

	Probability Sampling Ratio from Liked Items				
Fold	**0**	**0.3**	**0.5**	**0.7**	**1**
1	-0.03 (-7.61%)	0 (+0.67%)	0.02 (+5.07%)	0.04 (+9.54%)	**0.05 (+10.87%)**
2	-0.05 (-10.28%)	0.01 (+1.78%)	0.03 (+5.65%)	0.04 (+9.29%)	**0.06 (+12.89%)**
3	-0.05 (-12.17%)	-0.02 (-4.34%)	0 (-0.9%)	0.02 (+3.76%)	**0.03 (+7.44%)**
4	-0.05 (-12.06%)	0 (+0.52%)	0.02 (+4.86%)	**0.03 (+6.80%)**	0.02 (+5.00%)

Table 5.6: Absolute and relative incremental performance (in parentheses) of the multi-level sampler strategy compared to the BPR baseline performance for the unseen items recommendation. The performance is taken from the maximum MAP among all epochs from Fig. 5.2. Best improvement is written in bold

5.3.3 Discussion

We notice that the same items are sometimes presented multiple times to the user, and the fact that the user has watched a music video multiple times with a high playback ratio perhaps gives an implicit feedback that the user enjoy the item, and thus if this item is compared to any other items, it has a higher chance of being liked in the future. Based on the graph presented in Fig. 5.2 and 5.3, when the sampling ratio is greater than 0.5, in general, the performance of the ML-sampler BPR is better than the baseline. This trend is also verified in Table 5.5 and 5.6. However, the best performance is achieved when the sampling ratio is 1, meaning an explicit 'like' feedback is empirically shown to be a stronger signal of preference than items which have not been liked but have high average playback ratio.

Furthermore, from Fig. 5.2 and 5.3, it can be seen that in general, as the ratio is getting lower, the MAP is getting worse. While sampling positive items from the items with high average playback ratio seems to be productive to increase MAP, sampling none from the liked items, which means that the sampling probability is equal to zero, is not productive at all. Because we cannot generalize that a ratio of 1 always produces the best performance, the value of the sampling probability needs to be optimized for different dataset and period.

Finally, the result that we have obtained in this section can be used to answer RQ3 that: yes, by leveraging multiple types of feedback available in the curation-based personalized music video televisions, we can improve the recommender system performance on the 'like' prediction task.

5.4 Incorporating User Attention Prediction

The experimental results described in this section is related to answer RQ4: *Can we improve the performance of the recommender system by considering predicted user attention in a session?* Based on the empirical findings in Section 5.3, we observe that although in general, using sampling probabillity of 1 produces the best performance for the multi-level BPR model, the results with sampling probability between 0.5 and 1 are comparably competitive. Note that each consumption by the user affects the calculation of the average playback ratio and an illustration has been given in Section 3.5.1 to motivate the need to infer the user attention to obtain a 'better' representation of the average playback ratio of an item. Furthermore, it has also been described in the same section that 'like' prediction experiment performed by using sampling probability of *zero* is used to show the effects of using user attention prediction in the ML-sampler BPR.

Fig. 5.4 shows the performance for the 'like' prediction task using various play cut-off threshold. In general, across all folds, user attention prediction improves the performance compared to the baseline. However, the gain is different among different folds. Fig. 5.5 illustrates the performance evaluated only on unseen items in the test set. In general, the result is consistent with Fig. 5.4. Based on these findings, we observe that performing simple heuristic already filter out a lot of items which do not need to be considered as implicit positive feedback. Thus, the number of items which has high playback ratio is reduced and smaller than if no cut-off is applied. We think that this reduction affects the

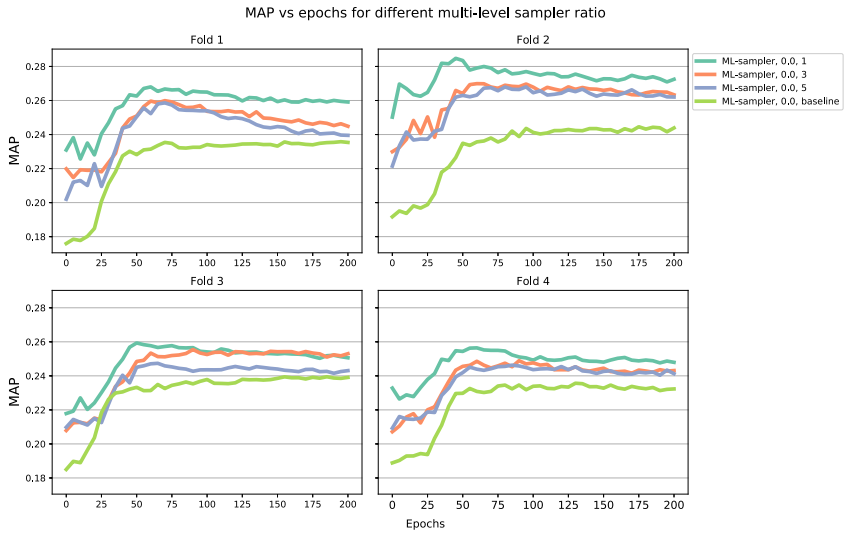


Figure 5.4: MAP performance considering different play cut-off threshold for the 'like' prediction task using multi-level sampler where the positive items are taken from items with average playback ratio more than 0.5. The evaluation is performed on all items that have not been liked yet by the user.

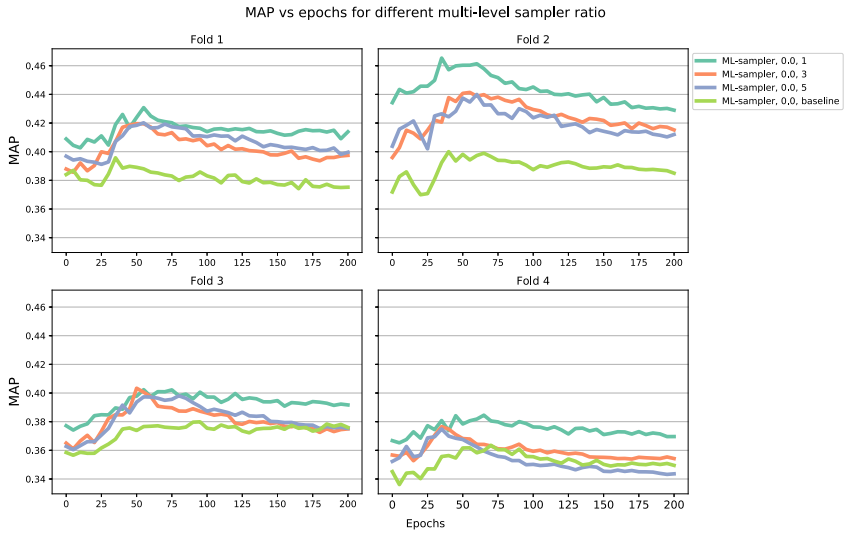


Figure 5.5: MAP performance considering different play cut-off threshold for the 'like' prediction task using multi-level sampler where the positive items are taken from items with average playback ratio more than 0.5. The evaluation is performed on all *unseen* items that have not been liked yet by the user.

implicit user preference which are inferred from the average playback ratio, and it leads to an overall higher performance to predict future 'liked' items. Based on Fig. 5.5, it can also

be observed that after several epochs, the performance degrades for the model using the play cut-off threshold, which is a signal of overfitting.

5.4.1 Combining the multi-level sampler with user attention prediction

Based on the experimental results presented in the previous section, it is quite clear that inferring user attention by only considering, as positive implicit feedback, few music videos completely consumed right after an active action is performed by the user, the performance of recommender systems based on ML-sampler BPR can be improved.

In the following experiments, further evaluation is performed by combining both the multi-level sampling strategy as performed in Section 5.3 with the user attention prediction. In the previous experiment, we limit the experiment only on the scenario where positive items are sampled from the collection of items having average playback ratio greater than 0.5. This is merely done to show the benefit of inferring user attention by eliminating the effect of using the 'liked' items on the performance. Due to the positive effect of inferring user attention that we have obtained so far, it is then natural to see if it also brings benefit when the sampling probability is not only set to zero.

We report the performance for various values of play cut-off threshold in Fig. 5.6 and 5.7. Note that in both of these figures, the baseline is set to the best performing model described in Section 5.3.1 and 5.3.2, which uses the ML-sampler BPR with no user attention prediction. Furthermore, for each use of play cut-off threshold, we report the best use of sampling probability threshold for each fold, which are drawn in the figures.

Based on Fig. 5.6 and 5.7, it is interesting to see two contrasting results from the perspective of evaluating the performance on all and only on unseen items. We discuss the interpretation of the results in the next section.

5.4.2 Discussion

Based on the experiments on considering implicit and explicit feedback reported in Section 5.3, as well as user attention prediction in this section, our general finding is that the best performance is achieved by taking into account both multiple types of feedback and the user attention prediction. However, we notice that the generalizability of using user attention prediction is not as clear as the use of multiple types of feedback, and we describe some discussion points next.

We note that the optimal numbers of sampling probability from liked items, as well as play cut-off threshold, are different between each fold. While in general, using higher number of sampling probability produces better performance, general trend cannot be deduced for the values of play cut-off threshold. What we can conclude is that the inclusion of play cut-off threshold successfully achieves statistical significant result when compared to the baseline in Fig. 5.4 and 5.5 when the sampling probability is set to zero. However, the incremental performance is very small when higher sampling probability is used as depicted in Fig. 5.6, and even does not affect the performance on predicting unseen items on 5.7.

Note that the user attention prediction might change the baseline value of average playback ratio on each item for each user. Consequently, the composition of items categorized

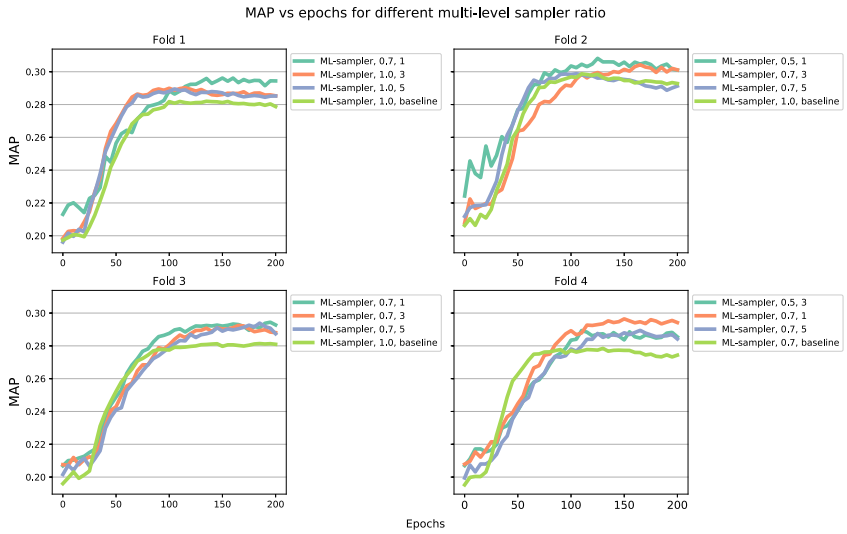


Figure 5.6: MAP performance considering different play cut-off threshold for the 'like' prediction task. The evaluation is performed on all items that have not been liked yet by the user.

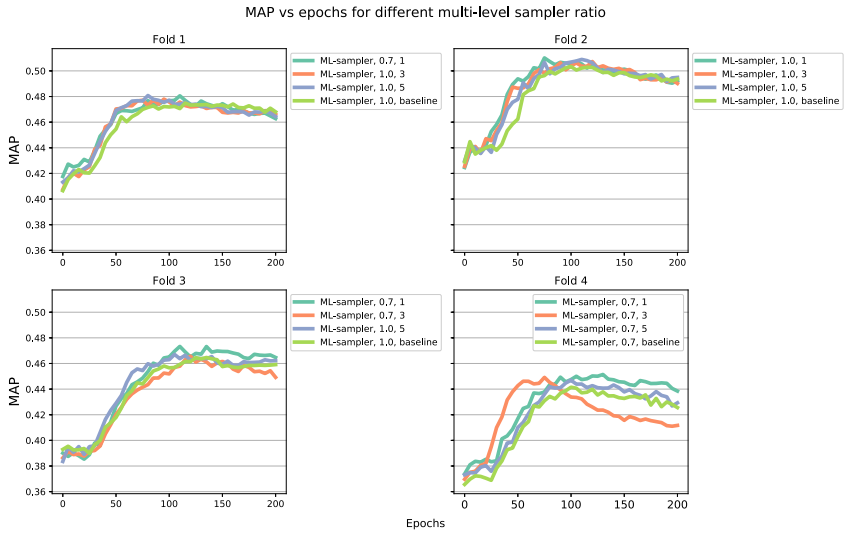


Figure 5.7: MAP performance considering different play cut-off threshold for the 'like' prediction task. The evaluation is performed on all *unseen* items that have not been liked yet by the user.

into level L_2 and L_3 (referring to the definition of level described in Section 3.4.2) might also change. We think that the user attention prediction helps to identify *better* approximation of user preference for items which have not been liked yet by the user, but have been presented in the training period. This is the reason why there is a noticeable increase in the scenario

where we only use the level L_2 and L_3 to sample items. However, when higher sampling probability is used, the effect of sampling positive items from 'liked' items is stronger to indicate preference than inferring implicitly from items with high average playback ratio, which explains why using the play cut-off threshold does not really affect the performance compared to the baseline. While our best performing model is achieved when both multi-level sampling strategy and user attention prediction are used, we conclude the incremental benefit of using the user attention prediction is quite small.

As a final note in this section, based on our experiments, we can already answer RQ4 that: yes, the user attention prediction helps to improve the overall performance of ML-sampler BPR model on the 'like' prediction task. The simple heuristics used in our user attention prediction show promising direction on inferring which videos that have a high probability to be *actually* watched by the user to improve recommender systems model. We pointed out that there is little research on this topic, and based on our results, using a simple cut-off heuristic for long continuous played events already produces a reasonable improvement. Nevertheless, we acknowledge that this approach is quite rudimentary. Another promising direction for the future work is to personalize the play cut-off threshold for different users by finding the pattern on the average length of continuous uninterrupted session from each user.

5.5 Recommendation List Analysis

This section highlights some exploratory studies on the quality of the recommendation list produced by using multiple types of feedback and user attention prediction. It does not directly answer any of our research questions, but rather serves to report our exploratory thinking on how the recommendation process can be implemented online. Previously, we focused on the numerical evaluation aspects based on the defined evaluation metrics. Due to the limited capability of users to select items, we can only evaluate with respect to the videos that the users have been exposed to. This brings limited utility since our evaluation is biased towards the music videos that the internal system at XITE has presented to the users.

Eventually, based on the historical interactions from the user, the recommender system will be used to provide an arbitrary recommendation from all the items candidates. Hypothetically, we imagine a scenario where a channel is dedicated to stream contents based on the videos that has been liked by the user. Note that this hypothetical use case is proposed merely to show the efficacy of our recommendation list analysis presented in this section, and by no means reflects any on-going or future product development at XITE.

Based on the internal discussion with the XITE team, there are two important requirements that need to be addressed when providing on-the-fly recommendation, which is purely based on the list of items produced by the algorithms.

1. The recommendation result should not be biased too much towards popular items. The recommendation process serves as a way to present great contents and let the user 'discover' interesting music videos, which are not necessarily popular.

2. The recommendation needs to maintain some level of coherence with the user preference, which can be inferred from their past behaviors, for example, based on their previous 'likes'.

With respect to these requirements, we try to find different methods to check and verify the quality of the recommendation lists. We propose to use two metrics namely the popularity aspects and genre incoherence, which are described in the following subsections. Based on the best performing configuration for a model with multi-level sampling strategy (ML-BPR), the recommendation lists are compared to the original BPR model.

Fold	Model	Like Probability	Play Cut-off
1	BPR	-	-
	ML-BPR	0.7	1
2	BPR	-	-
	ML-BPR	0.5	1
3	BPR	-	-
	ML-BPR	0.7	1
4	BPR	-	-
	ML-BPR	0.7	1

Table 5.7: BPR configuration of two models: ML-BPR and original BPR to evaluate popularity and genre incoherence aspect

Model Configuration

Considering the different sampling probability threshold from liked items, as well as the play cut-off threshold, for each fold, the configuration which produces the best performing models for ML-BPR are used. Table 5.7 shows the model's configuration.

5.5.1 Popularity Aspects

One of the important objective in music recommender systems is to promote great contents from the long-tail, meaning that not only popular music videos, which are commonly measured by publicly available charts or billboards, that are presented but also videos that the curator thinks are great but not so popular according to the charts. While there are various external sources to measure the popularity of a video, we can also measure the popularity aspect of videos internally by counting the number of 'like' that the videos received from the users. However, careful attention needs to be given to interpret the number of 'like', because it might be biased towards how many times that the internal system at XITE *decides* to stream the video to the user.

Considering the biased condition, we introduce a metric called *normalized popularity*, which are calculated by dividing the number of 'like' that a video receives with the total number of times that the video is presented to the users uniquely. Due to the normalization,

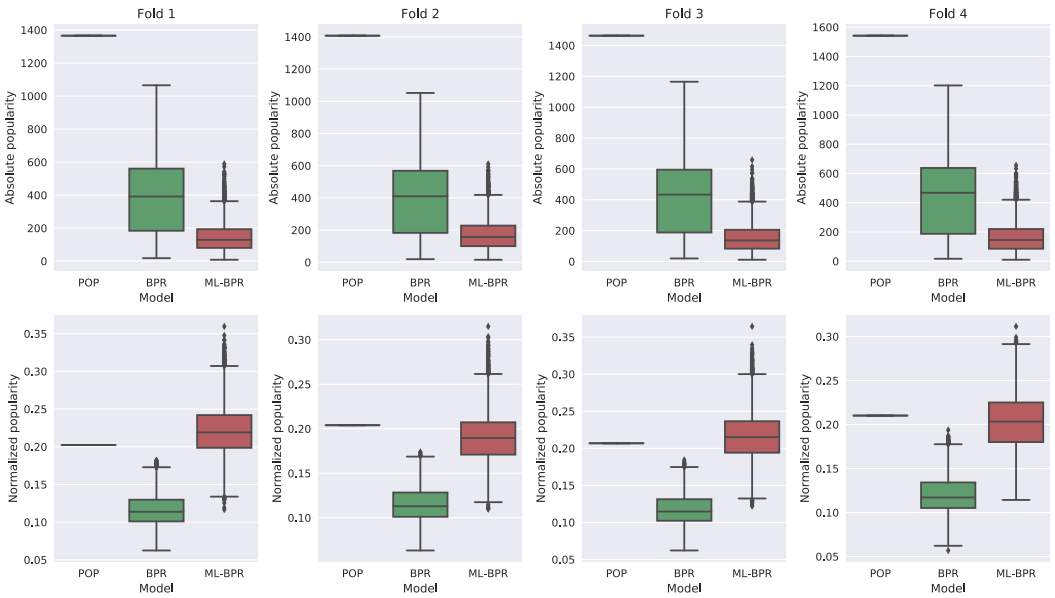


Figure 5.8: Percentile distribution of popularity metrics over different models and folds

the normalized popularity ranges between 0 and 1. On the other hand, the *absolute popularity* of a music video is determined simply by calculating the number of 'like' that it receives from the users uniquely.

After training the models, we generate top 50 recommendation for each user. For each item in the recommendation list, two metrics are measured: absolute and normalized popularity. Furthermore, both of these metrics are averaged for each user. The percentile distribution of these metrics is then plotted as in Fig. 5.8.

Based on Fig. 5.8, the mean absolute popularity for BPR model is higher than ML-BPR. On the contrary, the mean normalized popularity for ML-BPR is higher. This trend is consistent across different folds. This finding matches well with the first requirements mentioned in Section 5.5 which state that the recommender should be able to provide items from less popular ones. Note that the notion of popularity can be understood from both the number of 'like' as well as the number of times the items are streamed to the user.

The low number of absolute popularity for ML-BPR with higher normalized popularity means that the total number of times that the videos are streamed is also lower compared to the BPR model. It is interesting to note that ML-BPR not only performs remarkably better than BPR in terms of MAP, as extensively evaluated in Section 5.3, but also capable of providing items which are less popular but still appreciated by the user, indicated by higher normalized popularity.

Pre-filtering the Item Candidates

In our previous setup, the item candidates basically are not filtered, meaning that even items which have very low popularity scores are also considered. From the business perspective,

it might be risky to recommend such videos because maybe the user is not familiar with them, and the videos are only enjoyed by very specific users.

To minimize the effect of recommending items with low popularity, it is common practice to perform pre-filtering on item candidates for recommender systems. We perform further analysis by varying the number of minimum 'like' that the videos have in the item candidates to be scored by the model.

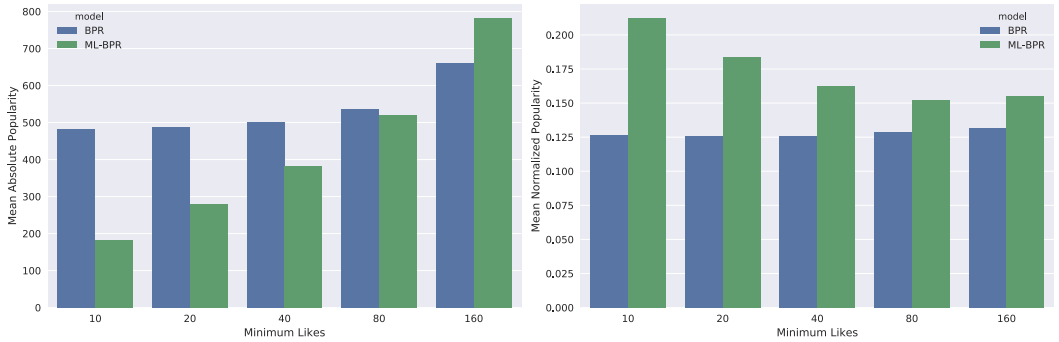


Figure 5.9: Mean absolute and normalized popularity for various number of minimum 'like' across two models : BPR and ML-BPR. The analysis is performed on the first fold of our dataset.

Fig. 5.9 shows the mean absolute and normalized popularity across a different number of minimum 'like' for the first fold of our dataset. It can be seen that as the number of minimum 'like' increases, ML-BPR recommends items with higher absolute but lower normalized popularity. On the other hand, the value of BPR is flat, except in the last part where the minimum number of 'like' is quite high. The flat trend for BPR reflects that actually the top recommended list mostly originates from very popular items. In general, across different values of minimum 'like', ML-BPR is able to recommend items with smaller absolute but higher normalized popularity than BPR, the finding which is also confirmed in our previous analysis.

5.5.2 Genre Incoherence

Considering the second requirement as mentioned in Section 5.5, it is important to avoid recommendation that deviates too much from the user's historical preference, which might eventually surprise the user. To calculate how far the recommendation deviates, we need a measure which captures the similarity between the user's taste and the recommended list. In the music domain, one taxonomy that is often used to capture similarity between songs is the genre.

The genre incoherence of the recommendation lists is evaluated as follows. Let $G = (g_1, g_2, ..., g_N)$ defines the genres that are of interest in our dataset. Based on the previously liked videos, we identify the ratio L_u from each genre liked by user u such that for each users, we have $L_u = (l_{g_1}, l_{g_2}, ..., l_{g_N})$. Note that each element in L_u will sum up to one. Furthermore, after training the model, the model will produce recommendations lists for all

users. The Top-N recommendation list is mapped into the associated genres, and the ratio of each genre is calculated, so that we have the ratios of genres in the recommended list as $R_u = (r_{g_1}, r_{g_2}, ..., r_{g_N})$. We are interested to evaluate for each genre, how different is the proportion of the genres in the recommended and previously liked lists. We define the genre incoherence C across all users as depicted in Eq. 5.1. To the best of our knowledge, we are the first to use the formulation of this genre incoherence.

$$\frac{1}{|U|}\sum_{u\in U}\text{dist}(L_u, R_u) = \frac{1}{|U|}\sum_{u\in U}\sum_{i=1}^{N}|L_u^{g_i} - R_u^{g_i}| \tag{5.1}$$

Basically, any distance measure can be used. Common distances or similarity metrics used are Euclidean, Manhattan, or cosine similarity. Manhattan distance of two vectors is calculated based on the absolute difference of each element in the vectors. The absolute difference makes the result of the Manhattan distance more interpretable than Euclidean or cosine similarity, which takes the square and dot product of the vectors respectively.

For example, consider that we have 100 users and we need to evaluate the genre incoherence of the recommendation lists for all these users using two model, model A and model B. Let the genre incoherence calculated for five genres, using the Manhattan distance. In the end, model A and B have on average genre incoherence of 0.1 and 0.2 respectively. We can interpret this finding by saying that the Top-N recommended list produced by model A on average deviate by 0.02 (or 2%) compared to the previously liked genre proportion for each user. Note that 0.02 is obtained by dividing 0.1 with five genres. Considering this interpretation, we can also say that model B on average produce recommended list which also deviates 2% higher for each genre compared to model A. The definition of absolute difference in Manhattan distance calculation enables us to directly compare the difference of two values, without complicating it with another transformation.

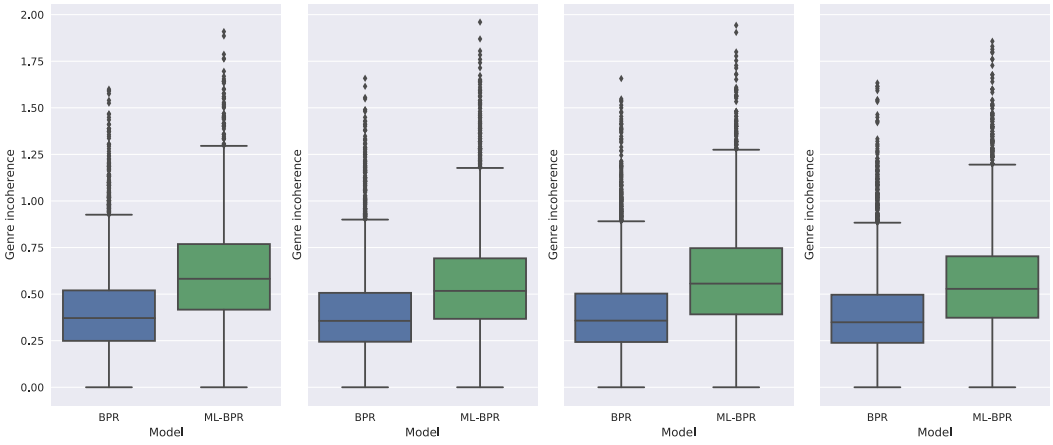


Figure 5.10: Percentile distribution of popularity metrics over different models and folds

Fig. 5.10 illustrates the genre incoherence across different folds between the two models: BPR and ML-BPR. The result is generated across all users in the dataset with a minimum likes of 10 items. The top 50 items which are recommended by both algorithms are

picked and analyzed. In general, ML-BPR produces recommendation with more varying genre compared to the user's liked genre. Note that during the training of the model, both BPR and ML-BPR do not actually 'aware' of the genre of the videos. The utility of using the genre incoherence might be useful to explain the diversity of the recommended list in terms of the genre to the music team at XITE.

Chapter 6

Conclusion and Future Work

This chapter first summarizes the conclusions that can be drawn based on the experimental results conducted in this thesis. We also propose some recommendations which might be useful for XITE. Finally, we close off this thesis by highlighting some potential future works which can be continued upon the findings in this thesis.

6.1 Conclusion

In this section, we answer the research questions identified at the beginning of this thesis by reflecting on the empirical findings that we have found in our experiments.

> **RQ1** *Can we successfully rerank sessions in an offline experiment for recommender systems evaluation for curation-based music video television?*

In general, we can answer that: yes, by exploiting playback ratio as feature, and using SVD algorithm combined with the individual average playback ratio of each user for each item, we are able to perform session reranking which is better than the random baseline.

We have laid out some of the challenges and thinking on the design of offline evaluation framework for music video television for the session reranking task. Our proposed framework described in Chapter 3.2.4 highlights the steps needed to evaluate the curated recommendation list. We exploit the fact that the user gives feedback in a session, which is represented as binary feedback of either 0 or 1. A session is then basically transformed into a list of binary relevances, which we can evaluate by using various metrics such as MAP and our new proposed metric: Weighted Streak Recall (WSR).

Based on the proposed evaluation framework, we test out empirically the performance of different recommender systems algorithms, namely: random, popularity, and matrix factorization with SVD. Since our work focuses on validation of our proposed evaluation framework, we do not test more algorithms. However, it comes as a natural extension of our work to extensively test more state-of-the-art recommender systems algorithms under the proposed evaluation framework.

Based on the empirical findings, the evaluation framework can therefore distinguish the performance of different algorithms, and the performance of models using popularity

and SVD algorithm are better than the random baseline. Statistical significance is achieved between the best performing models with the random baseline, highlighting the potential of using the evaluation framework to test out further algorithms more extensively.

> **RQ2** *Can we improve the recommender system performance on session containing items which have been played before by the user by considering the sequence of repeated item consumption behavior?*

Based on our experimental findings, we can answer this research question that: yes, we found that there is a slight improvement by exploiting sequential actions of repeated items by combining empirical probability of consuming repeated item again into the original model which exploit the average playback percentage as a rating. Nevertheless, the incremental difference is very small, and more investigations are needed to verify the usefulness of using such approach.

The same music videos might be presented multiple times to the user in the PMT. The user can decide to watch a video completely or skip it after certain duration. Based on the skip behavior, we can calculate playback percentage of each music video. Naturally, when a user watches a complete video, the playback percentage reaches a maximum score of 1 (or 100%). Based on the playback percentage, for each video, a preference score can be calculated by taking the average across all events experienced by the user on that particular video. We perform experiments based on score aggregation in Section 5.1.

Inspired by previous work in [2], we empirically check whether there is a certain pattern on the repeated item consumption probability when considering sequences of previous actions. We found that indeed different sequences produce a different empirical probability of consuming the item if it is presented again. For example, one of the interesting insight that we found is that the more recent the user give positive appreciation towards a video, the more likely that if the video is presented again, it will be appreciated by the user.

We derive a mapping of binary sequences of previous actions into an empirical probability of consuming the item again based on our training data. Using a simple linear combination, we combine the score predicted by the original model which use the average playback percentage as the rating with the empirical probability of consuming repeated item again.

> **RQ3** *Can we improve the performance of the recommender system by leveraging multiple types of user feedback?*

We conclude by answering that: different types of user feedback can be exploited along with the BPR framework by using multi-level sampling strategy to improve the performance of the recommender systems for the 'like' prediction task. By using positive implicit feedback to infer explicit feedback and sampling negative items not randomly from all items but from the items that the user have been presented but have not been liked yet, the framework is able to learn user preference better.

We perform experiments utilizing all the available types of feedback that the PMT has, which can be categorized into explicit and implicit feedback. A 'like' is considered as

explicit while a 'skip' and complete consumption are categorized as implicit feedback. To test the effects of using different combinations of this types of feedback, we have performed a 'like' prediction recommendation task, which tries to optimize the rank of the items in the test set so that the liked items are ranked earlier.

We laid out the experimental results to answer the research questions in Section 5.3. BPR framework is used as the baseline recommender systems algorithm. We also leverage the implicit data as auxiliary feedback. With the addition of auxiliary feedback, we use multi-level sampling strategy in BPR which has been used in some other domains as well [30].

We found that using different types of feedback gives benefit to the performance of the original BPR model. In particular, the optimal configuration which led to the best performance in our experiments is by sampling positive items from the liked music videos, and negative items from the videos that have been presented to the user, but have not been liked yet. We also notice increased performance when considering sampling some parts of the positive items (with a certain probability threshold) from items which have high average playback ratio. This shows the latent benefit of implicitly assume that the video with high average playback ratio is perhaps implicitly liked by the user.

> **RQ4** *Can we improve the performance of the recommender system by considering predicted user attention in a session?*

We answer RQ4 that: yes, we can, in general, improve the performance of the recommender system by considering predicted user attention in a session containing long continuous completely played videos. However, we highlight that our method is naive and while the heuristics work, the optimal configuration might need to be found for a different dataset and use case.

We build upon experiments conducted to answer RQ3 to show the utility of inferring user attention on the performance of the BPR with multi-level sampling framework. A simple heuristic is used by applying play cut-off on continuous played event in a training session. By considering the play cut-off, only few music videos completely played after an action is taken, as defined by a number N, would be considered as a completely watched event. Otherwise, the items are discarded.

The heuristics prediction by itself cannot be evaluated since there is no ground truth. However, we apply our end-to-end evaluation framework with and without the user attention prediction. We found that inferring user attention can slightly improve the performance of BPR with multi-level sampling strategy.

6.2 Recommendations

Based on the experimental results and analysis performed in this thesis, some recommendations are proposed to XITE, which we hope might be useful to be considered in the future recommender system development at the company.

1. We have shown that considering each user's historical feedback on items and training a matrix factorization-based model helps to achieve good performance in our offline experiments. In particular, we show the usefulness of considering average playback ratio as features for the recommender system model in our session reranking task, which is described in Section 5.1. Thus, naturally, we recommend to test this strategy online by using A/B testing to verify the benefit of implementing this method to real users.

   Once the online evaluations have been performed, we think that the actual benefit of developing offline evaluation framework will be more clear. It might be the case that the online-offline metrics correlates well, which is good since we can further improve the recommender algorithms based on the proposed evaluation framework. If it is not the case, then further study needs to be performed to improve the offline evaluation framework.

2. It is more useful to prioritize 'like' feedback over other forms of implicit feedback to indicate positive user preference towards items. We have shown in our 'like' prediction experiment that in general, by using multi-level sampler BPR framework, the best performance is achieved when sampling positive items from already 'liked' videos. This is in combination with sampling negative item from the items which have been presented but have not been liked yet by the user.

3. For even better performance, we recommend to also take the user attention prediction into account when building the recommender systems. Thus, we think that it is worth verifying as well via online A/B testing the recommendation model which exploit a combination of 'like', 'skip', and completely consumed videos, with the play cut-off threshold.

6.3 Future work

We have identified some potential future works that might be performed based upon the empirical findings in this thesis.

1. We acknowledge the lack of extensive evaluations of recommender systems algorithms in our experiments. It is however intended because, given the time constraint, we focus on the evaluation framework rather than rigorous experiments comparing different algorithms. In session reranker scenario, more algorithms can be used to compare performance across different models, for example, factorization machines, or even algorithms from the family of learning-to-rank methods such as BPR.

2. BPR is only one of the available recommender systems framework inspired by the learning-to-rank method. Considering the multi-level sampling strategy employed in our study, the same approach can be used in another family of algorithms utilizing pairwise comparison of positive and negative items. During initial work performed in this thesis, we modified the sample selection of the LightFM [1] when using WARP

[1] https://github.com/lyst/lightfm

loss to consider multi-level sampling scenario. However, we decided not to continue the path of comparing different algorithms as we focus more on the validation and analysis of the BPR framework. However, for future work, it would be interesting to apply empirical comparison across different algorithms since the idea of multi-level sampling strategy that we use can be easily applied to other algorithms by changing the sample selection method.

3. Furthermore, when considering repeated item consumption probability, our approach is quite rudimentary, in a sense that although we consider personalized sequence of items consumed by each user, the empirical probability that is assigned to each repeated item in the test set is only dependent upon the sequence of past action given on the item, but independent of the user and item types. It means that for different users and items, if the sequence of previous actions on that item is the same, then the empirical probability of consuming the item again is also the same. More research looking at personalizing the empirical probability for users or items might be interesting to capture user preference better.

4. A user study might help to validate the preference of recommendation lists which are produced by BPR and ML-BPR from the music team at XITE. We have highlighted two important characteristics from both algorithms, the popularity aspect and genre incoherence. It would be interesting to check whether user perception towards the lists is in line with those characteristics and whether they could distinguish the differences in the characteristics of the recommended lists. Eventually, it is also important to confirm whether the recommendation lists provided by the multi-level sampling strategy are preferred by the music team.

5. Last but not least, an online-offline study needs to be performed, since the actual performance of any recommender systems should be measured to real users.

Bibliography

[1] Charu C Aggarwal et al. *Recommender systems*. Springer, 2016.

[2] Ashton Anderson, Ravi Kumar, Andrew Tomkins, and Sergei Vassilvitskii. The dynamics of repeat consumption. *Proceedings of the 23rd international conference on World wide web - WWW '14*, pages 419–430, 2014.

[3] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992.

[4] Robin Burke. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[5] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[6] Jun Chen, Chaokun Wang, Jianmin Wang, and Philip S. Yu. Recommendation for Repeat Consumption from User Implicit Feedback. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):3083–3097, 2016.

[7] Charles L.A. Clarke, Maheedhar Kolla, Gordon V. Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '08*, page 659, New York, New York, USA, 2008. ACM Press.

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, (September):39, 2010.

[9] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[10] Nan Du, Yichen Wang, Niao He, and Le Song. Time-Sensitive Recommendation From Recurrent User Activities. *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 1:1–11, 2015.

[11] Michael D. Ekstrand, F. Maxwell Harper, Martijn C. Willemsen, and Joseph A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 161–168, New York, NY, USA, 2014. ACM.

[12] Ericsson Consumer Lab. Tv and media 2017. Technical report, October 2017.

[13] Carlos A. Gomez-Uribe and Neil Hunt. The Netflix Recommender System. *ACM Transactions on Management Information Systems*, 6(4):1–19, 2015.

[14] Dhruv Gupta, Mark Digiovanni, Hiro Narita, and Ken Goldberg. Jester 2.0 (poster abstract): evaluation of an new linear time collaborative filtering algorithm. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–292. ACM, 1999.

[15] F. Maxwell Harper and Joseph A. Konstan. The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems*, 5(4):1–19, dec 2015.

[16] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[17] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, oct 2002.

[18] Gawesh Jawaheer, Martin Szomszor, and Patty Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems - HetRec '10*, pages 47–51, 2010.

[19] Christopher C Johnson. Logistic Matrix Factorization for Implicit Feedback Data. *NIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations*, pages 1–9, 2014.

[20] Ahmed Kachkach. Analyzing user behavior and sentiment in music streaming services. Master's thesis, Kungliga Tekniska Hgskolan, 2016.

[21] Iman Kamehkhosh, Dietmar Jannach, and Lukas Lerche. Personalized next-track music recommendation with multi-dimensional long-term preference signals. In *Proceedings of the 1st Workshop on Multi-dimensional Information Fusion for User Modeling and Personalization (IFUP 2016)*, 2016.

[22] Komal Kapoor, Karthik Subbian, Jaideep Srivastava, and Paul Schrater. Just in time recommendations: Modeling the dynamics of boredom in activity streams. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 233–242, New York, NY, USA, 2015. ACM.

[23] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[24] Suzana Kordumova, Ivana Kostadinovska, Mauro Barbieri, Verus Pronk, and Jan Korst. Personalized implicit learning in a music recommender system. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6075 LNCS:351–362, 2010.

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

[26] Santiago Larrain, Christoph Trattner, Denis Parra, Eduardo Graells-Garrido, and Kjetil Nørvåg. Good Times Bad Times. In *Proceedings of the 9th ACM Conference on Recommender Systems - RecSys '15*, pages 269–272, New York, New York, USA, 2015. ACM Press.

[27] Martha Larson, Alessandro Zito, Babak Loni, and Paolo Cremonesi. Towards minimal necessary data: The case for analyzing training data requirements of recommender algorithms. In *FATREC Workshop on Responsible Recommendation Proceedings*, 2017.

[28] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 210–217, New York, NY, USA, 2010. ACM.

[29] Lukas Lerche and Dietmar Jannach. Using graded implicit feedback for bayesian personalized ranking. *RecSys 2014: Proceedings of the 8th ACM conference on Recommender systems*, pages 353–356, 2014.

[30] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. Bayesian Personalized Ranking with Multi-Channel User Feedback. *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys '16*, pages 361–364, 2016.

[31] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: A survey. *Decision Support Systems*, 74:12–32, jun 2015.

[32] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences*, 453:80–98, jul 2018.

[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[34] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[35] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*. Springer US, Boston, MA, 2011.

[36] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[37] 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, New York, NY, USA, 2001. ACM.

[38] Markus Schedl. The LFM-1b Dataset for Music Retrieval and Recommendation. *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval - ICMR '16*, pages 103–110, 2016.

[39] Markus Schedl, Peter Knees, Brian McFee, Dmitry Bogdanov, and Marius Kaminskas. *Music Recommender Systems*, pages 453–492. Springer US, Boston, MA, 2015.

[40] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *CoRR*, abs/1710.03208, 2017.

[41] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 30music listening and playlists dataset. In *ACM RecSys Conference Posters 2015*, 2015.

[42] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 109–116, New York, NY, USA, 2011. ACM.

[43] Douglas Véras, Thiago Prota, Alysson Bispo, Ricardo Prudêncio, and Carlos Ferraz. A literature review of recommender systems in the television domain. *Expert Syst. Appl.*, 42(22):9046–9076, December 2015.

[44] Sooyeon Yoo and Kyogu Lee. A Data-driven Approach to Identifying Music Listener Groups Based on Users' Playrate Distributions of Listening Events. *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 77–81, 2017.

[45] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.

Appendix B

Paper Contribution

Based on some part of this thesis, a paper was submitted and accepted as a poster at ACM RecSys Workshop on Offline Evaluation for Recommender Systems (REVEAL 2018) [1]. We attached the paper in our report for reference.

[1] https://sites.google.com/view/reveal2018/posters?authuser=0

Towards an Offline Evaluation Strategy for Recommender Systems in Personalized Music Video Television

Reza Aditya Permadi*
Delft University of Technology
Delft, The Netherlands
r.a.permadi@student.tudelft.nl

Bouke Huurnink
XITE Networks International
Amsterdam, The Netherlands
bouke@xite.com

Martha Larson
Radboud University & Delft University of Technology
Nijmegen/Delft, The Netherlands
m.larson@cs.ru.nl

ABSTRACT

This paper investigates an industry use case involving an online music video television service for which an offline evaluation framework is required. We present an evaluation framework designed to assess the effectiveness of recommender algorithms that present the user with a continuous stream of items, where the items are selected from a curated subset of items. Our proposal is motivated by an industrial use case in the area of personalized music television. This use case provides us with technical specifications and with user requirements on which we base our design for an offline evaluation framework, and our proposal for new evaluation metrics. We discuss the connection between the task of reordering a given set of items to create a personalized stream, and the conventional task of reranking items, e.g., in top-N recommendation. Our main contribution is a description of the challenges facing recommender systems for music video television and a proposal for offline evaluation of curation-based systems that attempt to address these challenges. Our work opens the path to further research, which will involve validating and extending our proposals.

1 INTRODUCTION

Offline evaluation can be a cost effective way of testing various recommender system algorithms as compared to implementing user studies or performing online evaluation with actual users [8]. However, it brings certain challenges, for example: how to choose the evaluation metrics and how to frame the recommendation problem itself. Both of these questions need to be answered by carefully identifying what makes a recommender system achieve success in its overarching goals to increase user satisfaction and bring more value to the business.

Recently, recommender systems have included curated elements to provide high quality, personalized content to users. Specifically, companies in the music domain like Pandora, Spotify, and Apple have incorporated teams of music experts to choose and curate music playlists for their users. In curation-based systems, the items presented to the user originate from pools of items that are preselected by the curators.

Online streaming music services offer personalized music content to users and are often accessed via personal devices such as mobile phones and laptops. In this paper, we focus on recommendation for the television screen. Televisions are wide-spread in our living rooms, and have become quite sophisticated with their 'smart' capabilities. Music recommender system research has gained a lot of attention during the past years [7], yet the focus is on audio streams and there has been little research in the music *video* domain. Moreover, in a literature study conducted in [9], most investigations of recommender systems for television perform research for TV program recommendation.

One clear difference between TV programs and music videos is the length of the content. Duration for music videos typically is quite short and falls between 3-5 minutes, while TV programs generally span half an hour or more. This difference affects the behavior of the user when interacting with TV programs and music videos. For example, if we assume a priori that a user wants to spend the next 30 minutes watching television, one good recommendation of a TV program that is selected by the user perhaps is enough to keep them engaged, while for music videos, we need to provide more items due to the short duration of the videos. Thus, choosing the right items to recommend might be more challenging for music videos television.

The context of this paper is a larger project that is aimed at implementing a personalized curation-based recommender system in the music video television domain. The paper makes two specific contributions: First, we describe an industry use case for which an offline evaluation framework for a personalized stream of multimedia content is needed. The streams are structured into channels with different music-related themes. Each channel is based on a curator-defined list of items and then personalized using a recommendation system. The user interaction with a channel falls naturally into sessions, and the task of the recommender is to recommend a sequence of items for the session. The description details the challenges faced by the recommender systems, as well as technical and user requirements for the evaluation. Second, on the basis of the industry use case, we propose an evaluation framework that takes the nature of the available offline data into account. We also propose new offline evaluation metric called Weighted Streak Recall (WSR) that is informed by user requirements. Specifically, users must see relevant content as soon as possible upon entering the stream, and also must enjoy consecutive relevant items.

The paper is structured as follows. In Section 2, we provide selected examples of related work from areas relevant to personalized music video television. Then, Section 3 describes our industry use case. Next, Section 4 provides the necessary background on the curation-based stream personalization task that is addressed by our recommender system. Note that this paper focuses on the issue

*This study is performed during the author's internship at XITE Networks International

ACM RecSys Workshop on Offline Evaluation for Recommender Systems (REVEAL 2018), October 2018, Vancouver, Canada

of offine evaluation and the specific recommender system is not described in detail. Then, in Sections 5 and 6 we describe our proposed evaluation framework and evaluation metrics. In Section 7, we summarize and describe future work, which will include empirical validation of the evaluation strategy that this paper proposes.

2 RELATED WORK

In this section, we point out key examples of two major areas of related work. First, in the area of recommendation for Internet-enabled television, [1] presents a recommender system for IPTV-based services. The duration of watching the program is mapped into ratings between 3-5. In our work, we also use a threshold on the duration of a music video in order to map the continuous value into a binary relevance score for an item. In [1], both offline and online evaluation is performed. The authors found that Item-Based Collaborative Filtering produces the best result in offline settings. Offline evaluation is performed with a leave-one-out scenario, and the recall at position 5 is used as the evaluation metric. In our work, we also explore collaborative filtering. However, the contribution of our paper is in the area of evaluation, and is not related to the algorithms that we are developing. Further, in our work, we also explore recall, but we integrate it into a metric that is aware of 'streaks', which are consecutive sequences of relevant items in the stream of recommendations.

The second area important to our work is the inclusion of the state of the user. In other words, modeling whether the user is actively consuming the recommender content or not. In the music video television scenario, the implicit feedback of the user should be treated differently depending on whether the user is actively watching the TV, or whether s/he is doing something else and the TV is playing in the background. As an example of related work on user states, we point to [4], which explores possibility of incorporating both explicit and implicit feedback based on user behaviour when listening to music to provide personalized recommendations. Some handcrafted features are derived, a few of which consider the state from which the user gives the explicit feedback, e.g., whether the user is in active or passive session [4]. While this study hints at the importance of considering the same feedback under different contexts, extensive evaluation across different recommender system algorithms is not performed.

3 USE CASE: PERSONALIZED MUSIC TV

Our use case is taken from our study at XITE Networks International (XITE)[1], a music television company active in The Netherlands, Germany, Belgium, Qatar, Canada, and the United States. XITE has developed an interactive television product called Personalized Music Television (PMT). The PMT is designed to continuously stream music videos and has multiple methods of serving music videos to its viewers. Important for this paper is that it allows viewers to select from a variety of curated *channels* such as "Hip-Hop Essentials", "All Out Dance", and "80's Flashback", or to curate their own channel by selecting specific genres, decades, moods, and visual aspects. Once a viewer selects a channel, the order of the streamed videos is controlled algorithmically, rather than by the viewer or by a curator as is typically the case with playlists. At any time the viewer can interact with the application by giving a *like* to indicate preference, or making *skip* to go to the next track. The goal at XITE is to design a recommender system that will provide the optimal ordering of a set of videos within a channel, given past viewer interactions.

[1] http://www.xitenetworks.com

3.1 Challenges

We wish to develop a recommender system that provides an optimal ranking of music videos within the curated channels of XITE's PMT product. The system leverages different types of user feedback. At this early stage of development, we chose to focus on collaborative filtering, which allows us to focus on exploiting the rich user interactions data already available to us, rather than exploring the use of content-based information. Based on our preliminary analysis of the product, we have identified several challenges as follows:

Limited user input. User interaction with the television is dominated by physical remote controls. While these are becoming ever more sophisticated, the TV remote has only limited sets of buttons available to use. Typing lots of words in particular is a burden for the user because of such limitation. Thus, searching for relevant content might not be as enjoyable as finding new music on computer or mobile phone. Considering this limitation of the remote control, typically user feedback is limited in the television scenario. In our use case, the explicit feedback is restricted to the *like* feedback, while complete watch (or a *skip*) may be considered an implicit form of feedback. The recommender system should be able to learn from only these limited types of user feedback.

Limited control to select items. In online music services such as Spotify, users can pick their own favorite songs to create playlists easily. However, in a linear music video viewing experience which lacks extended menu options, users do not have precise control over what content they will see. When the user selects a channel, a predefined personalized and curated set of music videos will be presented to them with a system-defined order. Imagine a similar situation where we tune-in to an internet radio. The songs are just continuously played until we stop the radio. This fact has consequences on the universe of items that the user can potentially observe, which eventually affect the item candidates that can be recommended to users. Although a user can search for music videos, we noticed that people only search for content in a small number of cases.

Active vs passive viewing behavior. While viewers can be actively engaged with their television experience, it is not uncommon for them to leave the television running while they perform tasks around the house. This is perhaps more true for music television, with its strong and "snackable" audio component, than for other forms of more traditional long form television. While we can infer a positive or negative preference if a user presses like or skip on a music video, we can also implicitly assume, to some extent, that when a user watches a music video completely, they give an implicit positive feedback to that item. However, we cannot be sure if that is the case, because we do not know if the user is actually paying attention to the whole music video. Without a sensor installed on the television, it is difficult to know whether the user is watching

the television or not. The availability of sensor is also intrusive and perhaps not preferable because it raises ethical and privacy issues.

Evaluation methodology and metrics. Typically, we hope that better performance in offline evaluation reflects the actual comparative online performance of the recommender system. For top-N recommendation task, ranking metrics such as Mean Reciprocal Rank (MRR), Mean Average Precision (MAP), or normalized Discounted Cumulative Gain (nDCG) are often used. While these metrics promote putting items on top of the list, there could be other requirements that are not captured, for example the consecutive occurrence of a number of relevant items. While enjoying multimedia content, the user may prefer to enjoy relevant items in a continuous sequence, rather than being presented with alternating order of relevant and irrelevant items. Typical evaluation metrics can not capture the existences of continuous sequence of relevant items across different positions in the list.

The evaluation framework also needs to be aware that the test set is composed of actions grouped by different sessions or curated channels. The occurrence of a music video with its associated feedback in a session can not be decoupled from the session, thus evaluation must be performed in a session basis, rather than on individual relevance of the music video. We elaborate on this session-based evaluation in Section 5.

3.2 Requirements

Recommender system development should be informed by an understanding of the requirements for a particular domain of application, both from user and business perspective. Based on internal discussions at XITE, we have identified two hypotheses that the company believes likely underlie relevance from the user perspective, that we wish to explore in more detail.

(1) After opening a channel, the user should be presented with the best, or most relevant, personalized content *as soon as possible*. By giving users the items that they like the most first, they will be drawn into the viewing experience, and thus be more likely to stay with and to explore the service.
(2) When enjoying streams of music videos on the PMT, it is normal for a user to regularly switch between *lean-forward* mode of active interaction, and a *lean-back* mode with passive consumption of music. In this lean-back mode, it could be helpful to maximize streaks of consecutive relevant music videos, minimizing the interruptions to the experience that occurs when a user feels the need to press skip during a session. In other words, videos should be presented so that relevant items and non-relevant items are clustered together, rather than interspersed with one another.

An overarching performance goal is to maximize the viewing duration of the PMT users. Although the two considerations itemized above originate mostly from the user perspective, we believe that if such considerations are fulfilled, users will be more engaged and satisfied with the product. Consequently, we expect the session time to be increased.

Current evaluation metrics are aimed at rewarding the early display of relevant items, but do not reward the "streaks" of relevant items identified in our second consideration. This motivates us to introduce a new evaluation metric, aimed at rewarding streaks of consecutive relevant items *as well as* rewarding high ranking of relevant items. We describe the proposed evaluation metric in Section 6.

4 CURATION-BASED RECOMMENDATION

In this section, we first give a basic description of how our curation-based recommender system works. The purpose of the description is to make clear what the input and the output of the system is. This information makes it possible to understand the evaluation framework, which is described in the next section.

4.1 Stream recommendation scenario

The basic unit in our stream recommendation scenario is the user session. A session is started when a user opens a channel and finished when they leave the channel or the application. We create sessions in the offline data by segmenting the user play history into groups of consecutive items. To formalize, let $s_{u,i}$ designates session i for a user u. A session is a list of pairs consisting of a music video v and a user action a that are ordered by time. A session of length N is defined as follows:

$$s_{u,i} = [(v_1, a_1), (v_2, a_2), ..., (v_N, a_N)] \quad (1)$$

Note that in a session it is theoretically possible that $v_i = v_j$, i.e., that the same video is played twice. However, in practice XITE streams rarely contain a repeated video. For this reason, we handle the videos in the session as if they were unique videos.

4.2 From user feedback to ground truth

In the XITE offline data, we consider the case where users perform one of two possible feedback actions a: a complete play or a skip, which means that the user jumps to the next video after watching the current video for certain time, but without reaching the end. In order to define the ground truth used within our evaluation framework, we map each user action into a binary relevance value (1 or 0).

Our goal in designing the mapping is to assign a score of 1 to actions of which reflect strong evidence of user preference, and assign a score of 0 to actions of which do not reflect evidence of preference. A complete play event is mapped into 1, but a skip cannot be directly mapped into 0. We use a threshold of 30 seconds. Music videos that are skipped after more than 30 seconds lead to a score of 1, and otherwise we assign a score of 0. We note that 30 seconds is also used in [3], as the cutoff threshold when mapping a skip action into 0 or 1. We also calculated the distributions of the durations of all skip events in our dataset and observed that 30 seconds corresponds to the 80th percentile. In other words, 80% of skips in the dataset are 30 seconds or shorter. Note that we use this mapping to create binary relevance scores for the purposes of evaluation. The recommender system algorithm being evaluated with our framework does not necessarily need to apply this mapping to the training set, but rather may integrate user feedback using a different method.

This relatively straightforward mapping hides a number of assumptions, which we discuss briefly here. First, it builds on the assumption that within a session, an action of a user on each of the music videos is the same, regardless of the ordering of the video,

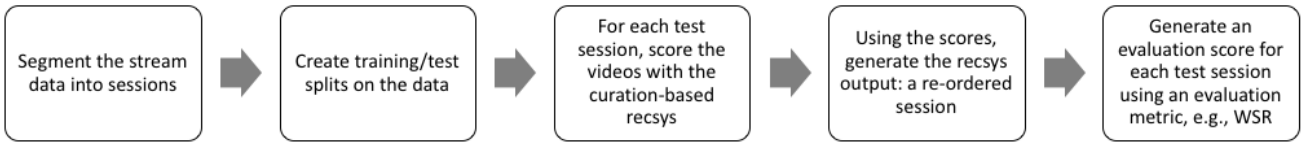


Figure 1: Pipeline for offline evaluation of curation-based stream recommendation in a music video TV use case

and that each action is independent of other actions. Naturally, the stream is a single coherent listening experience for the user. The nature of this experience leads us to expect that there is actually some correlation between those actions, for example, if many of the videos are from the same artist, the sequential ordering of the video might affect the actions. However, we point out that with curation-based recommendation, the underlying curation exerts a strong force on the resulting recommendations. In particular, it serves to control redundancy, and there is only a small probability that the videos in the stream originate from the same artists.

4.3 From user feedback to user state

Although the types of feedback actions gathered from XITE users is limited, this information does provide important hints as to the state of the user, i.e., the level of engagement with the music video stream. The feedback types in our use case can be understood as involving both explicit and implicit feedback. When the user clicks 'like', it can be considered as explicit feedback, while a skip and a complete play of music video can be considered as implicit feedback. We argue that understanding the actual utility of the feedback is more complex than understanding the effort of the user relating to the individual feedback types. A complete play for example might have some levels of uncertainty, especially in a session where there are long continuous played events without any user interruption, which can be considered as passive session [4]. As mentioned in Section 3.1, we cannot be sure that all those videos carry the same level of positive preference, because perhaps the user is not paying attention to all of those videos. Wouldn't it hurt the performance of our recommender system if we just consider all these videos with the same level of preference? If we suspect it would, the next question is how to model such uncertainty in the complete played signals?

To the best of our knowledge, there is little research on modeling user attention when watching long passive sessions with the goal of inferring different levels of implicit positive preference for individual items. We argue that such understanding is important and therefore cannot be simply neglected when developing recommender systems for a continuous streams of multimedia contents. In this work, we introduce a rudimentary method for taking user state into account. Specifically, in Section 6.3, we propose an extended version of our evaluation metrics which imposes a threshold on the length of the 'streaks' that are taken into account by our evaluation metric.

5 OFFLINE EVALUATION FRAMEWORK

Our goal is an offline evaluation framework that allows us to evaluate the same recommender algorithms that are used online at XITE. The framework consists of a definition of the recommendation task for the offline data, a procedure for splitting the data into test and training, and, finally, the evaluation pipeline. Each of these is covered in this section.

5.1 Offline curation-based recommendation

Recall that the online XITE recommender is a curation-based recommender. For a given channel, the recommender generates a personalized stream for a user by selecting items from a pre-selected item set. This item set has been chosen by human curators as being suited for the channel.

For our offline evaluation framework, we do not have access to information about which items were available in the pre-selected item set during the specific user sessions. For this reason, we need to make some well-chosen assumptions in order to create an offline evaluation framework that emulates online recommendations. The first assumption is offered by the technical specifications of our dataset (i.e., how it was generated). These specifications allow us to make the assumption that user sessions were either not personalized, lightly personalized, or personalized in a not-yet optimal way. From this point, we make a second assumption, namely, that the videos that we observe in a user session are reflective of the underlying videos in the pre-selected item set from which the recommender is allowed to choose.

We contrast our approach with one of the most popular protocols for offline evaluation of recommender systems: one-plus-random [2]. In this approach, for each user in the test set, one rating (or item) is selected as the held-out item. Then, randomly unrated items (1000 are used in [2]) are also picked, and the items together with the held-out item are ranked based on the scores produced by the algorithm. Based on this order, various types of ranking metrics can be measured. In contrast, in our use-case, we argue that randomly picking from universe of unobserved items is not fair, since there is a known control on what item is presented: the items presented are drawn from the curated item set for the channel. A more realistic assumption is to assume that the items are drawn from items in the session. In order to implement this assumption, we calculate our evaluation score at the session-level, and not at the item level. However, as we will see, a session level score allows us to capture information about streaks of consecutive relevant items. The importance of such streaks derives from our user requirements.

Based on these considerations, we arrive at the following procedure for evaluating an online curation-based recommender in the offline scenario. We evaluate the performance of the recommender algorithm for each user session in the test set individually. The input to the recommender is the unordered set of all videos occurring in that session. The recommender treats this set as the pre-selected

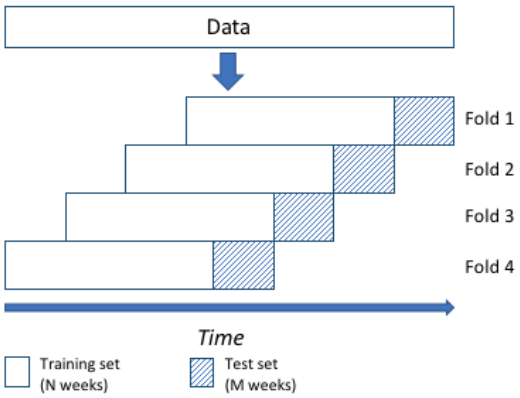


Figure 2: Time-based method: A sliding temporal window is used to create folds, each of which is split into training and test.

item set and scores each of the videos. The output of the recommender is a new session, which is an optimized re-ordering of the original input sessions. We evaluate the recommender algorithm by scoring this re-ordered session.

5.2 Splitting the offline data

We split the offline data in a way that is consistent with the actual online scenario where recommendation are made. Online, the recommender is trained on data collected for a certain period, and then makes predictions *after* that period. For this reason, we must constrain our offline evaluation framework such that the training procedure can never look forward in time to make use of future interactions.

We choose to apply a time-based method to split the data into training and test sets. Using a sliding temporal window, we create folds, following the method that has also been used in [5] as shown in Fig. 2. Each fold is split into training and test data following the same temporal split (N weeks for training and M weeks for test). For each fold, the sliding window advances by M weeks, so there is no overlap in the test sets between folds.

Recall that data used for offline evaluation is first split into user sessions, as described in Section 4.1. We note here explicitly that user sessions are defined *before* test training splits are defined. If the split between test and training falls in the middle of a user session, that user session is split into two, and considered to be two user sessions from the point of view of evaluation.

5.3 Evaluation pipeline

Our proposed evaluation pipeline is presented in Figure 1, and provides a summary of the overall process. Here, we describe each step individually.

(1) In the dataset, the user's watching history is aggregated on a per-session basis. Each session is represented as described in Equation 1.
(2) The dataset is split into folds, and each fold is split into training and test.
(3) For each fold, the curation-based recommender system is trained on the training set. This model is then applied to each video in the test set. It produces a score that is specific to the user and to the video.
(4) Based on the predicted score of music videos, a new predicted session is produced. High-scoring videos occur at the beginning of the session. Effectively, session re-ordering is a re-ranking process.
(5) Using the ground truth of binary relevance values from each music video in the test set, calculate the evaluation score for each session. The scores can then be averaged for all sessions in the test set, and/or averaged over all folds.

In the next section, we go on to introduce our metrics for evaluating sessions.

6 OFFLINE EVALUATION METRICS

In this section, we first discuss the ability of conventional, existing offline metrics to capture XITE's performance indicators. On the basis of insights revealed in this discussion, we propose a new performance metric that is specifically designed for offline evaluation of personalized music video television, and captures XITE's performance indicators better than the existing metrics.

6.1 Conventional offline metrics

Given a session $s_{u,i}$, the task of the recommender algorithm is to reorder the music videos in the list so that the order of video in the new reordered list is optimized to match the considerations described in Section 3.2. We refer to this process as 'reordering' because the music recommendations are delivered in a stream. However, if we consider that we would like to present the user with relevant content as early as possible, we can also consider the reordered stream to be a ranked list, common in Information Retrieval and top-N recommendation. Metrics that evaluate ranked lists reward relevant items occuring early in the list. The similarity between our reordered stream and conventional relevance ranking makes it natural to consider the ability of ranked-list metrics to reflect the quality of our recommendation stream. A classic ranking metric is Average Precision (AP). The precision score of the list is calculated at each rank position at which a relevant item occurs, and then these precision scores are averaged. While higher AP rewards ranked lists containing relevant items near the top, it does not take into account the fact if there is any continuous streak of relevant values in it. Continuous streaks are important since it will be inline with the second business requirement that we have described in Section 3.2.

List	AP	WSR
[0,1,0,1,0,1,0,1]	0.5	0
[0,0,1,1,0,1,1,0]	0.4762	0.25
[0,0,0,1,1,0,1,1]	0.3946	0.1964
[0,0,0,0,1,1,1,1]	0.3655	0.2

Table 1: Example of metrics calculation for two lists with the same number of relevant and irrelevant items

6.2 Weighted Streak Recall (WSR)

To take into account continuous streaks, we propose a metric called Weighted Streak Recall (WSR). WSR identifies individual streaks in the recommendation stream. It calculates the recall score of each streak, namely, the fraction of the total number of relevant items that that streak contains. It combines the recall scores by weighting them according to their position in the stream. Streaks that start early in the stream receive higher weights than streaks that start later.

The pseudo-code in Algorithm 1 specifies how WSR is calculated. We describe the WSR with respect to a 2-streak scenario, meaning that we consider a streak if there are at least two consecutive relevant items in the list. The input of the algorithm is the ground truth duration of the item in the reordered stream. This duration is then binarized with a certain threshold; by default a threshold of 30 seconds is used to indicate relevance.

Next, we give some examples to show the utility of the new proposed metric in Table 1. Both lists contain the same number of relevant and irrelevant items. Note that the first list is better in terms of AP, but fails to produce continuous streak of relevant items. The second list, on the other hand, is worse in terms of AP, but contain two streak of relevant items, both with length two. We think that the second list would be more enjoyable to the user compared to the first list where the user needs to press skip everytime after finished watching one complete video.

Also interesting is to compare the third and fourth examples. The third list is better in terms AP, but worse in terms of WSR. If we focus only on streaks, then it makes sense that the fourth list is better than the third list, because we have 4 continuous streak by only skipping one more video.

The WSR cannot be used in isolation to choose which model that would be picked among the candidates models. The system designer might need to think how to combine other metrics with WSR, which is out of the scope of this paper. For example, novelty and diversity of the session with respect to the popularity and genre of the music videos might also be considered along with WSR.

Algorithm 1 Weighted Streak Recall

procedure WeightedStreakRecall(L, t)
Form a modified list L, composed of binary relevance with a threshold of t seconds
Calculate length of the list len_l
Calculate number of relevant items n_rel from L
Initialize $group$,$group\_temp$ as empty lists
Initialize $find = False$, a boolean variable
Initialize weighted streak recall WSR to zero
Calculate list l_rel which contains the position of each relevant items in L
for $i = 0; i < len\_l; i++$ **do**
if $i \in l\_rel$ **then**
$find = True$
Add i to $group\_temp$
if $i == (len\_l - 1)$ **then**
if $length(group\_temp) \geq 2$ **then**
Add $group\_temp$ to $group$
end if
end if
else
$find = False$
if $length(group\_temp) \geq 2$ **then**
Add $group\_temp$ to $group$
end if
Assign $group\_temp$ as empty list
end if
end for
for $l \in group$ **do**
$WSR = WSR + \frac{1}{(l[0]+1)} \frac{length(l)}{n\_rel}$
end for
return WSR
end procedure

6.3 Weighted Streak Recall with Streak Threshold (WSR-t)

In this section, we point out that WSR has limited usefulness since it might capture what we think users want (i.e., our user requirements), but it is not correlated with session time, which is XITE business performance indicator. We point out one particular weakness is that we do not know the underlying cause of streaks, and long streaks might reflect lack of engagement. For this reason, streak threshold parameter should be incorporated into WSR, meaning that given a threshold of X, the utility of maintaining a streak longer than X might not be that *important* anymore, meaning that it is okay to break the streak after X continuous relevant items.

Based on our observation on user behaviour, a streak of 4 or 5 already represents between 50-60% of streaks, and might be a good candidate for threshold X. We believe that this assumption is consistent with how radio stations arrange the number of continuous playback of songs and how human attention is limited after watching continuous music videos for a certain amount of time. It might be useful to have multiple streaks instead of having a very long one to spread interesting content across the streams.

To give an illustration, after binarization of the durations of each list, let $L_1 = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]$ and $L_2 = [1, 1, 1, 1, 0, 1, 1, 0, 0, 0]$. Both L_1 and L_2 contain 6 relevant and 4 irrelevant items. If we merely use the original WSR as depicted in Algorithm 1, then WSR L_1 (1.0) is greater than L_2 (0.7222).

Now assume that the objective is changed, instead of focusing on the maximum length of streaks, we want to maximize the spread of streaks with a length threshold of X across the list. Considering this change of objective, the last term of second iteration from Algorithm 1 need to be modified (the part where we calculate $\frac{length(l)}{n\_rel}$). If $length(l)$ is greater than X, we assign $length(l)$ as equal to X. By slightly changing this term, now we have WSR-t of L_1 and L_2 equal to 0.6667 and 0.7222 respectively. Basically, we use a hard threshold X to penalize streaks longer than X. By considering the maximum threshold X, WSR-t encourages the split of streak across multiple position in the list instead of having one very long streak while still maintaining the utility of WSR to reward streaks early in the list.

7 CONCLUSION

First, in this paper we start by identifying some of the challenges that we found while experimenting with offline evaluation of recommender systems for personalized music television. We lay out some of our thinking on considering these challenges. We elaborate on how the evaluation framework should be designed, and mention our motivations on considering the evaluation on a per-session basis. We propose a new metric called Weighted Streak Recall (WSR) which might reflect better the user requirements in our domain. We also highlight the needs to interpret different types of user feedback, in particular when no explicit feedback is given, for example when inferring the meaning of a complete played and skipped event.

It is no accident that this paper does not contain any experimental results yet. While we are actively exploring the development of recommender systems considering challenges identified in this study, at this early stage of prototyping, we want to encourage discussions on this matter. The work is a preliminary examination of the problem domain, and our next steps will be to perform an extensive evaluation of the benchmark and proposed evaluation measure, contrasting the offline and online performance results. Once this evaluation is performed, we plan to provide a more complete version of our paper to the community.

For the future work, more exciting and challenging works lie ahead since this is a continuous steps toward our quest on developing personalized online recommender system for music video television. Naturally, we are interested in applying our proposed framework and metric to perform extensive evaluations of different recommender system algorithms. Furthermore, based on the identified challenges of understanding user feedback, we are also interested to explore how different levels of explicit and implicit feedback can be exploited by the recommender system algorithms. For example, we point out the potential of applying Bayesian Personalized Ranking (BPR) with different levels of feedback [6] in personalized music videos television domain.

REFERENCES

[1] Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. 2011. A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, Boston, MA, 299–331.

[2] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of Recommender Algorithms on Top-n Recommendation Tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*. ACM, New York, NY, USA, 39–46.

[3] Christopher C Johnson. 2014. Logistic Matrix Factorization for Implicit Feedback Data. In *Proceedings of NIPS 2014 Workshop on Distributed Machine Learning and Matrix Computations*. 1–9.

[4] Suzana Kordumova, Ivana Kostadinovska, Mauro Barbieri, Verus Pronk, and Jan Korst. 2010. Personalized Implicit Learning in a Music Recommender System. In *User Modeling, Adaptation, and Personalization*, Paul De Bra, Alfred Kobsa, and David Chin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 351–362.

[5] Martha Larson, Alessandro Zito, Babak Loni, and Paolo Cremonesi. 2017. Towards Minimal Necessary Data: The Case for Analyzing Training Data Requirements of Recommender Algorithms. In *Proceedings of FATREC Workshop on Responsible Recommendation*.

[6] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian Personalized Ranking with Multi-Channel User Feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 361–364.

[7] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. 2018. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval* 7, 2 (Jun 2018), 95–116.

[8] Guy Shani and Asela Gunawardana. 2011. Evaluating Recommendation Systems. In *Recommender Systems Handbook*, Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor (Eds.). Springer US, Boston, MA, 257–297.

[9] Douglas Véras, Thiago Prota, Alysson Bispo, Ricardo Prudêncio, and Carlos Ferraz. 2015. A Literature Review of Recommender Systems in the Television Domain. *Expert Syst. Appl.* 42, 22 (Dec. 2015), 9046–9076.