Project No: SCSE20-0248

NANYANG TECHNOLOGICAL UNIVERSITY

A Performance Analysis on Time-Series-Based Recommender System

Li Guanlong

School of Computer Science and Engineering

2021

Project No: SCSE20-0248

NANYANG TECHNOLOGICAL UNIVERSITY

SCSE20-0248
A Performance Analysis on Time-Series-Based Recommender System

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Computer Science of the
Nanyang Technological University

by

Li Guanlong
Matriculation Number: U1722033H

Supervisor: Assoc Prof Sun Aixin
Examiner: Mr Tan Kheng Leong

School of Computer Science and Engineering
2021

Abstract

With the ever-growing information technology, information overload has become an important issue faced by today's society. Recommender system, as a way of dealing this problem, has become an important tool in people's daily life and has been widely adopted in different businesses. In academia, it has also become a popular topic that keeps attracting the attention of researchers from many research fields. Among the papers that are published on this topic, most of the methods focus on personalized recommendation. Those methods usually adopt the idea of collaborative or content-based filtering and view the interactions between user and item as a matrix format. Their evaluation is also based on offline evaluation which usually does not consider the interactions' global timeline.

This project, on the other hand, focuses on non-personalized recommender system. Instead of considering the interactions between user and item as a matrix, the algorithm proposed in this project only cares about how many times an item is interacted. In other words, only the item's popularity is taken into account when making recommendations. In this report, a non-personalized recommender, which is based on the idea of considering item's popularity over time as a time series, is proposed. The training and evaluation of this recommender and the baseline methods also follow the global timeline and are performed in a manner that is similar to prequential evaluation. Moreover, by this evaluation and some additional explorations on the data itself, this project also provides an analysis on the temporal characteristics of the data and the recommender's performance on the data.

Acknowledgements

I would like to take this opportunity to express my gratitude to my supervisor, Associate Professor Sun Aixin, for his guidance and support throughout the Final Year Project.

Table of Contents

ABSTRACT ..... **III**

ACKNOWLEDGEMENTS ..... **IV**

TABLE OF CONTENTS ..... **V**

LIST OF TABLES ..... **VII**

LIST OF FIGURES ..... **VIII**

CHAPTER 1 INTRODUCTION **1**

1.1 MOTIVATIONS 1
1.2 OBJECTIVES AND SCOPE 2
1.3 REPORT ORGANIZATION 3
1.4 PROJECT SCHEDULE 3

CHAPTER 2 LITERATURE REVIEW **4**

2.1 ISSUES IN CURRENT RESEARCH 4
2.2 RELATED WORKS 5

CHAPTER 3 ALGORITHM DESIGN **7**

3.1 MOSTPOP 7
3.2 RECENTPOP 7
3.3 DECAYPOP 7
3.4 TIME-SERIES-BASED ALGORITHM 8

CHAPTER 4 EXPERIMENT AND DISCUSSION **10**

4.1 EXPERIMENT SETUP 10
4.2 BASELINE ALGORITHMS 11
4.3 TIME-SERIES-BASED ALGORITHM 15

4.3.1 Preselection *15*
4.3.2 Further processing *17*
4.3.3 Data Smoothing *19*
4.4 ALGORITHMS COMPARED 23
4.5 TIME SERIES CLUSTERING 24

CHAPTER 5 CONCLUSION AND FUTURE WORKS 28

5.1 CONCLUSION 28
5.2 FUTURE WORKS 28
5.2.1 Addition on Time-Series-Based Algorithm Design *28*
5.2.2 Customized Clustering Algorithm *29*

REFERENCES 30

APPENDIX 33

List of Tables

Table 1: Yearly average R-precision for MostPop……...……12

Table 2: Yearly average R-precision for RecentPop.……...…….12

Table 3: Yearly average R-precision for DecayPop.……...……..12

Table 4: Yearly average R-precision for time-series-based algorithm using linear regression……...……..……...……..……..17

Table 5: Yearly average R-precision for time-series-based algorithm using SVR with linear kernel……...……...……..……...……..…17

Table 6: Yearly average R-precision for time-series-based algorithm using SVR with RBF kernel……...……..……...……..……18

Table 7: Yearly average R-precision for time-series-based algorithm using ridge regression……...……..……...……..……..18

Table 8: Yearly average R-precision for time-series-based algorithm using Huber regression……...……..……...……..……..18

Table 9: Yearly average R-precision for time-series-based algorithm using linear regression after smoothing……...……..…….21

Table 10: Yearly average R-precision for time-series-based algorithm using SVR with linear kernel after smoothing……...……..……..21

Table 11: Yearly average R-precision for time-series-based algorithm using SVR with RBF kernel after smoothing……...……..……...21

Table 12: Yearly average R-precision for time-series-based algorithm using ridge regression after smoothing……...……..…….22

Table 13: Yearly average R-precision for time-series-based algorithm using Huber regression after smoothing……...……..…….22

Table 14: Yearly average R-precision for four methods……..……23

Table 15: Clustering experiment details……..25

List of Figures

Figure 1: Monthly average R-precision for MostPop……..……11

Figure 2: Monthly average R-precision for RecentPop……..……..…11

Figure 3: Monthly average R-precision for DecayPop…….……..…..12

Figure 4: Yearly R-precision pattern……...…….……...….13

Figure 5: Monthly R-precision pattern……...……..…….…..13

Figure 6: Weekly R-precision pattern……...……..…...…….....13

Figure 7: Yearly rating count……...……..…….....14

Figure 8: Yearly new movie count……...….……...…14

Figure 9: Yearly active user count……...…..……...……14

Figure 10: Monthly rating count……...…..……14

Figure 11: Monthly new movie count……...…..……14

Figure 12: Monthly active user count……...…..……...……..14

Figure 13: Weekly rating count……...…..……...……14

Figure 14: Weekly new movie count……..……...…..……14

Figure 15: Weekly active user count……...…..……...15

Figure 16: Ideal R-precision for th=0……...…..……..…16

Figure 17: Ideal R-precision for th=2…….……...…..……..……...16

Figure 18: Ideal R-precision for th=4……...…..……..……...16

Figure 19: Ideal R-precision for th=6……...…..……..…...16

Figure 20: Ideal R-precision for th=8……...…..…….....16

Figure 21: Ideal R-precision for th=10……...…..……....16

Figure 22: Ideal R-precision for th=0 after smoothing…….……...….20

Figure 23: Ideal R-precision for th=1 after smoothing…….……...….20

Figure 24: Ideal R-precision for th=2 after smoothing…….……...…20

Figure 25: Ideal R-precision for th=4 after smoothing…….……...….20

Figure 26: Ideal R-precision for th=6 after smoothing…….……...….20

Figure 27: Ideal R-precision for th=8 after smoothing…….……....…..20

Figure 28: Monthly average R-precision for four methods…….……...……..23

Figure 29: Yearly R-precision pattern for four methods…….……...….....23

Figure 30: Monthly R-precision pattern for four methods…….……....….....23

Figure 31: Weekly R-precision pattern for four methods……...….....24

Figure 32: Time series data in first cluster…….…….....25

Figure 33: Time series data in second cluster…….…….....25

Figure 34: Time series data in third cluster…….…….....26

Figure 35: Time series data in fourth cluster…….…….....26

Figure 36: Time series data in fifth cluster…….…….....26

Figure 37: Time series data in sixth cluster…….…….....26

Figure 38: Time series data in seventh cluster…….…….....26

Figure 39: Time series data in eighth cluster…….…….....26

Chapter 1 Introduction

1.1 Motivations

Recommender system is an effective tool to deal with information overload. It is adopted in different businesses such as video streaming, e-commerce and social media to provide their user with a better experience, which then contributes to the value of many businesses such as YouTube and Netflix [1] [2].

There are a large number of existing implementations of recommender system, from complex deep learning models to simple heuristic methods [3]. However, most of the methods focus on personalized recommendations. They usually view the interactions between user and item as a matrix and use different approaches to predict the relevance between users and items. Moreover, their evaluation methods are usually based on split-by-ratio or leave-one-out which do not consider the global timeline [4].

This project aims to move away from the personalized recommender systems and focus on the non-personalized approaches. Instead of making recommendations based on user-item relevance, this project explores the approach of considering the number of interactions received by each individual item over time as time series data. Specifically, this approach adopts the idea of time series forecasting, and performs recommendation by making prediction on each item's future popularity using its own time series. The training and evaluation done in this project also adopt and idea of prequential evaluation and take the global timeline of all the interactions into account, which means that the evaluation instances will be released for training after they are used to evaluate the model.

1.2 Objectives and Scope

This project utilizes MovieLens 25M dataset [5]. In this project, all the algorithms are trained and evaluated using this dataset and are built for movie recommendation. All the algorithms discussed in this project are popularity based top-N recommender. Popularity of a movie is defined as the number of user interactions received by the movie. Moreover, the training and evaluation of all the algorithms are designed to follow the global timeline. Global timeline refers to the timeline that is shared by all the users and items in the dataset. For instance, if the evaluation set is sampled by choosing all the interactions from a specific range of timestamps, then this evaluation follows the global timeline.

The work in this project can be divided into three parts. In the first part, three non-personalized algorithms are fine-tuned, evaluated, analysed and then used as the baseline for performance comparison. The three algorithms follow the definitions proposed in [6]. In the second part, a novel algorithm which is based on time series forecasting is proposed, fine-tuned, evaluated, analysed and then compared with the baseline methods to show its effectiveness. In the third part, an exploration is done to examine how to perform clustering on movies' time series data and how could it facilitate the popularity prediction.

In general, this project aims to (i) provide an analysis on some non-personalized recommenders' performance by applying an evaluation that follows the global timeline, (ii) prove the effectiveness of adopting the idea of time-series forecasting in non-personalized movie recommenders, and (iii) explore the temporal characteristics of movies' time-series data and examine their usefulness.

1.3 Report Organization

This report contains 5 chapters. Chapter 2 is the literature review of the current research papers and their differences from this project. Chapter 3 shows the implementation details of the algorithms used in this project. Chapter 4 includes the experiment results and the results discussion. Finally, Chapter 5 concludes this report and points out some possible future works.

1.4 Project Schedule

Aug 10 - Sept 25 (Sem 1 week 1 - week 7): Go through some academic papers to have a better idea on the recent works related to recommender system. Learn the programming practices that are needed for this project.

Sept 25 - Nov 13 (Sem 1 week 7 - week 13): Come up with a more comprehensive idea on what need to be implemented in this project. Start to work on the source code. Meanwhile, go through relevant academic papers, adjust the implementation.

Nov 13 - Jan 8 (end of Sem1 teaching week - start of Sem2): Complete majority of the source code for experiment. Start to work on the FYP interim report.

Jan 8 - Jan 22 (Sem 2 week 1 - week 2): Wrap up the interim report for submission. Continue to work on the source code for experiment.

Jan 22 – Mar 8 (Sem 2 week 2 - week 8): Finalize and complete the source code for experiment. Meanwhile, start to work on the final report.

Mar 8 - Mar 19 (Sem 2 week 8 - week 9): Complete the final report for submission.

Chapter 2 Literature Review

2.1 Issues in Current Research

Recommender system has been a popular topic for decades and has been attracting the attention of many researchers. In recent years, there are also numerous novel recommendation systems being proposed, many of which utilize deep learning model due to the flourishing development of applying deep learning method to recommender system research [7].

However, an important issue is that, how much progress is really achieved by all the research work done in this large volume of papers and publications. Novel recommendation systems, especially the ones utilizing deep learning technique which has been drawing increasing attention, face the issue of poor reproducibility and weak methodology. As suggested in [8], many of the papers either do not share the source code or do not provide enough implementation details. Besides, many papers also have methodology issues such as using baseline methods that are weak or not fine-tuned. For instance, in [9], 26 relevant papers published at prestigious scientific conferences were sampled. Out of these 26 papers, it turned out that only the methods proposed in 12 of them could be reproduced. Moreover, 11 out of the 12 reproducible methods were not able to consistently outperform existing conceptually simpler methods. Similar concerns are also discussed in [10] and [11].

Besides, there is also some issue in the evaluation methodology. As pointed out in [12], many of the recommenders proposed are evaluated without considering the global timeline. For instance, the traditional leave-one-out method samples evaluation instances by selecting each user's last interaction. However, different user's last

interaction can be performed at different timestamps. This leads to the possibility that some interaction data which occurs after some users' last interaction can be used in the training and then used to predict the interaction of such users. In [12], it was suggested that such issue can lead to the fluctuation of the evaluation result, which makes the performance comparison among different recommenders hard to be reproduced.

Additionally, for majority of the recommender algorithms, especially the personalized ones, time is usually not directly used as an important factor. Even time-aware methods, for instance, timeSVD++ proposed in [13] and Convolutional Sequence Embedding Recommendation Model proposed in [14], are usually still personalized algorithms which mainly focus on user dynamics.

As discussed in section 1.1, this project explores a novel method which uses time in a more direct manner. Instead of trying the mainstream approach of making personalized recommendations based on modeling user-item relevance, this novel method is designed to be a non-personalized method which makes recommendations only based on the predicted future popularity of the items. Moreover, the prediction of item popularity is by considering item's past popularity as time series data and performing a time series forecasting. Besides, the evaluation done in this project considers the global timeline by adopting the idea of prequential evaluation and utilizes R-precision as the evaluation metrics, which aims to make the evaluation to be a better representation of the recommenders' actual performance [15].

2.2 Related Works

This project utilizes the methods proposed in [6], namely MostPop, RecentPop and DecayPop. Their implementation details will be discussed in chapter 3. These three

methods are used as baseline methods for performance comparison with the time-series-based method. In general, these three methods are all conceptually simple non-personalized methods.

The methodology of training and evaluation in this project is inspired by the idea proposed in [12]. This paper suggests that the problem of ignoring global timeline during evaluation can lead to a biased comparison among recommenders. Therefore, in this project, the training and evaluation follow the global timeline by adopting the idea of prequential evaluation, an evaluation approach which has also been adopted in some online evaluation studies [16] [17] [18]. The details of the training and evaluation setup will be discussed in section 4.1.

Chapter 3 Algorithm Design

All recommenders experimented in this project are top-N recommenders. Specifically, they first predict the popularity of each individual movie on day t_0 based on the data up to day $t_0 - 1$. Then, a recommendation list is generated by ranking the movies based on their predicted popularity in descending order and obtaining the top-N movies. The same recommendation will be provided to all users.

3.1 MostPop

MostPop predicts movie's popularity on day t_0 based on the total number of interactions the movie received from day 1 to day $t_0 - 1$ in the training set.

3.2 RecentPop

RecentPop focuses on the recent interactions by introducing a Δt such that the predicted popularity for a movie on day t_0 is based on the total number of interactions the movie received during time period $[t_0 - \Delta t, t_0 - 1]$ in the training set.

3.3 DecayPop

DecayPop introduces a decay parameter on top of RecentPop. In DecayPop, the calculation is a weighted sum of interactions, with more recent interactions carry a higher weight. For instance, when making prediction for a movie on day t_0 given that Δt is n days and in these n days, the movie is interacted for i_1, i_2, i_3 … i_n times respectively, where i_1 is corresponding to $t_0 - 1$ (the most recent day) and i_n is corresponding to $t_0 - \Delta t$ (the least recent day). Then the predicted popularity of the movie is calculated by $\sum_{t=1}^{n} e^{-t} \cdot i_t$.

3.4 Time-Series-Based Algorithm

In the scope of this project, the time series of a movie refers to the number of interactions it received on each day during the past time period. For example, if a movie m is rated for {7, 6, 5, 4, 3, 2, 1} times during the past 7 days, where the right side indicates most recent, and the left side indicates least recent. Then, this array is called the time series of m during past 7 days. Suppose the most recent day is $t_0 - 1$, and the popularity of the movie on day t_0 is to be predicted. The difference between this time-series-based algorithm and the other three algorithms is that the other algorithms are summation based, such that the predicated popularity of m on t_0 would be a sum or weighted sum of elements in the array (if $\Delta t = 7$ days), or a sum of all the interactions received by m from day 1 to $t_0 - 1$. Meanwhile, the time-series-based approach considers this array as something continuous and makes prediction based on the idea of time series forecasting. A simple example of such time-series-based algorithm would be to predict the popularity of m on t_0 to be 0, since the numbers of interactions have a trend to decrease by 1 per day.

The time-series-based algorithm can be divided into four parts. Suppose for each movie, the length of the time series used for prediction is Δt days, and the prediction is for day t_0. In part (i), the movies that have been rated at least once during $[t_0 - \Delta t, t_0 - 1]$ are obtained, those movies are called possible candidate movies. Then, in part (ii), a preselection is done to obtain a subset of all the possible candidate movies, the preselection criterion is that only the movies which had been rated for at least th times in any of the day during $[t_0 - \Delta t, t_0 - 1]$ will be kept, where th is a variable that stands for threshold. The movies which satisfy this criterion are called preselected movies. After that, in part (iii), for each individual movie in preselected movies, its own time

series during [t_0 - Δt, t_0 - 1] is used to train a regression model, which is then used to predict the movie's popularity on day t_0. The regression models experimented in this part include linear regression, SVR with linear kernel, SVR with RBF kernel, ridge regression and Huber regression. Finally, in part (iv), based on the ranking of predicted popularity, a non-personalised top-N recommendation is performed to all active users on day t_0 for further evaluation. Active users on day t_0 refers to the users that have at least performed one action of rating on day t_0.

Chapter 4 Experiment and Discussion

4.1 Experiment Setup

MovieLens 25M dataset which is used in this project contains the user-movie interaction data between January 09, 1995 and November 21, 2019. Since recent data is a better representation of actual situation in recent time, only the data between November 01, 2009 and October 31, 2019 are used in the experiment.

For evaluation of the algorithms, the number of stars that user gives in a rating is not considered as a metric. Instead, the evaluation only focuses on whether the user performs the action of rating, i.e., a user-movie interaction. For example, if the system makes a prediction on movies' popularity on day t_0 based on the data up to day $t_0 - 1$. Then, suppose there is a movie m in the recommendation list generated based on the popularity predicted. In this case, the movie m is considered to be a correct recommendation for user u as long as user u does rate movie m on day t_0, regardless of the number of stars that the user u gives. The evaluation adopts the idea of prequential evaluation. That is to say, the evaluation set is sampled by choosing a day t_0 on global timeline and all interaction data that fall on day t_0 are used as evaluation set. After this evaluation is done, the data on day t_0 will be available to training set. Then, the next batch of evaluation set is sampled by incrementing t_0 by 1 ($t_0 = t_0 + 1$), so on and so forth. To be specific, the evaluation is done as follows:

(i) t_0 = November 01, 2014, t_{ini} = November 01, 2009

(ii) To make prediction for day t_0, MostPop uses data from $[t_{ini}, t_0 - 1]$ for training, while RecentPop, DecayPop and time-series-based algorithm use data from $[t_0 - \Delta t, t_0 - 1]$ for training

(iii) All the methods perform movie recommendation for day t_0 based on the ranking of predicted popularity of each individual movie

(iv) For each method, calculate its R-precision for day t_0

(v) $t_0 = t_0 + 1$ (increment t_0 by 1 day)

(vi) If t_0 <= October 31, 2019, jump to step (ii)

(vii) Monthly average R-precision and yearly average R-precision are calculated based on the daily R-precision that falls in each year and month interval

4.2 Baseline Algorithms

Figure 1, 2, 3 shown below indicate the monthly average R-precision for MostPop, RecentPop and DecayPop along the global timeline during evaluation. Similarly, Table 1, 2, 3 indicates the yearly average. For RecentPop and DecayPop, different values of Δt = 1, 2, 4, 8, 16, 32 days, were experimented. The separator between any two years was October 31 and November 01. For simplicity, in the rest of this report, year 2015 represents the time between November 01, 2014 and October 31, 2015, similarly for year 2016, 2017, 2018, 2019.

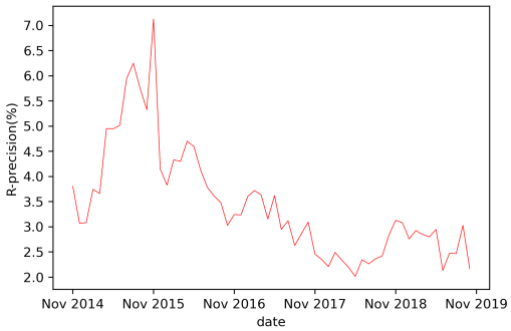

Figure 1 Monthly average R-precision for MostPop

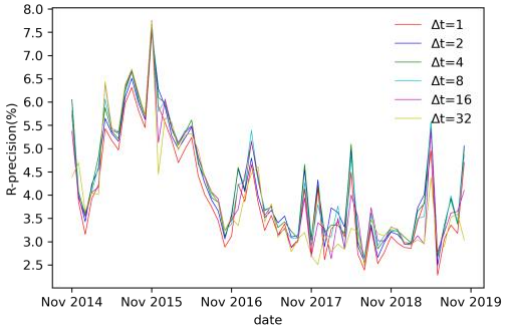

Figure 2 Monthly average R-precision for RecentPop

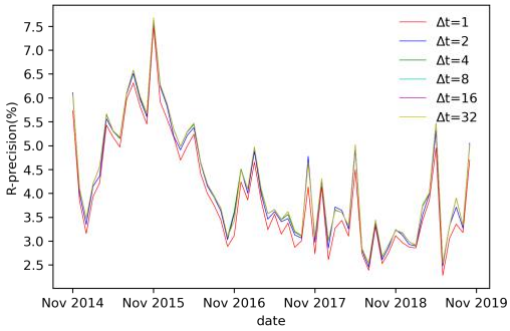

Figure 3 Monthly average R-precision for DecayPop

mostPop R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
Δt=1-32	4.630	4.244	3.234	2.354	2.725

Table 1 Yearly average R-precision for MostPop

recentPop R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
Δt=1	5.005	4.791	3.584	3.131	3.392
Δt=2	5.222	5.016	3.823	3.368	3.655
Δt=4	5.320	5.089	3.894	3.388	3.665
Δt=8	5.310	5.107	3.814	3.334	3.610
Δt=16	5.265	5.059	3.722	3.185	3.447
Δt=32	5.256	4.946	3.546	2.975	3.232

Table 2 Yearly average R-precision for RecentPop

decayPop R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
Δt=1	5.005	4.791	3.584	3.131	3.392
Δt=2	5.180	4.973	3.819	3.317	3.581
Δt=4	5.239	5.029	3.865	3.367	3.639
Δt=8	5.244	5.035	3.858	3.372	3.644
Δt=16	5.244	5.035	3.858	3.371	3.645
Δt=32	5.244	5.035	3.858	3.371	3.645

Table 3 Yearly average R-precision for DecayPop

From figures 1, 2, 3 and tables 1, 2, 3 above, it can be concluded that (i) The optimal Δt was 4 days for RecentPop, and 8 days for DecayPop, (ii) when using optimal Δt, RecentPop outperformed DecayPop and MostPop and (iii) R-precision calculated had a tendency to reduce as time moved towards a more recent time. In the rest of this report, Δt = 4 days is used for RecentPop, Δt = 8 days is used for DecayPop unless otherwise stated.

Figure 4, 5, 6 shown below indicate the yearly, monthly and weekly patterns of R-precision calculated based on the evaluation set. Yearly, monthly and weekly patterns were calculated by taking the average value of all the daily R-precision that falls in each specific year, month and weekday, respectively.

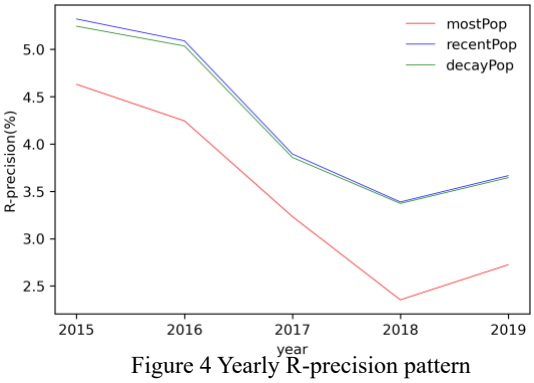

Figure 4 Yearly R-precision pattern

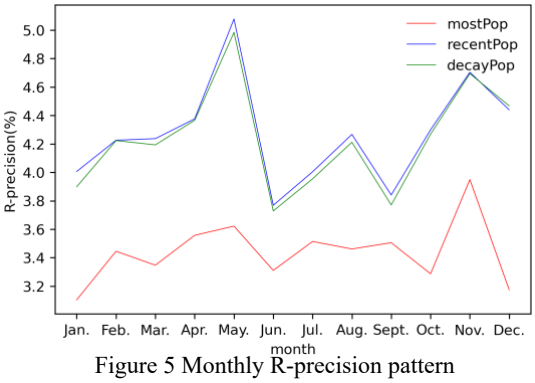

Figure 5 Monthly R-precision pattern

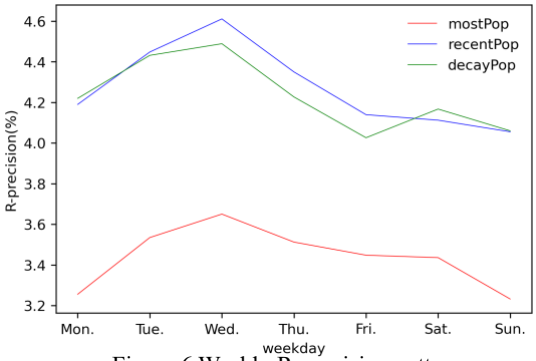

Figure 6 Weekly R-precision pattern

From above figures, it can be concluded that (i) apart from the small increase of yearly R-precision on 2019, the yearly R-precision had a tendency to drop as time moved towards a more recent time, which confirms with the observation from Figure 1, 2, 3, (ii) MostPop, RecentPop and DecayPop generally had similar pattern, (iii) monthly R-precision pattern had two local maxima on May and November, and two minima on January and June, and (iv) weekly R-precision pattern reached maxima on Wednesday and reached minima on Monday and Sunday.

In order to find the possible explanations behind the R-precision patterns, below figures were plotted. Figure 7, 10, 13 show the total number of ratings, i.e., user-item interactions, that falls in each year, month and weekday. Figure 8, 11, 14 show the total number of new movies that are induced in each year, month and weekday. Figure 9, 12, 15 show the total number of active users that are presented in each year, month, and weekday. Active user was defined as user who provided at least 1 rating during the

specific year, month, or weekday. For a clearer comparison, yearly, monthly, weekly R-precision patterns were also plotted in yearly, monthly, weekly associated figures.

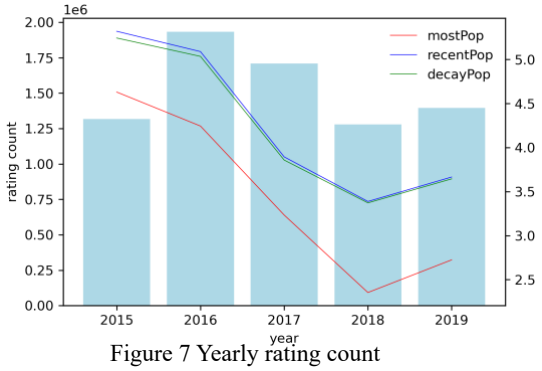

Figure 7 Yearly rating count

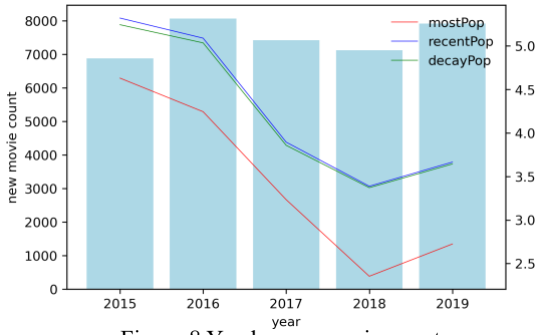

Figure 8 Yearly new movie count

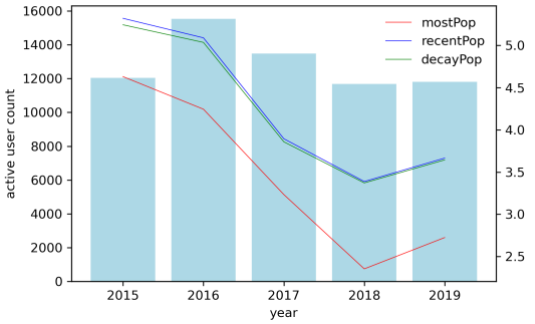

Figure 9 Yearly active user count

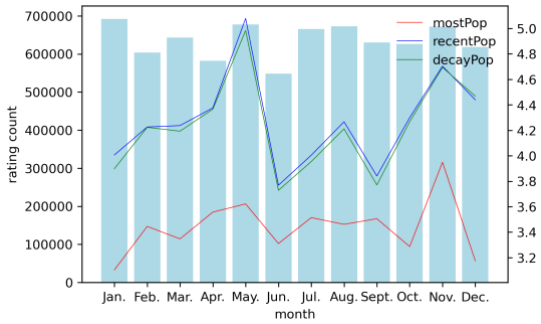

Figure 10 Monthly rating count

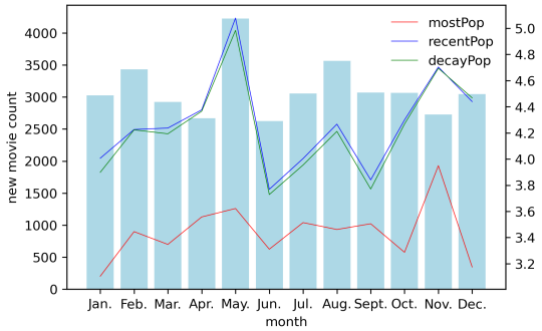

Figure 11 Monthly new movie count

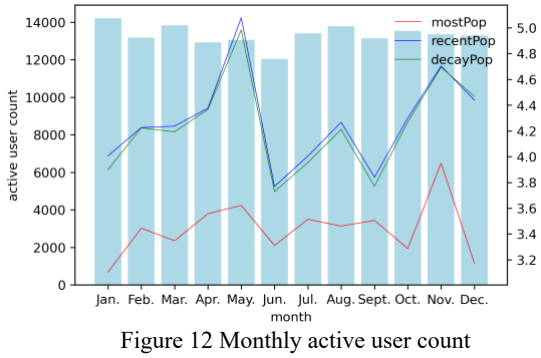

Figure 12 Monthly active user count

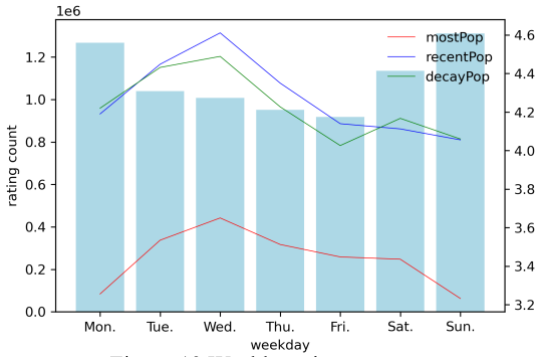

Figure 13 Weekly rating count

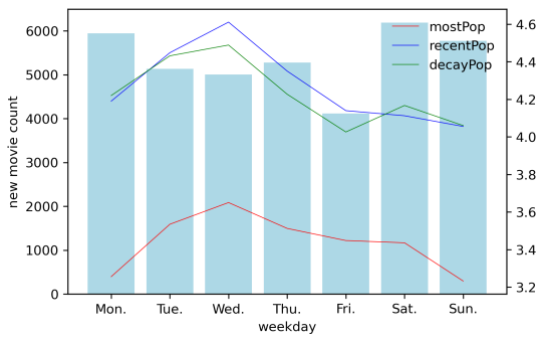

Figure 14 Weekly new movie count

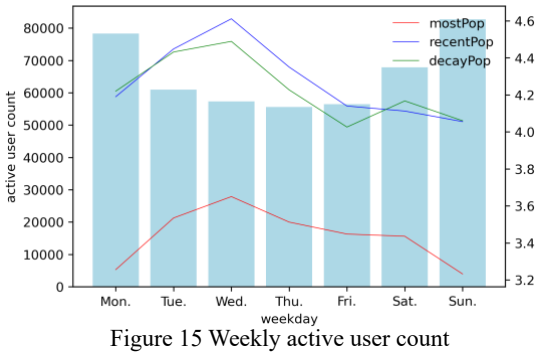

Figure 15 Weekly active user count

From the above figures, it can be concluded that (i) yearly and monthly patterns did not show an obvious correlation with the rating count, new movie count or active user count of the year and month, (ii) weekday pattern showed a negative correlation with the rating count and active user count. The two weekly R-precision minima, Sunday and Monday were also the two weekdays that had largest number of rating and active user. In general, although the correlation did not appear to be very strong according to the figures above, there do exist some temporal patterns in rating count, new movie count, active user count and the recommenders' R-precision when taking the scale of year, month and week.

4.3 Time-Series-Based Algorithm

4.3.1 Preselection

Time-series-based algorithm is computationally expensive due to that it performs one set of operations per movie. Therefore, there is a need for preselection in order to only keep a subset of the possible candidate movies. The goal is to reduce the computation cost while having a relatively small impact on the performance. The preselection will be done by setting a threshold th based on the number of ratings a movie received during each of the past Δt days as described in section 3.4.

As shown in Figure 16, 17, 18, 19, 20, 21 below, different th values of 0, 2, 4, 6, 8, 10 were experimented, and their corresponding ideal R-precision values were plotted. Ideal R-precision was calculated based on the assumption of perfect ranking of the recommendation list. That is to say, as long as the movie rated by the user is among the preselected movies, the recommendation would be considered as a correct one.

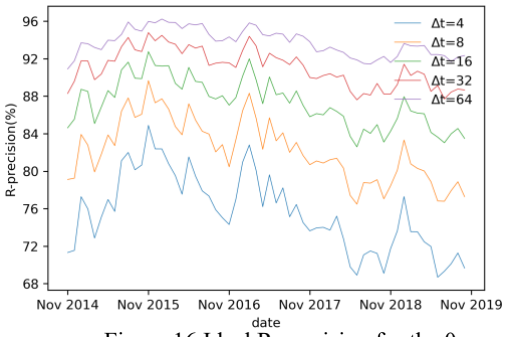

Figure 16 Ideal R-precision for th=0

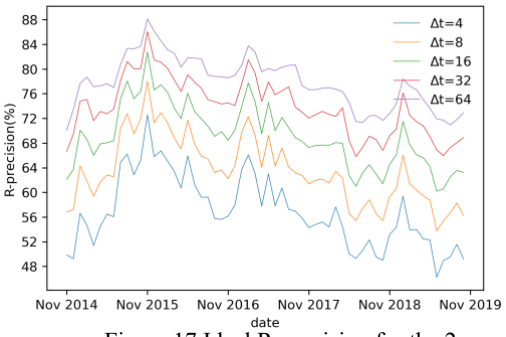

Figure 17 Ideal R-precision for th=2

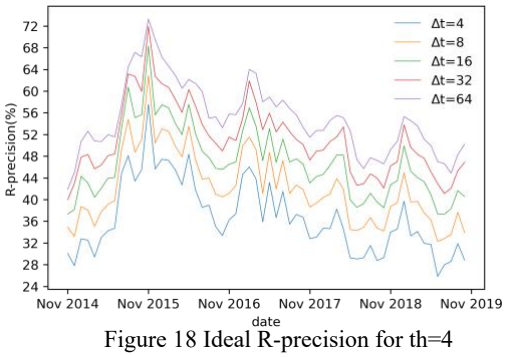

Figure 18 Ideal R-precision for th=4

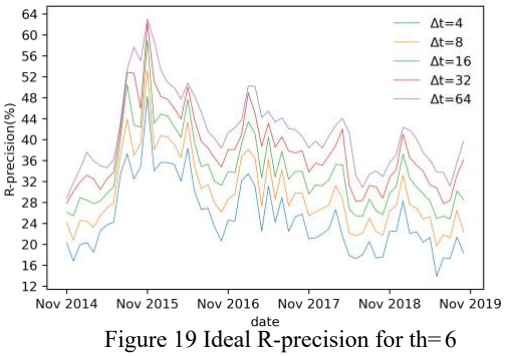

Figure 19 Ideal R-precision for th=6

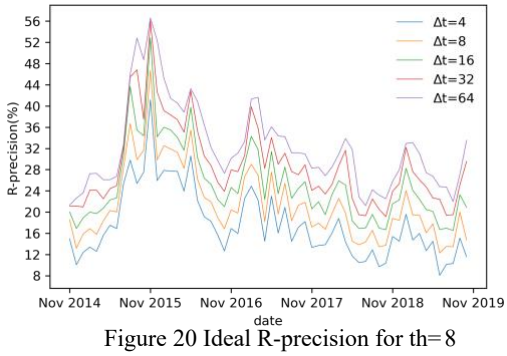

Figure 20 Ideal R-precision for th=8

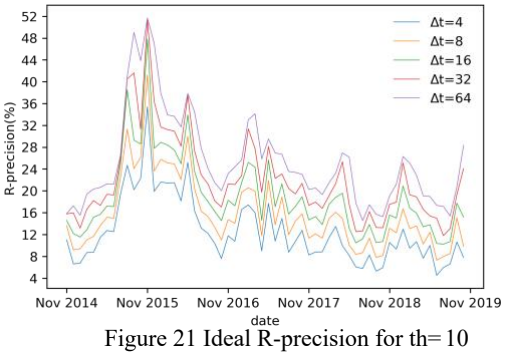

Figure 21 Ideal R-precision for th=10

From above figures, it can be concluded that when th = 2, 4, 6, 8, the ideal R-precision could generally stay above 10%, which should be sufficient to preserve the

performance of the recommender while filtering out some of the less popular movies. Those 4 values were experimented later for parameter selection.

4.3.2 **Further processing**

After the preselection was done, for each of the preselected movies, its time series during $[t_0 - \Delta t, t_0 - 1]$ was used to train each of the five regression models, which were then used to predict the popularity of that specific movie on day t_0. For each regression model, different values of Δt = 4, 8, 16, 32, 64 days, and different values of th = 2, 4, 6, 8 were experimented. The yearly average R-precision along the global timeline for each regression model were as shown in Table 4, 5, 6, 7, 8 below.

linear regression R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=2, Δt=4	4.828	4.629	3.437	2.995	3.223
th=2, Δt=8	5.184	4.958	3.849	3.401	3.638
th=2, Δt=16	5.268	5.087	3.977	3.419	3.757
th=2, Δt=32	5.347	5.155	3.873	3.381	3.654
th=2, Δt=64	5.315	5.122	3.716	3.271	3.453
th=4, Δt=4	4.838	4.633	3.444	3.008	3.232
th=4, Δt=8	5.177	4.954	3.845	3.395	3.627
th=4, Δt=16	5.257	5.083	3.972	3.411	3.747
th=4, Δt=32	5.340	5.151	3.869	3.376	3.646
th=4, Δt=64	5.309	5.119	3.713	3.267	3.447
th=6, Δt=4	4.751	4.607	3.422	2.959	3.175
th=6, Δt=8	5.092	4.927	3.814	3.354	3.578
th=6, Δt=16	5.186	5.063	3.947	3.377	3.708
th=6, Δt=32	5.295	5.138	3.854	3.350	3.619
th=6, Δt=64	5.275	5.110	3.703	3.250	3.427
th=8, Δt=4	4.527	4.527	3.327	2.767	2.983
th=8, Δt=8	4.913	4.855	3.728	3.217	3.432
th=8, Δt=16	5.039	5.010	3.882	3.278	3.602
th=8, Δt=32	5.188	5.103	3.808	3.278	3.550
th=8, Δt=64	5.185	5.089	3.673	3.202	3.381

Table 4 Yearly average R-precision for time-series-based algorithm using linear regression

SVR with linear kernel R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=2, Δt=4	4.998	4.815	3.573	3.030	3.376
th=2, Δt=8	5.070	4.805	3.716	3.228	3.454
th=2, Δt=16	5.211	4.935	3.775	3.251	3.539
th=2, Δt=32	5.283	5.009	3.754	3.088	3.452
th=2, Δt=64	5.240	5.027	3.577	3.015	3.302
th=4, Δt=4	4.987	4.813	3.571	3.028	3.371
th=4, Δt=8	5.069	4.804	3.715	3.227	3.451
th=4, Δt=16	5.209	4.934	3.773	3.249	3.535
th=4, Δt=32	5.283	5.007	3.753	3.088	3.450
th=4, Δt=64	5.243	5.026	3.577	3.016	3.301
th=6, Δt=4	4.884	4.776	3.533	2.966	3.300
th=6, Δt=8	4.996	4.781	3.688	3.191	3.408
th=6, Δt=16	5.148	4.917	3.752	3.220	3.502
th=6, Δt=32	5.245	4.996	3.739	3.065	3.426
th=6, Δt=64	5.216	5.019	3.568	3.002	3.284
th=8, Δt=4	4.638	4.673	3.409	2.755	3.089
th=8, Δt=8	4.835	4.714	3.606	3.065	3.273
th=8, Δt=16	5.012	4.867	3.690	3.129	3.405
th=8, Δt=32	5.148	4.963	3.697	2.999	3.363
th=8, Δt=64	5.133	4.999	3.540	2.957	3.242

Table 5 Yearly average R-precision for time-series-based algorithm using SVR with linear kernel

SVR with RBF kernel R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=2, Δt=4	5.267	5.035	3.854	3.314	3.618
th=2, Δt=8	5.273	5.053	3.803	3.256	3.539
th=2, Δt=16	5.270	5.008	3.655	3.051	3.400
th=2, Δt=32	5.259	4.919	3.497	2.949	3.302
th=2, Δt=64	5.188	4.882	3.446	2.878	3.272
th=4, Δt=4	5.221	5.016	3.835	3.283	3.582
th=4, Δt=8	5.253	5.045	3.794	3.242	3.522
th=4, Δt=16	5.260	5.004	3.651	3.045	3.390
th=4, Δt=32	5.256	4.917	3.495	2.947	3.298
th=4, Δt=64	5.188	4.880	3.445	2.878	3.270
th=6, Δt=4	5.039	4.947	3.758	3.155	3.451
th=6, Δt=8	5.140	5.007	3.751	3.178	3.450
th=6, Δt=16	5.187	4.982	3.624	3.007	3.347
th=6, Δt=32	5.212	4.904	3.479	2.922	3.271
th=6, Δt=64	5.157	4.872	3.436	2.862	3.252
th=8, Δt=4	4.723	4.804	3.579	2.880	3.178
th=8, Δt=8	4.922	4.918	3.641	3.006	3.277
th=8, Δt=16	5.029	4.926	3.553	2.898	3.237
th=8, Δt=32	5.105	4.869	3.434	2.850	3.203
th=8, Δt=64	5.074	4.851	3.406	2.814	3.207

Table 6 Yearly average R-precision for time-series-based algorithm using SVR with RBF kernel

ridge regression R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=2, Δt=4	4.966	4.767	3.560	3.089	3.329
th=2, Δt=8	5.195	4.963	3.860	3.411	3.650
th=2, Δt=16	5.269	5.087	3.978	3.420	3.758
th=2, Δt=32	5.347	5.153	3.873	3.382	3.654
th=2, Δt=64	5.315	5.123	3.716	3.271	3.453
th=4, Δt=4	4.962	4.765	3.560	3.092	3.326
th=4, Δt=8	5.187	4.959	3.855	3.404	3.639
th=4, Δt=16	5.259	5.082	3.973	3.412	3.747
th=4, Δt=32	5.340	5.150	3.869	3.376	3.646
th=4, Δt=64	5.310	5.120	3.714	3.267	3.447
th=6, Δt=4	4.852	4.727	3.523	3.026	3.252
th=6, Δt=8	5.101	4.931	3.823	3.361	3.588
th=6, Δt=16	5.187	5.062	3.948	3.378	3.708
th=6, Δt=32	5.295	5.136	3.853	3.350	3.619
th=6, Δt=64	5.275	5.111	3.703	3.250	3.427
th=8, Δt=4	4.611	4.630	3.404	2.817	3.044
th=8, Δt=8	4.919	4.858	3.736	3.223	3.441
th=8, Δt=16	5.040	5.010	3.883	3.278	3.602
th=8, Δt=32	5.188	5.102	3.808	3.278	3.550
th=8, Δt=64	5.185	5.090	3.673	3.202	3.381

Table 7 Yearly average R-precision for time-series-based algorithm using ridge regression

huber regression R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=2, Δt=4	4.765	4.581	3.400	2.952	3.164
th=2, Δt=8	5.132	4.870	3.745	3.273	3.523
th=2, Δt=16	5.244	5.013	3.865	3.344	3.594
th=2, Δt=32	5.335	5.075	3.837	3.212	3.504
th=2, Δt=64	5.291	5.049	3.665	3.100	3.342
th=4, Δt=4	4.786	4.590	3.411	2.972	3.179
th=4, Δt=8	5.130	4.868	3.742	3.270	3.517
th=4, Δt=16	5.236	5.009	3.860	3.337	3.584
th=4, Δt=32	5.328	5.072	3.833	3.207	3.497
th=4, Δt=64	5.286	5.046	3.663	3.096	3.336
th=6, Δt=4	4.716	4.570	3.394	2.931	3.130
th=6, Δt=8	5.052	4.844	3.716	3.237	3.474
th=6, Δt=16	5.170	4.990	3.836	3.305	3.547
th=6, Δt=32	5.285	5.058	3.818	3.182	3.469
th=6, Δt=64	5.252	5.037	3.652	3.079	3.316
th=8, Δt=4	4.503	4.497	3.301	2.748	2.951
th=8, Δt=8	4.883	4.780	3.637	3.109	3.338
th=8, Δt=16	5.031	4.938	3.772	3.207	3.445
th=8, Δt=32	5.181	5.024	3.773	3.110	3.401
th=8, Δt=64	5.164	5.016	3.622	3.031	3.270

Table 8 Yearly average R-precision for time-series-based algorithm using Huber regression

From above tables, it can be concluded that (i) linear regression and ridge regression achieved the highest R-precision, for both of them, the optimal parameters were th = 2 and Δt = 16 days, (ii) ridge regression slightly outperformed linear regression, but the increase in R-precision was only in the scale of 0.001%, and (iii) with the same Δt value, the R-precision tended to drop as the value of th increased.

4.3.3 **Data Smoothing**

The data smoothing was performed by:

(i) Change the unit of Δt from day to week, such that the movie's time series during $[t_0 – 7*\Delta t, t_0 - 1]$ is used when making prediction for day t_0

(ii) For the movie's time series during $[t_0 – 7*\Delta t, t_0 - 1]$, calculate an average value for each 7-day time interval

(iii) Obtain an array of such average values, such that the array's length=Δt

(iv) Change the preselection criterion to that at least one value in the array should be larger or equal to th

(v) Train each of the five regression models using the array of average values

Apart from the points listed above, the experiment setup and the rest of the algorithm design remained the same as what were discussed in previous sections. With this data smoothing, a longer time series was considered, and the training data became smoother. In the rest of this section, the impact of this action will be examined.

Firstly, as shown in Figure 22, 23, 24, 25, 26, 27 below, different th values of 0, 1, 2, 4, 6, 8 were experimented, and their corresponding ideal R-precision values were

plotted. The th values were selected differently compared with section 4.3.1, since the preselection criterion became more restrictive given the same th value after performing the data smoothing.

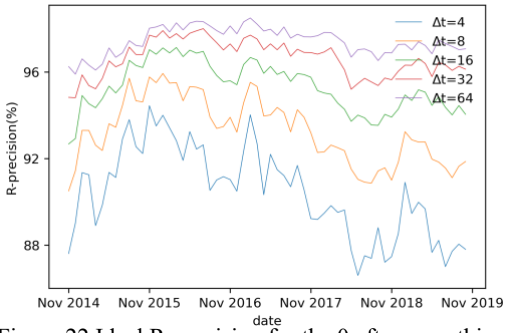

Figure 22 Ideal R-precision for th=0 after smoothing

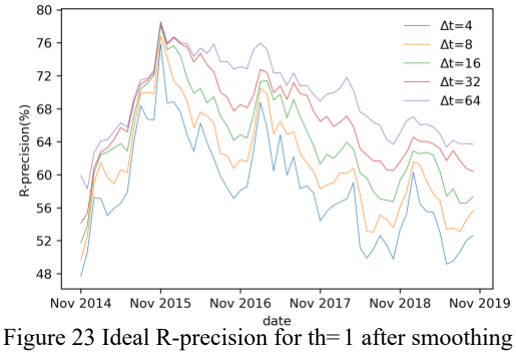

Figure 23 Ideal R-precision for th=1 after smoothing

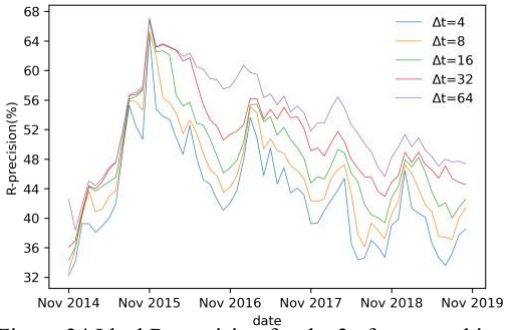

Figure 24 Ideal R-precision for th=2 after smoothing

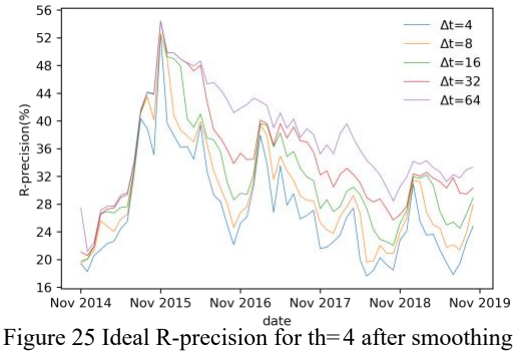

Figure 25 Ideal R-precision for th=4 after smoothing

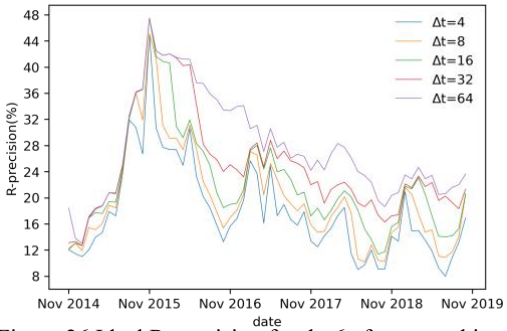

Figure 26 Ideal R-precision for th=6 after smoothing

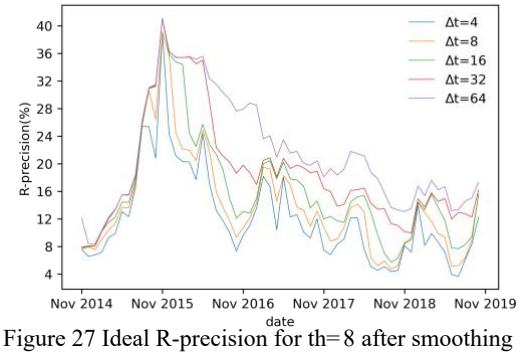

Figure 27 Ideal R-precision for th=8 after smoothing

From above figures, it can be concluded that when th = 1, 2, 4, 6, the ideal R-precision could generally stay above 10%, which should be sufficient to preserve the performance of the recommender while filtering out some of the less popular movies. Those 4 values were experimented later for parameter selection.

For each regression model, different values of Δt = 4, 8, 16, 32, 64 weeks, and different values of th = 1, 2, 4, 6 were experimented. Their yearly average R-precision values along the global timeline were as shown in Table 9, 10, 11, 12, 13 below.

linear regression R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=1, Δt=4	5.342	5.126	4.030	3.428	3.677
th=1, Δt=8	5.330	5.129	3.801	3.322	3.527
th=1, Δt=16	5.239	5.112	3.608	3.100	3.291
th=1, Δt=32	5.208	5.048	3.662	3.093	3.160
th=1, Δt=64	5.137	4.698	3.631	2.812	3.066
th=2, Δt=4	5.326	5.120	4.023	3.415	3.664
th=2, Δt=8	5.314	5.124	3.796	3.311	3.515
th=2, Δt=16	5.224	5.108	3.604	3.092	3.282
th=2, Δt=32	5.195	5.045	3.659	3.088	3.153
th=2, Δt=64	5.127	4.696	3.629	2.809	3.060
th=4, Δt=4	5.174	5.068	3.965	3.314	3.572
th=4, Δt=8	5.174	5.086	3.747	3.229	3.441
th=4, Δt=16	5.096	5.081	3.567	3.030	3.222
th=4, Δt=32	5.076	5.030	3.634	3.052	3.108
th=4, Δt=64	5.038	4.687	3.614	2.789	3.027
th=6, Δt=4	4.823	4.945	3.814	3.021	3.311
th=6, Δt=8	4.865	4.985	3.623	2.978	3.238
th=6, Δt=16	4.829	5.013	3.467	2.849	3.064
th=6, Δt=32	4.830	4.998	3.566	2.945	2.998
th=6, Δt=64	4.830	4.669	3.575	2.737	2.942

Table 9 Yearly average R-precision for time-series-based algorithm using linear regression after smoothing

SVR with linear kernel R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=1, Δt=4	5.252	5.022	3.754	3.124	3.429
th=1, Δt=8	5.283	5.009	3.692	3.108	3.290
th=1, Δt=16	5.121	5.066	3.614	2.928	3.135
th=1, Δt=32	5.232	5.005	3.627	3.001	3.137
th=1, Δt=64	5.086	4.695	3.584	2.750	3.018
th=2, Δt=4	5.231	5.014	3.745	3.109	3.412
th=2, Δt=8	5.266	5.003	3.687	3.097	3.278
th=2, Δt=16	5.107	5.063	3.610	2.920	3.126
th=2, Δt=32	5.220	5.002	3.624	2.997	3.130
th=2, Δt=64	5.077	4.693	3.581	2.748	3.013
th=4, Δt=4	5.062	4.957	3.681	2.997	3.309
th=4, Δt=8	5.128	4.964	3.638	3.014	3.204
th=4, Δt=16	4.983	5.036	3.573	2.860	3.067
th=4, Δt=32	5.103	4.988	3.600	2.962	3.087
th=4, Δt=64	4.991	4.685	3.566	2.730	2.982
th=6, Δt=4	4.702	4.825	3.520	2.689	3.030
th=6, Δt=8	4.825	4.863	3.515	2.767	3.003
th=6, Δt=16	4.721	4.968	3.474	2.681	2.912
th=6, Δt=32	4.865	4.956	3.532	2.855	2.978
th=6, Δt=64	4.783	4.667	3.527	2.677	2.897

Table 10 Yearly average R-precision for time-series-based algorithm using SVR with linear kernel after smoothing

SVR with RBF kernel R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=1, Δt=4	5.227	4.996	3.602	3.021	3.299
th=1, Δt=8	5.089	4.874	3.498	2.785	3.108
th=1, Δt=16	5.041	4.750	3.415	2.675	3.036
th=1, Δt=32	4.983	4.598	3.390	2.484	2.880
th=1, Δt=64	4.953	4.533	3.365	2.464	2.878
th=2, Δt=4	5.205	4.988	3.594	3.006	3.282
th=2, Δt=8	5.072	4.868	3.492	2.774	3.096
th=2, Δt=16	5.027	4.746	3.411	2.667	3.026
th=2, Δt=32	4.971	4.596	3.387	2.479	2.873
th=2, Δt=64	4.944	4.532	3.362	2.461	2.872
th=4, Δt=4	5.033	4.930	3.528	2.890	3.176
th=4, Δt=8	4.930	4.827	3.441	2.687	3.018
th=4, Δt=16	4.899	4.718	3.372	2.604	2.965
th=4, Δt=32	4.854	4.581	3.361	2.441	2.828
th=4, Δt=64	4.853	4.523	3.347	2.440	2.840
th=6, Δt=4	4.665	4.794	3.359	2.571	2.885
th=6, Δt=8	4.635	4.722	3.310	2.429	2.804
th=6, Δt=16	4.635	4.648	3.270	2.417	2.803
th=6, Δt=32	4.610	4.548	3.290	2.330	2.715
th=6, Δt=64	4.641	4.505	3.305	2.382	2.753

Table 11 Yearly average R-precision for time-series-based algorithm using SVR with RBF kernel after smoothing

ridge regression R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=1, Δt=4	5.340	5.147	3.900	3.373	3.663
th=1, Δt=8	5.329	5.132	3.789	3.316	3.524
th=1, Δt=16	5.239	5.108	3.607	3.097	3.287
th=1, Δt=32	5.208	5.048	3.661	3.093	3.160
th=1, Δt=64	5.137	4.698	3.631	2.812	3.065
th=2, Δt=4	5.322	5.141	3.892	3.360	3.649
th=2, Δt=8	5.312	5.127	3.784	3.305	3.512
th=2, Δt=16	5.225	5.105	3.603	3.088	3.278
th=2, Δt=32	5.195	5.045	3.658	3.088	3.153
th=2, Δt=64	5.127	4.696	3.629	2.809	3.060
th=4, Δt=4	5.163	5.086	3.831	3.254	3.551
th=4, Δt=8	5.172	5.089	3.735	3.223	3.438
th=4, Δt=16	5.096	5.078	3.566	3.026	3.218
th=4, Δt=32	5.076	5.030	3.633	3.052	3.108
th=4, Δt=64	5.038	4.687	3.614	2.789	3.027
th=6, Δt=4	4.803	4.957	3.673	2.951	3.279
th=6, Δt=8	4.862	4.988	3.610	2.972	3.234
th=6, Δt=16	4.829	5.010	3.466	2.845	3.060
th=6, Δt=32	4.830	4.998	3.565	2.944	2.998
th=6, Δt=64	4.830	4.669	3.575	2.737	2.942

Table 12 Yearly average R-precision for time-series-based algorithm using ridge regression after smoothing

huber regression R-precision(%)

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
th=1, Δt=4	5.352	5.096	4.027	3.458	3.683
th=1, Δt=8	5.387	5.098	3.785	3.244	3.430
th=1, Δt=16	5.140	5.106	3.620	2.941	3.138
th=1, Δt=32	5.228	5.026	3.664	3.014	3.143
th=1, Δt=64	5.091	4.703	3.607	2.758	3.019
th=2, Δt=4	5.337	5.091	4.021	3.446	3.670
th=2, Δt=8	5.372	5.092	3.780	3.234	3.418
th=2, Δt=16	5.126	5.102	3.616	2.932	3.128
th=2, Δt=32	5.217	5.024	3.661	3.009	3.136
th=2, Δt=64	5.081	4.701	3.604	2.755	3.013
th=4, Δt=4	5.186	5.039	3.963	3.346	3.579
th=4, Δt=8	5.236	5.054	3.732	3.152	3.345
th=4, Δt=16	5.001	5.076	3.579	2.870	3.068
th=4, Δt=32	5.103	5.009	3.636	2.972	3.091
th=4, Δt=64	4.996	4.692	3.589	2.735	2.980
th=6, Δt=4	4.837	4.917	3.812	3.054	3.319
th=6, Δt=8	4.930	4.953	3.608	2.901	3.143
th=6, Δt=16	4.736	5.007	3.478	2.688	2.909
th=6, Δt=32	4.867	4.977	3.567	2.863	2.980
th=6, Δt=64	4.795	4.674	3.548	2.680	2.894

Table 13 Yearly average R-precision for time-series-based algorithm using Huber regression after smoothing

From above tables, it can be concluded that after data smoothing, (i) Huber regression achieved the highest R-precision when th = 1 and Δt = 4 weeks, (ii) the R-precision tended to drop as the value of th and Δt increased.

However, even for the Huber regression with its optimal parameters, its performance in terms of R-precision was still lower than the ridge regression with its optimal parameters but without data smoothing. In general, based on the experiment results, it can be concluded that this data smoothing approach did not work well for the time-series-based algorithm proposed. In the rest of this report, the best performing combination, ridge regression with th = 2, Δt = 16 days without data smoothing, will be used to represent the time-series-based algorithm.

4.4 Algorithms Compared

In Table 14 and Figure 28 below, the yearly and monthly average R-precision of the time-series-based method were plotted together with those of MostPop, RecentPop and DecayPop. Both the time-series-based method and the baseline methods used their respective optimal parameters for comparison.

	Nov 2015	Nov 2016	Nov 2017	Nov 2018	Nov 2019
mostPop	4.630	4.244	3.234	2.354	2.725
recentPop	5.320	5.089	3.894	3.388	3.665
decayPop	5.244	5.035	3.858	3.372	3.644
time-series-based	5.269	5.087	3.978	3.420	3.758

Table 14 Yearly average R-precision for four methods

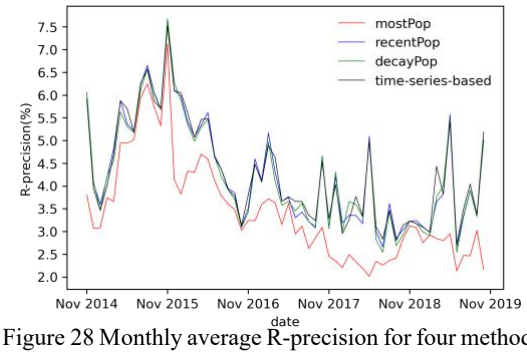

Figure 28 Monthly average R-precision for four methods

It can be concluded that when using ridge regression with th = 2, Δt = 16 days in the time-series-based method, the R-precision it achieved was higher than RecentPop with Δt = 4, DecayPop with Δt = 8 and significantly higher than MostPop.

In Figure 29, 30, 31 below, the yearly, monthly and weekly R-precision patterns of the time-series-based method were plotted together with those of MostPop, RecentPop and DecayPop, all the methods used optimal parameters.

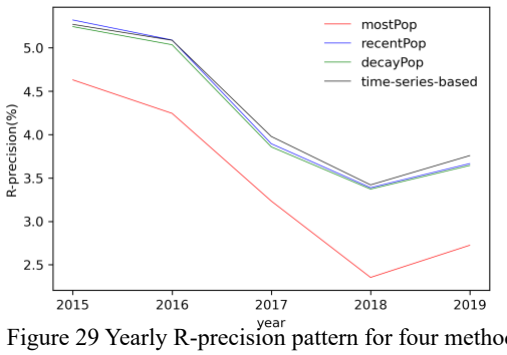

Figure 29 Yearly R-precision pattern for four methods

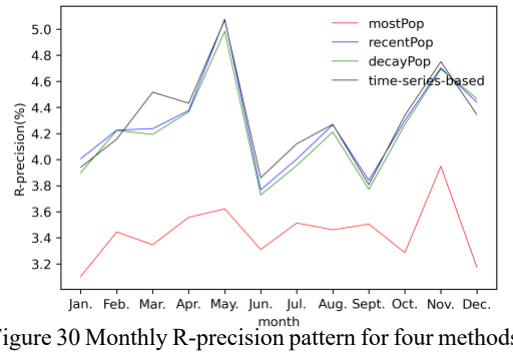

Figure 30 Monthly R-precision pattern for four methods

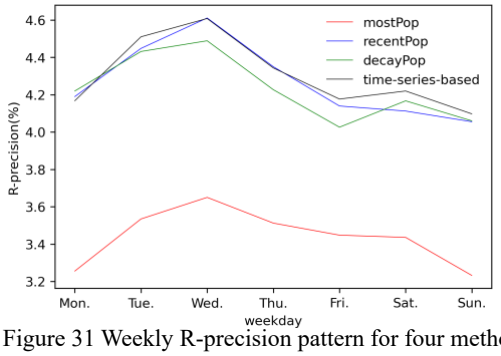

Figure 31 Weekly R-precision pattern for four methods

From above figures, it can be concluded that the yearly, monthly and weekly R-precision patterns of the time-series-based method were similar to the patterns of MostPop, RecentPop and DecayPop. Therefore, the findings concluded in section 4.2 still applied.

4.5 Time Series Clustering

In this section, some exploration on clustering was done to find additional temporal patterns within the movies' time series data.

The clustering was performed as follows:

(i) For each movie presented between November 01, 2009 and October 31, 2019 in the MovieLens 25M dataset, its 1-year (364 days) time series starting from its release date is obtained

(ii) Only keep the movies that received no less than 100 ratings during the 1-year time series (among all 58594 movies, only 2746 of them fit this criterion)

(iii) Perform a data smoothing by taking average value in every 7-day time interval, so the length of each time series becomes 364 / 7 = 52

(iv) Perform clustering based on cosine similarity between time series data

(v) Utilize HDBSCAN algorithm for clustering

For the HDBSCAN algorithm, different min cluster size values of 2, 3, 4, 5, 6, 7, 8, 9, 10 were experimented. The optimal min cluster size was decided based on (i) number of clusters formed, (ii) number of outliers, (iii) number of movies that are outside the major cluster, and (iv) shape of actual formed clusters.

Based on the experiment results in Table 15 below, and the visualizations of clusters formed, the most proper min cluster size was decided to be 6.

	num of cluters	num of outliers	num of movies outside major cluster
min cluster size=2	2	1	5
min cluster size=3	2	1	5
min cluster size=4	2	2	5
min cluster size=5	2	2	5
min cluster size=6	8	1982	104
min cluster size=7	6	1783	76
min cluster size=8	3	1837	33
min cluster size=9	4	1827	41
min cluster size=10	3	1852	20

Table 15 Clustering experiment details

When min cluster size was 6, there were 8 clusters formed, 1982 time series data were considered as outliers, and 660 movies were clustered into the major cluster. The sets of time series in each cluster were plotted in Figure 32, 33, 34, 35, 36, 37, 38, 39 below.

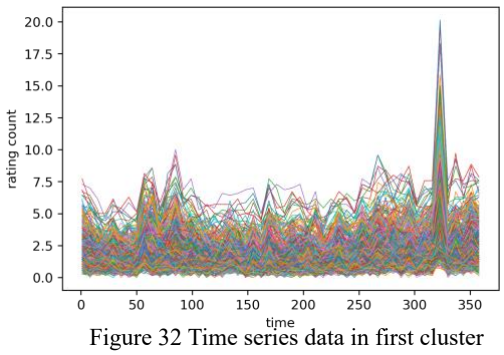

Figure 32 Time series data in first cluster

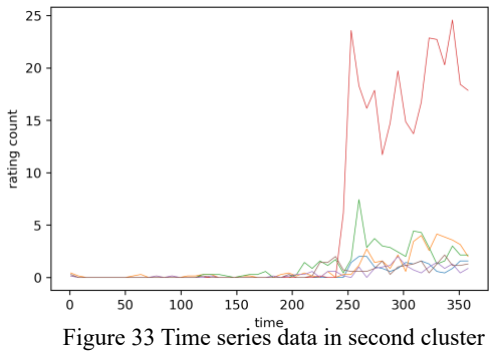

Figure 33 Time series data in second cluster

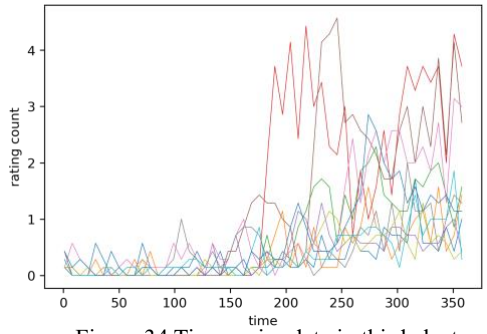

Figure 34 Time series data in third cluster

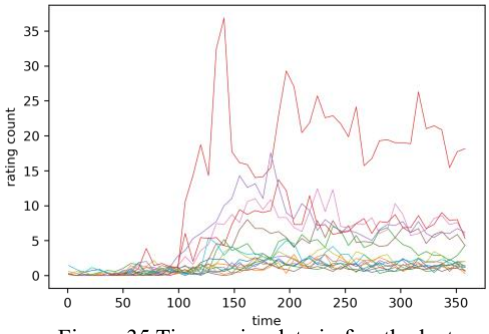

Figure 35 Time series data in fourth cluster

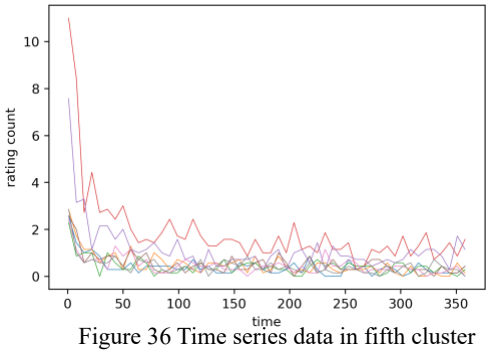

Figure 36 Time series data in fifth cluster

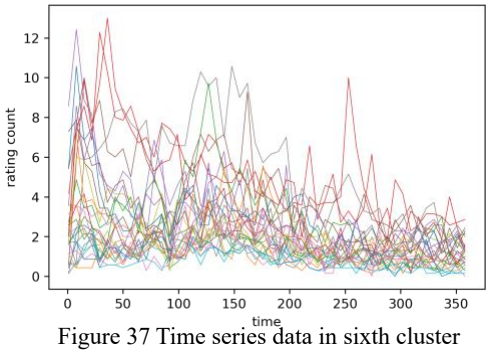

Figure 37 Time series data in sixth cluster

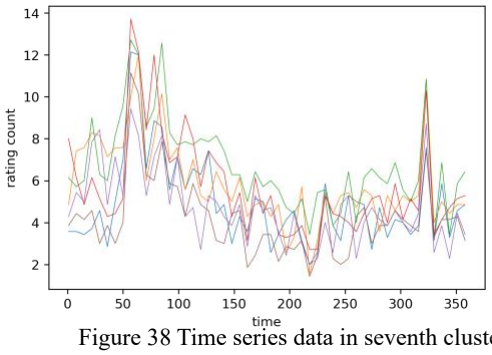

Figure 38 Time series data in seventh cluster

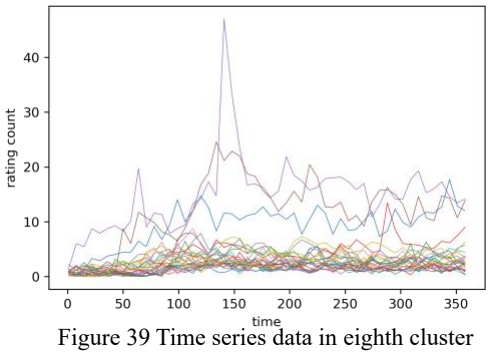

Figure 39 Time series data in eighth cluster

From the clustering results shown in the figures above, it can be concluded that (i) the vast majority of the movies' time series data was clustered into the major cluster (as in Figure 32), (ii) there were some temporal patterns inside the rest seven clusters. For instance, the second, third and fourth clusters (as in Figure 33, 34, 35) contained some movies that had a sudden popularity increase after being released for more than around 100 days, the fifth cluster (as in Figure 36) contained some movies that had a high

popularity right after the release but then the popularity had a sudden drop after a period of time.

However, among all 58594 movies, the clustering performed in this section could only clearly distinguish less than 0.2% of them, which was not sufficient to facilitate the popularity prediction. The vast majority of the movies had a low popularity along the entire 1-year time series, and their popularity also experienced a strong fluctuation. Those reasons made it very difficult to obtain a decent clustering result.

Chapter 5 Conclusion and Future Works

5.1 Conclusion

In this project, a performance analysis on the time-series-based algorithm is performed based on the evaluation which follows the global timeline. By comparing this algorithm with the three baseline methods, this project proves the effectiveness of adopting the idea of time-series forecasting in non-personalized movie recommenders.

Besides, this project also explores the temporal characteristics of the movies' time series data by using clustering technique. The clustering result indicates that there do exist some temporal patterns in the time series data, but it is difficult to use the result for facilitating popularity prediction.

Furthermore, this project calls for more attention on (i) the recommender system's evaluation methodology from the view of global timeline and (ii) the recommender algorithm design from the view of time series forecasting.

5.2 Future Works

5.2.1 Addition on Time-Series-Based Algorithm Design

In this project, the design of the time-series-based algorithm is that a regression model is trained for each individual possible candidate movie. Therefore, it is mathematically not feasible to fine-tune the parameters for every single regression model. In this project, this is done in a way such that after setting the value of th and Δt, all linear regression models share the same set of heuristically chosen parameters, the same goes for SVR with linear kernel, SVR with RBF kernel, ridge regression and Huber

regression. A possible improvement on this could be to choose the regression model's parameters based on the value of th and Δt instead of heuristics.

Another possible future work is to use this time-series-based algorithm as a building block for a more complex recommender system. For instance, something similar to [19] could be done such that the time-series-based algorithm could be used in a post-processing to re-rank the recommendation list.

5.2.2 **Customized Clustering Algorithm**

The clustering performed in this project does not yield a decent result. If a highly customized clustering algorithm could be designed specifically for movie time series clustering, it could be possible to use such clustering algorithm to distinguish the temporal patterns in a larger percentage of movie time series data.

References

[1] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. V. Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston and D. Sampath, The YouTube Video Recommendation System, Proceedings of the fourth ACM conference on Recommender systems (RecSys '10), 2010, p. 293–296.

[2] C. A. Gomez-Uribe and N. Hunt, The Netflix Recommender System: Algorithms, Business Value, and Innovation, ACM Transactions on Management Information Systems, 2015.

[3] F. Ricci, L. Rokach, B. Shapira and P. B. Kantor, Recommender Systems Handbook, Springer Science & Business Media, 2010.

[4] Z. Sun, D. Yu, H. Fang, J. Yang, X. Qu, J. Zhang and C. Geng, Are We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison, 14th ACM Conference on Recommender Systems (RecSys 2020), 2020.

[5] F. M. Harper and J. A. Konstan, The MovieLens Datasets: History and Context, ACM Transactions on Interactive Intelligent Systems, 2015.

[6] Y. Ji, A. Sun, J. Zhang and C. Li, A Re-visit of the Popularity Baseline in Recommender Systems, Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’20), 2020, pp. 1749-1752.

[7] S. Zhang, L. Yao, A. Sun and Y. Tay, Deep Learning Based Recommender System: A Survey and New Perspectives, ACM Computing Surveys, 2019.

[8] M. F. Dacrema, P. Cremonesi and D. Jannach, Are we really making much progress? A worrying analysis of recent neural recommendation approaches, Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19), 2019, pp. 101-109.

[9] M. F. Dacrema, S. Boglio, P. Cremonesi and D. Jannach, A Troubling Analysis of Reproducibility and Progress in Recommender Systems Research, ACM Transactions on Information Systems, 2021.

[10] J. Beel, C. Breitinger, S. Langer, A. Lommatzsch and B. Gipp, Towards reproducibility in recommender-systems research, User Modeling and User-Adapted Interaction, 2016.

[11] J. Lin, The Neural Hype and Comparisons Against Weak Baselines, ACM SIGIR Forum, 2019.

[12] Y. Ji, A. Sun, J. Zhang and C. Li, On Offline Evaluation of Recommender Systems, 2020.

[13] Y. Koren, Collaborative filtering with temporal dynamics, Communications of the ACM, 2010.

[14] J. Tang and K. Wang, Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding, Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18), 2018, pp. 565-573.

[15] P. Cremonesi, Y. Koren and R. Turrin, Performance of recommender algorithms on top-N recommendation tasks, Proceedings of the 2010 ACM Conference on Recommender Systems, 2010.

[16] M. Al-Ghossein, P.-A. Murena, T. Abdessalem, A. Barré and A. Cornuéjols, Adaptive collaborative topic modeling for online recommendation, Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18), 2018, pp. 338-346.

[17] R. Pálovics, A. A. Benczúr, L. Kocsis, T. Kiss and E. Frigó, Exploiting Temporal Influence in Online Recommendation, Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14), 2014, pp. 273-280.

[18] J. Vinagre, A. M. Jorge, C. N. Rocha and J. Gama, Statistically robust evaluation of stream-based recommender systems, IEEE Transactions on Knowledge and Data Engineering, 2019.

[19] M. Jugovac, D. Jannach and L. Lerche, Efficient optimization of multiple recommendation quality factors according to individual user tendencies, Expert Systems with Applications, 2017.

Appendix

The source code used for this project can be found in the following GitHub repository: https://github.com/liguanlong/Final-Year-Project.