UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉNICA SUPERIOR

Máster Universitario en Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos

TRABAJO DE FIN DE MÁSTER

Estrategias de calentamiento en bandidos multi-brazo para recomendación

Author: Esther López Ramos
Advisor: Pablo Castells Azpilicueta

febrero 2021

Francisco Tomás y Valiente, n^o 1
Madrid, 28049
Spain

Esther López Ramos
Estrategias de calentamiento en bandidos multi-brazo para recomendación

Esther López Ramos

PRINTED IN SPAIN

Resumen

Los sistemas de recomendación se han convertido en los últimos años en una pieza esencial de múltiples plataformas online como servicios en streaming o comercios electrónicos ya que permiten a los usuarios encontrar artículos que les puedan resultar de interés ofreciéndoles una experiencia personalizada. El problema de la recomendación cuenta con muchas líneas de investigación abiertas, entre ellas la que abordamos en este trabajo: el arranque en frío.

El problema del arranque en frío en el contexto de los sistemas de recomendación se refiere a la situación en la que un sistema no cuenta con la suficiente información para proporcionar buenas sugerencias a los usuarios. Suele deberse a tres motivos principales: el usuario al que se le recomienda es nuevo en el sistema y por tanto no se conocen sus gustos, algunos de los artículos que se recomiendan han sido añadidos recientemente y no cuentan con reseñas de los usuarios, o el sistema es completamente nuevo y no hay información alguna sobre los usuarios y los artículos.

Las técnicas más clásicas que se aplican a los sistemas de recomendación provienen del ámbito de la Inteligencia artificial y proponen la recomendación como un proceso estático en el que se le ofrecen al usuario una serie de sugerencias y este las valora. No obstante resulta más conveniente entender la recomendación como un ciclo de interacción constante entre el usuario y el sistema en el que cada vez que el usuario puntúa un artículo, el sistema lo utiliza para aprender sobre él. En este sentido se plantea sacrificar el acierto del sistema a corto plazo para obtener información sobre el usuario y mejorar el acierto a largo plazo. Se debe establecer por tanto un balance entre la exploración (recomendaciones no óptimas para obtener conocimiento) y la explotación (recomendaciones óptimas para maximizar el acierto). Las técnicas conocidas como bandidos multi-brazo se utilizan para lograr ese balance entre exploración y explotación y son las que proponemos en este trabajo para abordar el problema del arranque en frío.

Partiendo de la hipótesis de que una exploración en las primeras épocas del ciclo de la recomendación puede ayudar a mejorar el acierto en las épocas más tardías, se ha dividido el bucle de recomendación en dos etapas: la de calentamiento, en la que se persigue una mayor exploración para obtener la máxima información posible sobre el usuario; y la de explotación, en la que se aprovecha el conocimiento adquirido para maximizar el acierto. Para estas dos fases se han combinado distintas estrategias de recomendación, entre las que consideramos tanto algoritmos de bandidos multi-brazo como algoritmos clásicos. Se han evaluado de manera offline sobre tres conjuntos de datos: CM100K (música), MovieLens1M (películas) y Twitter. Se ha estudiado también cómo influye un mayor tiempo de calentamiento sobre la fase de explotación. Los resultados muestran que en dos de los conjuntos de datos (MovieLens y Twitter) las estrategias clásicas obtienen un mayor rendimiento en términos del recall durante la fase de explotación tras una fase de calentamiento mayoritariamente exploratoria.

Palabras clave

Aprendizaje por refuerzo, sistemas de recomendación, bandidos multi-brazo, problema de arranque en frío

Abstract

Recommender systems have become an essential piece of multiple online platforms such as streaming services and e-commerce in the last years as they provide users with articles they may find interesting and thus granting them a personalised experience. The recommendation problem has many opened investigation lines. One of them is the topic we tackle in this work: the cold-start problem.

In the context of recommender systems the cold-start problem refers to the situation in which a system does not have enough information to give proper suggestions to the user. The cold-start problem often occurs because of the following three main reasons: the user to be recommended is new to the system and thus there is no information about its likes, some of the items that are recommended have been recently added to the system and they do not have users' reviews, or the system is completely new and there is no information about the users nor the items.

Classical recommendation techniques come from Machine learning and they understand recommendation as an static process in which the system provides suggestions to the user and the last rates them. It is more convenient to understand recommendation as a cycle of constant interaction between the user and the system and every time a user rates an item, the system uses it to learn from the user. In that sense we can sacrifice immediate reward in order to earn information about the user and improve long term reward. This schema establishes a balance between exploration (non-optimal recommendations to learn about the user) and exploitation (optimal recommendations to maximise the reward). Techniques known as multi-armed bandits are used to get that balance between exploration and exploitation and we propose them to tackle cold-start problem.

Our hypothesis is that an exploration in the first epochs of the recommendation cycle can lead to an improvement in the reward during the latest epochs. To test this hypothesis we divide the recommendation loop in two phases: the warm-up, in which we follow a more exploratory approach to get as much information as possible; and exploitation, in which the system uses the knowledge acquired during the warm-up to maximise the reward. For this two phases we combine different recommendation strategies, among which we consider both multi-armed bandits and classic algorithms. We evaluate them offline in three datasets: CM100K (music), MovieLens1M (films) and Twitter. We also study how the warm-up duration affects the exploitation phase. Results show that in two dataset (MovieLens and Twitter) classical algorithms perform better during the exploitation phase in terms of recall after a mainly exploratory warm-up phase.

KEYWORDS

Reinforcement learning, recommender systems, multi-armed bandits, cold-start problem

Table of Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Goals . . . 2

1.3 Methodology . . . 2

1.4 Document structure . . . 3

2 State of the art **5**

2.1 Recommender systems . . . 5

2.2 Collaborative filtering algorithms . . . 7

2.3 Reinforcement learning and multi-armed bandits . . . 8

2.4 The cold-start problem . . . 10

2.4.1 New item . . . 11

2.4.2 New user . . . 11

2.4.3 New system . . . 12

3 Problem description and algorithms **13**

3.1 Problem description . . . 13

3.1.1 Recommendation as an iterative process . . . 14

3.2 Algorithms . . . 15

3.2.1 Non-personalised algorithms . . . 15

3.2.2 Collaborative filtering algorithms . . . 16

3.2.3 Multi-armed bandit algorithms . . . 18

4 Experiments **25**

4.1 Experimental Methodology . . . 25

4.1.1 Experiment description . . . 25

4.1.2 Evaluation . . . 25

4.1.3 Datasets . . . 26

4.2 CrowdFlower . . . 27

4.2.1 Non-personalised exploit . . . 27

4.2.2 Multi-armed bandit exploit . . . 29

4.2.3 Collaborative filtering exploit . . . 31

4.2.4 Final comparison and wrap-up . . . 32

4.3 MovieLens . . . 33

4.3.1 Non-personalised exploit . . . 33

4.3.2 Multi-armed bandit exploit . . . 35
4.3.3 Collaborative filtering exploit . . . 37
4.3.4 Final comparison and wrap-up . . . 37
4.4 Twitter . . . 39
4.4.1 Non-personalised exploit . . . 40
4.4.2 Multi-armed bandit exploit . . . 40
4.4.3 Collaborative filtering exploit . . . 43
4.4.4 Final comparison and wrap-up . . . 45
5 Conclusions **47**
5.1 Summary and contributions . . . 47
5.2 Future work . . . 48
Bibliography **53**

Lists

List of equations

3.1 Función puntuación o *rating* de un item . . . 13
3.2 Estimación para la recompensa de un ítem . . . 14
3.3 Average . . . 16
3.4 Popularity . . . 16
3.5 User-based KNN . . . 17
3.6 Item-based KNN . . . 17
3.7 ε-Greedy . . . 19
3.8 Inicialización optimista/pesimista . . . 19
3.9 UCB . . . 19
3.10 Thompson Sampling . . . 22
3.11 Update of hits and misses in KNN-bandit algorithm . . . 23
3.12 Item choosing in KNN-bandit algorithm . . . 23

4.1 Recall . . . 26

List of figures

2.1 Diagram of the behaviour of a recommender system . . . 6
2.2 Diagram of the behaviour of a recommender system with a reinforcement learning perspective . . . 9

3.1 Matrix Factorization decomposition . . . 18
3.2 UCB evolution for two different items . . . 21
3.3 Thompson Sampling evolution for two different items . . . 22

4.1 Results of non-personalised algorithms in the exploitation phase for CM100K dataset . 28
4.2 Results of bandits algorithms in the exploitation phase for CM100K dataset . . . 30
4.3 Results of collaborative filtering algorithms in the exploitation phase for CM100K dataset 31
4.4 Comparison of the best combinations of warm-up and exploitation for CM100K dataset 32
4.5 Recall at the end of the warm-up phase for CM100K dataset . . . 33
4.6 Results of non-personalised algorithms in the exploitation phase for MovieLens1M dataset . . . 34

4.7 Results of bandits algorithms in the exploitation phase for MovieLens1M dataset 36
4.8 Results of collaborative filtering algorithms in the exploitation phase for MovieLens1M dataset 38
4.9 Comparison of the best combinations of warm-up and exploitation for MovieLens1M dataset 38
4.10 Recall at the end of the warm-up phase for MovieLens1M dataset 39
4.11 Results of non-personalised algorithms in the exploitation phase for Twitter dataset . . . 41
4.12 Results of bandits algorithms in the exploitation phase for Twitter dataset 42
4.13 Results of collaborative filtering algorithms in the exploitation phase for Twitter dataset 44
4.14 Comparison of the best combinations of warm-up and exploitation for Twitter dataset . . 45
4.15 Recall at the end of the warm-up phase for Twitter dataset 46

List of tables

3.1 Association between recommender system and multi-armed bandits 14

4.1 Datasets description 27

1

Introduction

Recommender Systems can be found today in multiple online applications. They bring value to both customers and owners of a specific platform. They help customers discover new items available in the system giving them a better and personalised experience. They also help item vendors as they promote items to the customer otherwise they would not have known. However when a user is new to the system, a new item is added or even when the system has just been created, we experiment what is called the cold start problem. In such scenarios the system lacks enough information about the users or items and how they link to each other to start providing quality recommendations to the users. In this work we will explore multi-armed bandits strategies to address the cold-start problem in Recommender Systems.

1.1. Motivation

Cold-start is a common concern in Recommender Systems because the effectiveness of most state of the art algorithms relies on having enough information about the items and users. The cold-start problem is still an open question for the research community with several different proposed solutions, so closing the gap between the problem and a good method to mitigate it would potentially impact the field positively.

Furthermore, the use of multi-armed bandits strategies is a relatively new approach to solve the cold-start problem with a much room for improvement and growing interest. Multi-armed bandits are a classic probability theory problem with a clear formalisation and a vast theoretical background [Sutton and Barto, 2018]. We believe multi-armed bandits are convenient for solving cold-start problem because they can start recommending items to the users from scratch, without any previous knowledge about the system, as they continuously learn from the users interactions and update their knowledge. Multi-armed bandits try to find a balance between exploitation and exploration, which is a recurrent problem in computer science and has many applications in the field of information retrieval, for example A/B testing [Scott, 2015, Hill et al., 2017], so the study of multi-armed bandits strategies has many connections with other areas in the field.

1.2. Goals

We define the main goal of studying the effect of warm-up, i.e. the phase in which the recommender system focuses on learning about the users and their likes rather than maximising the number of successful recommendations, in the comparison of bandits algorithms to each other and to other classical algorithms, both non-personalised and collaborative filtering algorithms. To do such task we are going to define the following three different variables to explore:

- Warm-up duration. The goal is to study which algorithms adapt better to less time of warm-up. We are assuming a new system with no previous information of the items nor the users and we want to determine which algorithms are able to perform better recommendations with minimum warm-up time.
- Warm-up strategy. We are going to explore whether multi-armed bandits strategies are useful for the warm-up of a recommender system in terms of the amount of relevant information they are able to gather during the warm-up phase. To do so we are going to analyse how the exploitation phase evolves after the chosen strategy for the warm-up and we are going to compare multi-armed bandits strategies to the non-personalised algorithms in the warm-up phase.
- Post warm-up strategy. For three different domains where recommender systems can be applied (music, films and social networks) we want to determine which is the best strategy after the warm-up has finished, whether is still multi-armed bandits, non-personalised or collaborative filtering algorithms.

We are going to design several offline experiments in which we will vary the three variables defined above to determine which is the best combination of them for every use case scenario (music, films and social networks).

1.3. Methodology

In my BSc degree dissertation [López Ramos et al., 2019] I analysed multi-armed bandits algorithms for recommender systems in depth, and explore the best configuration of its parameters for the three different domains mentioned in section 1.2. As a continuation of that analysis in this work I explore one of the applications of multi-armed bandits to recommender systems, aiming at solving the cold-start problem. To do so,I firstly study the main publications in the literature of recommender systems, multi armed-bandits and cold-start problem. After that, I use the recommender systems software developed by the IR group [Vargas et al., 2016] to build a framework which performs both warm-up and exploitation phases in a given dataset. Once that framework is ready, I prepare three datasets for the experiments and run them offline, simulating an iterative recommendation process. Lastly, I analyse the results of

the experiments and document them.

1.4. Document structure

This document is divided in three chapters, described below:

- In chapter 1, introduction, I explain the motivation of this work, as well as the main goals I will pursue. There is also an explanation of the methodology I followed to achive such goals.
- In chapter 2, state of the art, I summarise the most relevant studies related to this work, starting from recommender systems in general, and then specifying more about collaborative filtering strategies. I will also present the area of reinforcement learning focusing in multi-armed bandits strategies. Finally I will describe the cold-start problem and the most relevant approaches the scientific community have adopted to solve it.
- In chapter 3, problem description and algorithms, I first formalise the task of interactive recommendation, establishing an equivalence between the elements of a recommender systems and the elements of the multi-armed bandit problem. Then I briefly explain the algorithms I will use later in the experiments focusing in how they perform the recommendations.
- For three different use cases (music, films and social networks) I performed offline experiments to determine which configuration of the three variables mentioned in section 1.2 works better for each use case. In chapter 4, experiments, I present the results of those experiments.
- In chapter 5, conclusions, I summarise the results of the offline experiments and present some topics we can still explore in the future as a continuation of this work.

2

State of the art

This chapter focuses on giving context to the problems we are tackling in this work. First, in section 2.1 we describe the problem of recommendation and the main types of recommendation as well as the types of evaluation. In section 2.2 we introduce collaborative filtering algorithms as an example of classic recommendation techniques. In section 2.3 we introduce the paradigm of reinforcement learning and its main lines of research focusing on multi-armed bandits and how we can deal with the task of recommendation from a reinforcement learning perspective. In section 2.4 we introduce the cold-start problem and its types and the main solutions in the literature.

2.1. Recommender systems

The main goal of a recommender system [Ricci et al., 2011] is to improve user experience by suggesting different products available in an online platform. In the last years popularity of recommender systems has increased a lot due to the multiple online services available such as e-commerce platforms, social networks or streaming platforms.

We can define two basic sets in a recommender system: the user set, denoted by U, and the item set, denoted by I. With these two set, the main task of the recommender system is to provide a subset of items, called ranking, to a user ordered descending by a chosen criteria, which often corresponds with an estimation of how useful or interesting the user would find the item. With this recommendation schema the user can easily discover new items.

We can find two different types of recommendations in the literature, explained below:

1. **Personalised recommendation**. It takes user preferences into account when computing the ranking. This results in a different ranking for each user. To do so, every time a user interacts with an item in the system, it stores a numeric value called rating, where the higher the rating the more the user is interested in it. The idea behind the recommendations is to predict the rating of the items that a user has not rated yet and then use that prediction to sort the items descending and present the top n to the user. There are several approaches

to achive personalised recommendation in the literature, the most common ones being **collaborative filtering** [Ricci et al., 2011, chap. 3] and **content-based filtering** [Ricci et al., 2011, chap. 4]. In content-based approaches, the recommender uses item features to compute a similarity function and recommends those unknown items that are most similar to the items the user liked in the past. In collaborative filtering, the system uses past user ratings to compute a simmilarity with other users and recommends the items that the most similar users liked but the user to be recommended does not know. As we are going to face the cold-start problem for some collaborative filtering algorithms, in section 2.2, there is a more detailed explanation of this methods.

2. **Non-personalised recommendation**. Unlike the personalised recommendation, this type of recommendation shows the same ranking to all users without taking into account their past interaction with the items. It may seem trivial but it is specially convenient for the cold-start user problem, that is, when a user is new to the system. Usually it takes past raitings from all users to identify the most popular items in the system.

Figure 2.1 schematically shows the behaviour of a recommender system.

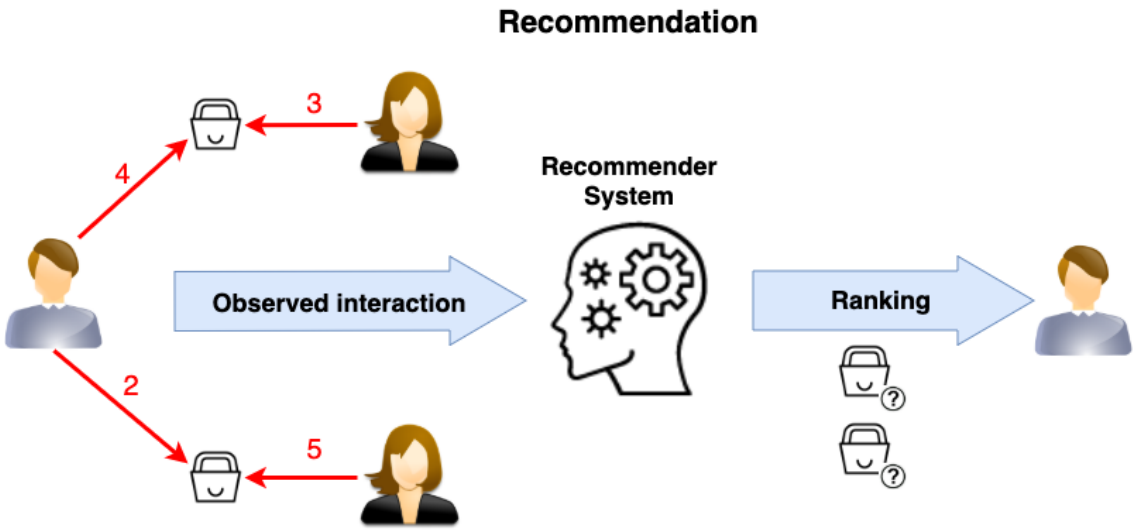

Figure 2.1: Diagram of the behaviour of a recommender system

To assess the efectivity of a recommender sytem there are two types of evaluation: offline and online evaluation.

In the **offline evaluation**, inspired by Machine Learning (ML) supervised learning techniques, we take the user set U and the item set I and some known interactions, called the rating set R. The set of ratings R is divided in two separate sets: the training R_{train} and testing R_{test} sets. The system uses R_{train} to learn, that is, to present the tuples $(u, i, r(u, i))$, where $(u, i) \in U \times I \subset R_{train}$ and $r(u, i) \in \mathbb{R}$ to the chosen algorithm so that it can learn the users likes. The rest of pairs (u, i) not present in R_{train} are the ones that the recommender algorithm has to predict the value of $r(u, i)$ for. We will denote this prediction as $\hat{r}(u, i)$ from now on. With these predictions, the algorithm is evaluated using R_{test} as

follows: we take the value of $r(u,i)$ from those pairs $(u,i) \in R_{test}$ and compare it with the estimation $\hat{r}(u,i)$ of the recommender algorithm using different metrics such as precision or recall [Ricci et al., 2011]. When a pair (u,i) is not present in R_{test} we can ignore this rating for the chosen metric as we do not know the real value or we can assume the real value of $r(u,i)$ to be zero (the user u does not like the item i), and consider it in the chosen metric.

Online evaluation is more complicated to perform but it gives more realistic results as it evaluates the recommender algorithm in a production environment. To do so, it needs a reliable platform with enough users, items and interactions so that the results significantly represent the peformance of the system. Furthermore, testing a new algorithm can negatively impact the experience of the users, so to avoid this the traffic of the system is usually divided into the number of different variations of the algorithm that we are going to try plus the current production version. This is scheme is referred in the literature as A/B testing [Kohavi and Longbotham, 2017].

Usually these two types of evaluation are combined together to get a better evaluation of the performance of the algorithm. First, we use offline evaluation techniques and compare offline metrics with the offline results of the current algorithm in production. If these metrics improve the metrics of the current algorithm, then an online evaluation experiment is conducted to compare both algorithms in the production environment.

2.2. Collaborative filtering algorithms

Collaborative filtering algorithms are a group of personalised algorithms that use past interactions of the users with the items to build the ranking. Among these techniques there are two different approaches: the neighbourhood approach and the latent factor models [Ricci et al., 2011, chap. 3].

The **neighbourhood approach** uses past user ratings to compute a similarity function. This similarity measure can be defined over the items (item-based approach) or over the users (user-based approach). In the item-based approach the system recommends the unknown items (items that the user has not rated yet) that are most similar to those that the user has already liked. In the user-based approach, the system determine the most similar users to a given user and recommend the top rated items by these users (unknown by the user to be recommended). The most common model of this type of recommendation is the K-Nearest Neighbours (KNN) which can be both user-based or item-based. As we will study how the different warm-up strategies affect the user-based KNN algorithm, we will describe this model in chapter 3.

In the **latent factor approach**, the systems uses past ratings from users to items to represent both item and users in the same vector space as latent factors. The goal of this techniques is to extract some abstract features of both items and users. These features are usually computed by representing the user to item ratings as a matrix and then factorising it. Of these kind of algorithms we will study the

Matrix Factorization algorithm in the context of the warm-up problem so we will deeper describe this algorithm in chapter 3.

Both types of collaborative filtering described above need a training phase, where some known ratings are presented to the recommender system and it learns them so that it can produce future recommendations. The cold-start problem which we are tackling in this work appears when there are not enough known ratings for the training of the recommender. In the experiments of chapter 4, we will assume a new system with no previous knowledge of the interactions of users with the items and we will consider the recommendation as a interactive cyclic process where every feedback the system gets from the user is instantly incorporated in its knowledge base, so we will skip the training step and study different warm-up strategies instead. More details of this cyclic interactive view of the recommendation can be found in section 2.3 and details about the cold-start problem in section 2.4.

2.3. Reinforcement learning and multi-armed bandits

Inside the field of machine learning, reinforcement learning is a paradigm that formalise the natural idea of an agent that learns by interacting with its environment. Every time the agent performs and action, this action yields a positive or a negative response. This response allows the agent to learn.

Classic machine learning algorithms focus on maximising cumulative reward. In reinforcement learning, that goal is still considered but changing the way to reach it by finding a balance between **exploration** and **exploitation**. In the reinforcement learning context, exploration implies that the agent makes an action whose consequences are unknown sacrificing the reward with the goal of learning. Meanwhile exploitation means taking the known optimal action to get an immediate reward.

The agent interacts with the environment in discrete steps by choosing an action among all possible ones. Every interaction with the environment yields a reward and changes the state of the agent. Usually the agent is able to observe the whole environment, however in some contexts only partial information about the environment is available. This process of continuous interaction with the environment usually lasts for a long period of time and thus a sacrifice of immediate reward is assumed so that a better long time cumulative reward is achieved.

In recommender systems there is a constant interaction between the user and the items and thus between the user and the ranking that the recommender system builds. In this context, the interactive process of reinforcement learning is very convenient for the recommender systems because the feedback from the user is the input of the system to improve future recommendations. Traditional approaches of recommender algorithms consider a static approach where the system is trained with some known ratings and then the unknown ratings are predicted. With a reinforcement learning perspective, the recommendation becomes cyclic and the immediate hit in the recommendations is sacrificed with the goal of improving long term recommendations. In the context of reinforcement learning, the process

described in figure 2.1 is transformed following figure 2.2.

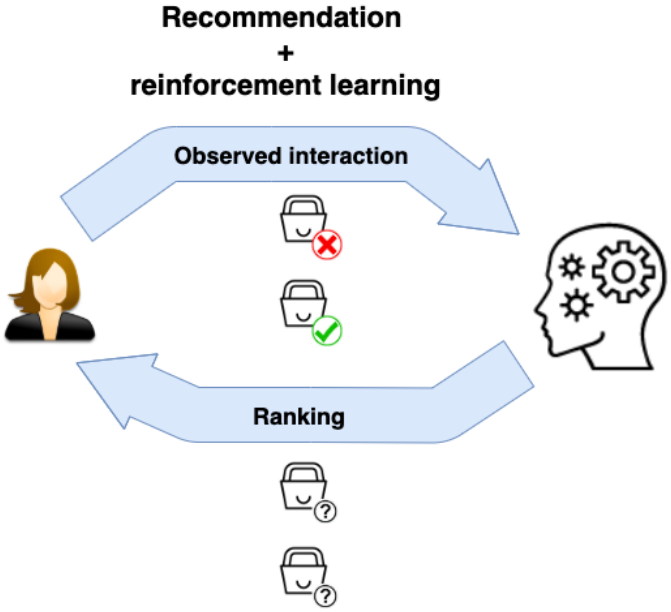

Figure 2.2: Diagram of the behaviour of a recommender system with a reinforcement learning perspective

In this work we will focus on the cold-start problem applying reinforcement learning techniques. In the cold-start problem (more details on section 2.4), the system does not have enough information about the users, the items or both. To mitigate this problem usually some systems directly ask to the user about their preferences, but it is not possible to show all items available in the system to the user. The balance between exploration and exploitation that reinforcement learning presents, can be useful for solving this scenario.

There are two main lines of research in the field of reinforcement learning: Markov decision processes (MDP) [Shani et al., 2005] and multi-armed bandits (MAB) [Sutton and Barto, 2018].

A **Markov decision process (MDP)** is a tuple (S, A, P, R) where S is a finite set of states, A is a finite set of actions, P is a probability function that given two states $s_1, s_2 \in S$, and an action $a \in A$, it assigns a probability to transit from state s_1 to state s_2 by performing the action a, and R is the reward associated with the transition from state s_1 to s_2 by the action a. In the context of recommender systems, each state represents the information that the system has at a time, each action corresponds with recomending an item and thus the reward associated with the transition means the hit or miss of the recommendation. Both the probability function P and the rewards R are known each time the system chooses an action. The goal of the agent (recommender system) is to maximise cumulative reward, and to do so, it observes its environment and chooses an action (recommends an item) and it transitions to another state with a reward. This process is repeated and the successive actions are aggregated. The problem is solved by giving the sequence of actions that produces the biggest possible reward. Article [Shani et al., 2005] proposes a policy iteration algorithm to solve de MDP. In this algorithm each possible solution is a policy which assigns each state an action. There are two phases in the algorithm.

In the first phase every policy is evaluated which represents the exploratory side of the reinforcement learning paradigm. In the second phase, the exploitation, the system chooses the best policy from step one and tries to improve it. Both steps are iterated until convergence.

Multi-armed bandit problem is inspired by a gambler in front of some slot machines who has to choose at each time which machine to play in order to get the best possible reward. In a formal definition [Sutton and Barto, 2018, chap. 2], an agent has to choose between a set of actions known as arms. Every time the agent chooses an action it yields a number known as the reward which the agent does not know until it chooses the arm. The value associated with an arm is not the same every time the agent plays that arm but it follows a probability distribution. The problem of multi-armed bandits is solved by giving the sequence of arms that the agent must play to get the best possible reward, which translates to finding the arm whose distribution of reward has the highest mean. All of the proposed solutions to multi-armed bandit problem are an approximation and they focus on finding a balance between exploration and exploitation. In this context, exploitation means choosing the arm that the agents believes to be the best at the time, while exploration consists on choosing another different arm to gain some knowledge about the chosen action. The most popular algorithms to solve the multi-armed bandit problems are ε-Greedy [Sutton and Barto, 2018, chap. 2], Upper Confidence Bound (UCB) [Auer et al., 2002] and Thompson Sampling [Chapelle and Li, 2011], which we will further discuss in section 3.2.

All the described elements from a multi-armed bandits problem (agent, arms and rewards) must be translated to the elements of a recommender system (users, items and ratings). Most of the times in the literature each item is associated with an arm, so playing an arm means recommending an item as the basic action of a recommender system is to recommend. The reward of an action is the rating that the user gives to the recommended item. In section 3.1 there is further details on the equivalence of recommender system and multi-armed bandits elements. In the multi-armed bandits problem there are no elements similar to the users of a recommender system. But as the goal of the user is to provide feedback to the system by evaluating its recommendations, we will consider the user as the element which gives context to the bandit. The main difference between the MDP and the multi-armed bandits is that in MDP there are a set of states which represent the information that the agent has at each time while in the multi-armed bandit problem the agent does not have a state. However there are some approaches called context multi-armed bandits [Sutton and Barto, 2018, chap. 2] that consider some features of both the agent and the system. So taking the user feedback as the context means that we will be applying context multi-armed bandits to the field of the recommender systems.

2.4. The cold-start problem

As mentioned in section 2.1, a recommender system needs information about the users, the items and their interactions (ratings) to perform the recommendations, specially in personalised algorithms. There

are some situations in which the recommender systems lacks enough information to choose the items to recommend to the users, this is what we call the cold-start problem. There are three main situations where the cold-start problem happens: a new item, a new user or a new recommender system. We will explain these three situations in subsections 2.4.1, 2.4.2 and 2.4.3 and the main approaches to solve them in the literature.

2.4.1. New item

When an item is new to the system it has no ratings from any user. As we have seen in section 2.2, collaborative filtering algorithms rely on previous ratings from users to compute a similarity between the users. Then, with this similarity function and the past ratings, they can infer the rating over the items that a user has not rated yet. As this new item has no ratings, collaborative filtering algorithms cannot do this inference for this specific item. We can also consider the cold-start problem when an item has very few ratings. In this case, collaborative filtering algorithms can infer the unknown ratings but the resulting ratings would not be realistic.

One of the main approaches to solve this problem in the literature is the use of content-based algorithms [Rohani et al., 2014]. These algorithms use properties or features from the items that are independent to the users. They use the previous ratings from a user to find which items are similar to those that this user has liked in the past using machine learning techniques like classifiers [Lika et al., 2014] or clustering [Puthiya Parambath and Chawla, 2020].

Although these techniques have proven to mitigate the cold start problem in the previously mentioned studies, they have some drawbacks. Content-based algorithms tend to over-specialise on the user preferences as they recommend items that have similar features to those which that user liked. This leads to a poor recommendation, not introducing novel content to the user [Pereira and Varma, 2016]. They also rely on the items features which are not available in all contexts.

2.4.2. New user

When a user is new, the recommender system has to suggest items without any knowledge, as the user has not interacted with any item yet. This scenario affects both collaborative filtering and content based algorithms. In the case of collaborative filtering, as this user has not rated any item, algorithms cannot establish a similarity with other users and thus they cannot infer any rating. In the case of content based algorithms, they cannot choose items similar to those that the user likes because they have no preferences from the user.

Recommender systems play a key role when a user is new to the system because they can help them discover the items available, so dealing with the cold-start user problem is important to provide users a better experience. Some studies in the literature propose directly querying the user to learn their likes [Sun et al., 2013]. These techniques can negatively affect the user experience and the rated items could be not representative of the user current likes. Other authors suggest a simmilar approach to content-based algorithms establishing a user profile using external features like demographic data [Safoury and Salah, 2013]. However this information may not be available or may not be related to the user preferences. Other approaches include using non-personalised methods like showing the user the most popular items in the system but this can lead to a bias in the most popular items not introducing variability in the recommendations [Rashid et al., 2008].

2.4.3. New system

This is the most challenging scenario as it means the whole system is new so it has no information about the users, items and their interactions. It combines both scenarios of new user and new item simultaneously. This is the scenario we will address in this work, as we are going to assume no previous information about the users nor the items of any kind.

We will compare different approaches using non-personalised recommenders and multi-armed bandits to solve the new system cold-start problem. More details on this proposed solution can be found in chapters 3 and 4.

3

Problem description and algorithms

This chapter presents the methodology we followed to adapt multi-armed bandits algorithms to the context of recommendation. There are two sections in this chapter. In section 3.1 we describe the cold-start problem and translate the recommender task as a multi-armed bandit problem. In section 3.2 we describe the non-personalised, collaborative filtering an multi-armed bandits algorithms we are going to evaluate during the exploitation phase in the case of collaborative filtering and during the warm-up and exploitation phases for non-personalised and multi-armed bandits in the experiments of chapter 4.

3.1. Problem description

In this study, we propose multi-armed bandits as a solution for cold-start problem (see section 2.4 for more details). Multi-armed bandits are very convenient for this task because they do not need previous information about users preferences, because recommendations can begin with an exploratory phase to get knowledge about the system and then continue with an exploit phase. There are many domains where we can apply multi-armed bandits so, before describing the experiments we conducted, we are going to formulate the task of recommendation as a multi-armed bandit problem.
As described in chapter 2 a recommender system consists of a set of users, denoted by U and a set of items, denoted by I. For each pair (u, i) where $u \in U$ and $i \in I$ we define the function rating, r, as in equation 3.1.

$$
\begin{aligned}
r\colon R \subset U \times I &\to \{0, 1\} \\
(u, i) &\to r(u, i) = \begin{cases} 1 & \text{if } u \text{ likes } i \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{3.1}
$$

where R is the set of ratings, that is, pairs (u, i) where $r(u, i)$ is known. The task of the recommender system is to determine those ratings that are unknown by estimating them. From now on we will denote this estimation by $\hat{r}(u, i)$.

As the basic action for a recommender system is to choose an item, in most of the algorithms we

have decided to consider each arm of the bandit as an item, except from the KNN-bandit, where the bandits are the users to be chosen as neighbours (more details on section 3.2.3). The immediate reward associated to an arm will be the value of the rating $r(u, i)$. As a consequence, the value associated to each arm will be the estimation $\hat{r}(u, i)$. The user represents the context in the multi-armed bandit problem and it will have two basic functions: it will provide feedback each time an arm is chosen and will determine which arms the system cannot choose anymore, as we introduce the restriction that we can recommend the user an item once.

Table 3.1 shows a summary of the association between the elements of a recommender system and the elements of the multi-armed bandits problem.

Recommender system	Multi-armed bandit
item	arm
average rating	value of an arm
rating	reward
user	context

Table 3.1: Association between recommender system and multi-armed bandits

Usually recommender systems return a set of k items ordered by the estimated relevance as a ranking for each user [Ricci et al., 2011]. Although is possible to return a ranking using multi-armed bandtis strategies [Zhao et al., 2013], in the literature the most common approach is to recommend and item each iteration, which aligns better with the multi-armed bandit paradigm [Wang et al., 2018].

3.1.1. Recommendation as an iterative process

With multi-armed bandits the recommendation becomes an interactive and continuous task. In this process, the user asks the system for a recommendation, the systems answers with an item and in exchange it gets a positive or negative rating back from the user which the system uses to update its knowledge. If the user does not answer to the recommended item, the system also registers that interaction to avoid recommending the item in the future and thus not negatively affecting the user experience, but the information about the item is not updated.

As we previously mentioned in section 2.1 sometimes the system begins with a set of ratings (the training set), which in multi-armed bandits algorithms is used to initialise the value of each arm (item) ($\hat{r}(i)$) to the mean of the ratings given by all users who have rated this item. As in this project we are studying cold-start problem, we will not have a training set and thus we will initialise this value to 0. Each time the system recommends an item to the user and the user answers with a rating, we will update the value of $\hat{r}(i)$ according to equation 3.2. As it is an iterative process, the value of an item changes every epoch so we need the parameter t to capture that temporariness.

$$\begin{cases} \hat{r}_t(i) = \dfrac{\alpha_t(i)}{\alpha_t(i) + \beta_t(i)} = \dfrac{\sum_{u \in \hat{U}_t(i)} r(u,i)}{|\hat{U}_t(i)|} \\ \hat{r}_0(i) = 0 \end{cases} \tag{3.2}$$

where $\hat{U}_t(i) = \{u_k : (u_k, i)$ chosen for $k \leq t\}$, is the set of users to which the system has recommended the item i until epoch t. Equation 3.2 estimates the value associated with an item by dividing the hits ($\alpha_t(i)$) by the total of times the item was chosen, i.e. the sum of hits and misses ($\alpha_t(i) + \beta_t(i)$).

Algorithm 3.1 shows the process we have followed to recommend an item to a selected user and update the information of the bandit.

input : $u \in U$ user to be recommended, I item set, R rating set

```
1 i ← select_arm(u);
2 r ← get_reward(u, i, R);
3 update_arm(i, r);
```

Algorithm 3.1: Algoritmo general de recomendación con bandidos

The chosen multi-armed bandit algorithm determines the policy to select the item or arm. The arm is updated by the formula 3.2.

3.2. Algorithms

In this section we will present the algorithms that we will combine in chapter 4 during the warm-up and exploitation phase. To do so we will divide this section in two subsections. In subsection 3.2.2 we explain the chosen collaborative filtering approaches K-Nearest Neighbours (KNN) and Matrix Factorization (MF) while in subsection 3.2.3 we will describe multi-armed bandits algorithms: ε-greedy, Upper Confidence Bound (UCB), Thompson Sampling and KNN Bandit.

3.2.1. Non-personalised algorithms

As we previously described in section 2.1 non-personalised algorithms offer the same recommendation to all users because the value they assign to each item does not depend on the user that requests the recommendation. Non-personalised approaches are often used during the warm-up of a recommender system to start knowing about the user likes. In this subsection we briefly explain the random, average and popularity approaches as we are going to compare them to the rest of algorithms in the experiments chapter 4.

Random

The random approach is the most simple recommendation we could choose. Every time a user request an item the systems selects a random item between all possible excluding those items that the system has previously recommended to the user (beacause we do not recommend the same item twice to the same user) or those items that the user has rated before. It is equivalent to the 0-greedy algorithm, i.e. take $\varepsilon = 0$ in equation 3.7 for the ε-greedy algorithm. It is often used as the baseline for the rest of the algorithms to evaluate as we consider the random approach to be the worst a recommended system can perform.

Average

In the average algorithm the system computes the average rating from all the ratings an item has received according to equation 3.3. The system uses this values to sort the items to build the ranking for the user. As we are recommending a single item, we recommend the item that maximises the value of equation 3.3. This approach is equivalent to the 1-greedy algorithm, i.e. take $\varepsilon = 1$ in equation 3.7.

$$\hat{r}(u,i) = \hat{r}(i) = \frac{\sum_{u \in U_i} r(u,i)}{|U_i|} \tag{3.3}$$

where $U_i = \{u \in U | (u,i) \in R\}$, i.e. the subset of users that have rated the item i.

Popularity

In the popularity approach, the recommender system sorts the items decreasingly by the number of positive ratings each item has received. This is similar to the average approach but without normalising the number of positive ratings by the total of ratings received by the item (see equation 3.4). The value of $\hat{r}(u,i)$ in this approach is not an approximation of $r(u,i)$ because without the normalisation the value computed by the popularity algorithm can be arbitrarily big and not in $[0,1]$ interval.

$$\hat{r}(u,i) = \hat{r}(i) = \sum_{u \in U_i} r(u,i) \tag{3.4}$$

where $U_i = \{u \in U | (u,i) \in R\}$, i.e. the subset of users that have rated the item i.

3.2.2. Collaborative filtering algorithms

Collaborative filtering algorithms offer personalised recommendation to the user based on the previous ratings that the users gave. In this section we will describe the K-Nearest Neighbour and Matrix Factorization methods which are the collaborative filtering algorithms we will explore in the experiments chapter 4.

K-Nearest neighbours

The idea behind the K-Nearest Neighbours (KNN) [Ricci et al., 2011, chap. 2.3] algorithm in recommender systems is that similar uses prefer similar items. In a recommender system, each user has rated a set of items and we want to find out what other items that the user has not rated yet could like. If we want to recommend the item i to the user u we can proceed by two different approaches listed below:

1. In the **user-based** approach, the system finds the top k most similar users to u according to a similarity function. This is what we call the neighbourhood of u, $\mathcal{N}(u)$. Then every user $v \in \mathcal{N}(u)$ "votes" for the item i using the rating that v has previously given to i (if it exists, if not we consider it as 0) weighted by the similarity between u and v. This schema gives an estimation or prediction of the rating that u would potentially give to the item i, what we previously denoted as $\hat{r}(u,i)$. Then all the items that can be recommended to u are sorted by these estimations and presented to the user as a ranking. Equation 3.5 describes how to predict the rating given to i by u.

$$\hat{r}(u,i) = \frac{\sum_{v \in \mathcal{N}_k(u)} sim(u,v) r(v,i)}{\sum_{v \in \mathcal{N}_k(u)} |sim(u,v)|} \tag{3.5}$$

2. In the **item-based** approach, instead of finding the neighbourhood of u, the system computes the neighbourhood of i, $\mathcal{N}_k(i)$ i.e. the top k items most similar to i. Then the system takes the ratings that u has given to the neighbours of i weighted by the similarity of i with its neighbours to estimate the rating that u would give to the item i. As in the user based approach, the items are ranked according to these estimations. Equation 3.6 describes how to compute the estimated rating from u to i.

$$\hat{r}(u,i) = \frac{\sum_{j \in \mathcal{N}_k(i)} sim(i,j) r(u,j)}{\sum_{j \in \mathcal{N}_k(i)} |sim(i,j)|} \tag{3.6}$$

In equations 3.5 and 3.6, $r(a,b)$ denotes the value of the rating that the user a has given to the item b. The function sim is the similarity between the users in equation 3.5 or the items in equation 3.6. There are different similarity functions in the literature, the most common ones being cosine similarity [Ricci et al., 2011, page 124, eq. 4.4] and Pearson correlation [Ricci et al., 2011, page 230, eq. 7.5].

In our formulation of cyclic and interactive recommendation we will not produce a ranking for the user u but we will recommend the item i that maximises the estimation of $\hat{r}(u,i)$.

Matrix Factorization

Matrix Factorization models (MF) [Ricci et al., 2011, chap. 3.3] aim at finding latent features using the ratings from the users to the items. In this context the ratings are represented as a matrix and this matrix is decomposed in two lower density matrices. Once this decomposition is done, every user u is associated with a vector $q_u \in \mathbb{R}^k$ and every item i is associated with a vector $q_u \in \mathbb{R}^k$ (where k is the chosen number of latent factors. Intuitively if a user has a high value in a chosen latent factor that means that she would be interested in the items that also have a high value in the same latent factor. One the representation of the users and items as vectors is computed, we use the dot product to estimate the rating from user u to item i: $\hat{r}(u,i) = q_u^T p_i$. Figure 3.1 schematically represents how the rating matrix decomposes to the user latent factor matrix and the item latent factor matrix.

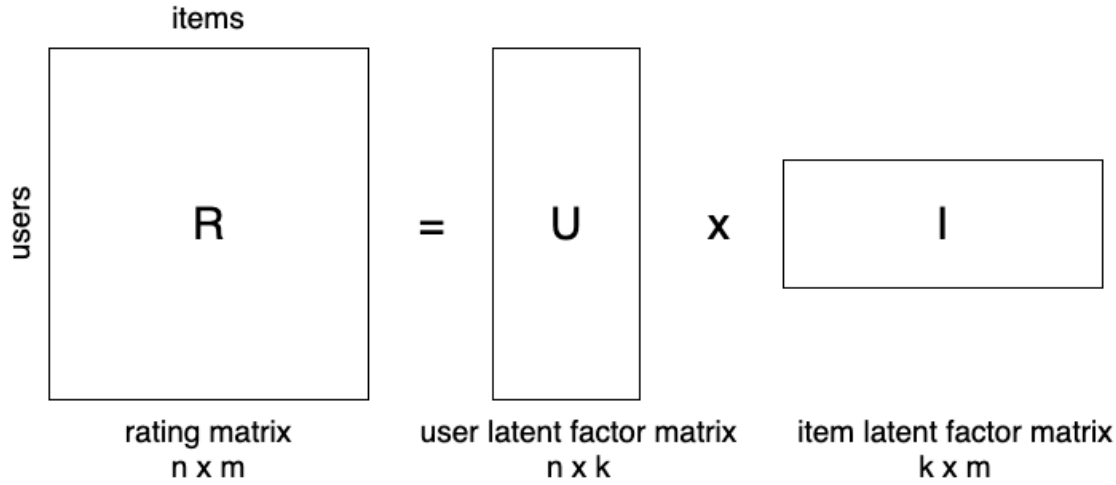

Figure 3.1: Matrix Factorization decomposition of the rating matrix in two lower density matrices. The rows of matrix U are the representation of users in the latent factor subspace while the columns of matrix I are the representation of the items in the latent factor subspace.

The key question in MF algorithms is how to decompose the rating matrix. A Singular Value Decomposition (SVD) is undefined in this context because the rating matrix is not square and it is very sparse which means that there are a lot of missing ratings. Furthermore, the rating matrix is huge so it is computationally expensive to get the SVD. Other methods have been proposed to learn the latent factors such as solving the least squares problem [Paterek, 2007].

3.2.3. Multi-armed bandit algorithms

There are a lot of multi-armed bandits algorithms that can be adapted to the recommender task we are tackling in this work. The main difference between these strategies is how they balance exploration and exploitation.

In this subsection we will describe the algorithms of ε-Greedy, Upper Confidence Bound (UCB) and Thompson Sampling as non-personalised bandits algorithms because they consider the same

value of each arm for all users. To get personalised recommendations each item or arm should keep a different value for every user. There are in the literature some examples of personalised bandits [Zhao et al., 2013, Wang et al., 2018, Kawale et al., 2015, Gentile et al., 2014, Li et al., 2010], however we will consider the KNN-bandit as our example of personalised multi-armed bandit recommendation.

ε-Greedy

In ε-greedy algorithm [Sutton and Barto, 2018] the balance between exploration and exploitation is very simple: we fix a real number $\varepsilon \in [0, 1]$ and with probability ε the system uniformly chooses a random item or arm and with probability $1 - \varepsilon$ the system chooses the item that has maximum $\hat{r}(i)$. The exploration side of the algorithm correspond with the random choosing while the exploitation consists on choosing the most popular item to the system knowledge. We summarise this behaviour in equation 3.7.

$$i_t = \begin{cases} i \in I \text{ uniformly random }, & 1 - \varepsilon \\ \underset{i \in I}{\operatorname{argmax}}[\ \hat{r}_t(i)\ ], & \varepsilon \end{cases} \tag{3.7}$$

The value of $\hat{r}_0(i)$ (see equation 3.2) is initialised to 0 for every item. We can choose different values to initialise this constant such as choosing an initial number of hits α_0 and misses β_0 and compute the ratio between the hits and the sum of the hits and misses as in equation 3.8.

$$\hat{r}_0(i) = \frac{\alpha_0}{\alpha_0 + \beta_0} \tag{3.8}$$

We call this type of initialisation optimistic when the number of hits is greater than the number of misses $\alpha_0 > \beta_0$ and pessimistic when the number of misses is greater than the number of hits $\alpha_0 < \beta_0$. However as in this project we are evaluating the performance of multi-armed bandits algorithms in the cold-start problem, we will not explore the optimistic or pessimistic initialisation. More details of this type of initialisation can be found in [López Ramos et al., 2019, chap. 4.3].

Upper Confidence Bound

To do a balance between exploration and exploitation, the UCB algorithm [Auer et al., 2002] takes the popularity based recommendation or greedy approach as the exploitation part of the algorithm and it adds an exploratory term. Every epoch the algorithm select the items that maximises 3.9.

$$i_t = \underset{i \in I}{\operatorname{argmax}} \left[\hat{r}_t(i) + \sqrt{\frac{\gamma\ ln(t)}{\alpha_t(i) + \beta_t(i)}} \right] \tag{3.9}$$

As we said before, the first term of the equation 3.9 is the value of each arm which in this context of recommender systems corresponds with the reward estimation of an item.

The second term is the exploratory part of the recommendation and without this term the UCB

algorithm would be a greedy and trivial approach. This term is often called uncertainty. The γ parameter is a constant that the higher it is set, the more importance the algorithm gives to the uncertainty. We can see in the numerator the variable t which is the current iteration of the algorithm. The logarithm smooths this variable. In the denominator there is the sum of the hits and misses of the item i, which is the total number of times i was selected until epoch t. The more i is recommended, the higher this aggregation is and thus the lesser is the uncertainty. This fits with the intuitive idea of uncertainty: the more we recommend an item, the more certain we are of the value of $\hat{r}_t(i)$. In the first epochs of the recommendation loop, the value of the uncertainty is higher and similar to all the items so the UCB algorithm tends to a more exploratory behaviour in the first iterations. An item that has not been recommended as much as the other items tends to have a higher value of the uncertainty (because the numerator is the same for all the items) and this allows the algorithm to explore this items.

At the start of the recommendation loop, when $t = 0$, the value of the uncertainty is undefined and thus the UCB algorithm needs an initialisation. This initialisation consists on recommending all the items at least once to get either a hit or a miss and set the value of the uncertainty. After the initialisation $t = |I|$ because we have performed as much epochs as items, $\hat{r}_t(i)$ would be 1 or 0 depending on whether we hit or not the recommendation and the same for $\alpha_t(i)$ and $\beta_t(i)$.

Figure 3.2 shows the evolution of UCB algorithm for two different items at the beginning of the recommendation loop, when half of the epochs are reached and at the end of the recommendation.

Thompson Sampling

In the multi-armed bandit paradigm the gambler must determine the arm that yields the best reward as soon as possible. A complete greedy approach would always choose the arm with the highest observed value. Multi-armed bandits add an exploratory part to that greedy approach which captures the uncertainty of the value of each arm. In the ε-greedy approach we have seen that this exploration is very simple and it just adds a rate for a random selection. In UCB, the uncertainty term is added to the value of each arm so the less explored items (arms) that have a higher uncertainty are selected more until the system is more certain about the value. In Thompson Sampling [Chapelle and Li, 2011] the uncertainty is captured by introducing a distribution on the value associated to each item. This distribution is computed based on the current knowledge of the system of the hit and misses in the past recommendations. Each item or arm has its own distribution and the system samples a value for every item using its distribution and then chooses the item with the highes sampled value.

In equation 3.1, r can only be either 0 or 1 so we can assume that r follows a Bernoulli distribution with parameter p_i, i.e. it acts like a coin with p_i probability of head. When an item is recommended t times the system counts how many time it has hit the recommendation $\alpha_t(i)$ (or how many times the value of r was 1), and how many times it failed the recommendations $\beta_t(i)$ (or how many times $r = 0$). We describe parameter p_i as a Beta distribution with parameters $\alpha_t(i)$ and $\beta_t(i)$ which is the

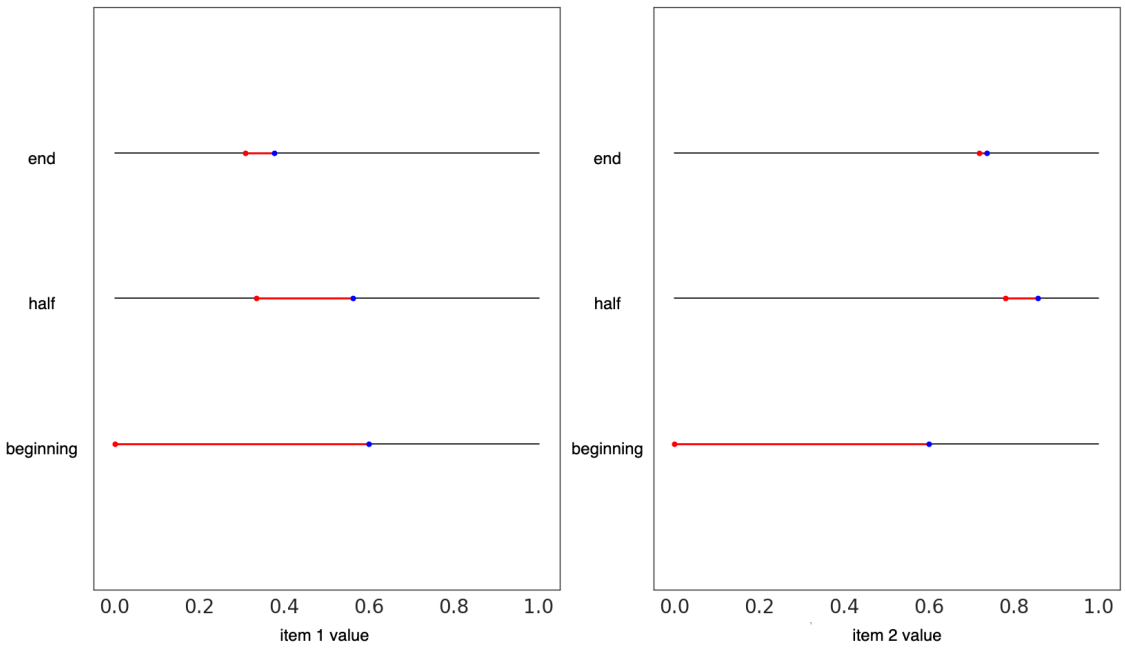

Figure 3.2: UCB evolution for two different items. The red dot represents the value of $\hat{r}_t(i)$, the exploitative side of the algorithm. The blue dots are the sum of $\hat{r}_t(i)$ and the uncertainty so the algorithm chooses the item that maximises the blue value each iteration. Initially every item has the value of the red dot fixed to 0. Initially the value of the uncertainty is very similar to all the items and it is the result of the initialisation, so the first recommendations are random. When the algorithm progresses the uncertainty value decreases for the most popular items and thus the red and blue dots get closer. We can see that the second item is more popular than the first one not only because the value of $\hat{r}_t(i)$ is higher, but because the blue and the red dots are closer for this item than for the first one.

conjugate distribution of the Bernoulli. The more an item is recommended, the higher is the value of $\alpha_t(i) + \beta_t(i)$ and thus the Beta distribution stretches more around its mean, $\hat{r}_t(i) = \frac{\alpha_t(i)}{\alpha_t(i)+\beta_t(i)}$, which intuitively means that the system is more certain about the value of the mean rating for each item. Each epoch the algorithm selects the item according equation 3.10.

$$i_t = \underset{i \in I}{\operatorname{argmax}}[x \leftarrow B(\alpha_t(i) + \alpha_0, \beta_t(i) + \beta_0)] \tag{3.10}$$

$B(a, b)$ represents the Beta distribution with parameters $a,\ b$ and "$\leftarrow$" means choosing a random number. We will initialise the parameters of the Beta distribution to 1 and initially all the distribution will be uniform. We can also choose an initial number of hits α_0 and misses β_0 and perform an optimistic initialisation ($\alpha_0 > \beta_0$) or a pesimistic initialisation ($\alpha_0 < \beta_0$).

Figure 3.3 shows the behaviour of the Thompson Sampling algorithm for two different items at the beginning, half and end of the recommendation loop.

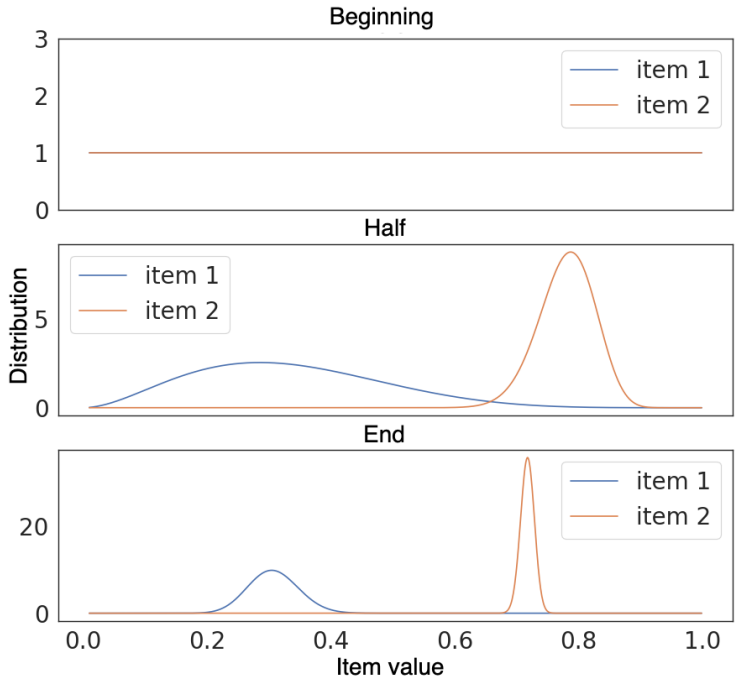

Figure 3.3: Thompson Sampling evolution for two different items. Initially both items start with an uniform distribuition of their respective average reward. In the half plot we can see that item 2 is more popular than item 1 not only beacause the Beta distribution is centered around a higher value but also beacause the distribution is stretcher. This behaviour is reinforced in the end plot.

K-Nearest Neighbours Bandit

The K-Nearest Neighbours Bandit (KNN-Bandit) [Sanz-Cruzado et al., 2019] is the only chosen multi-armed bandit algorithm that performs personalised recommendations. This algorithm is based on the collaborative filtering K-Nearest Neighbours (subsection 3.2.2) algorithm. Unlike the other bandit stra-

tegies we described before (ε-greedy, UCB and Thompson Sampling) in KNN-bandit the arms to be chosen by the agent are not the items to be recommended but the users. When a user u ask for a recommendation, the KNN-bandit algorithm selects another user v as the neighbour of u like in KNN algorithm but following a multi-armed bandit approach, i.e. balancing exploration and exploitation. Once the system has selected the neighbour v, it recommends the item that v likes the most. The chosen multi-armed bandit algorithm to select the neighbour is Thompson Sampling so for each user we have to determine how to update the α, number of hits, and β, number of misses, of the algorithm. Equation 3.11 shows how the Thompson Sampling parameters are updated.

$$\begin{cases} \alpha_t(u,v) & = \alpha_{t-1}(u,v) + r(v,i) * r(u,i) \\ \beta_t(u,v) & = \beta_{t-1}(u,v) + r(v,i) * (1 - r(u,i)) \end{cases} \tag{3.11}$$

where i is the item selected by the neighbour v. Intuitively, we add a hit when both u and v liked the item i and a miss when either u or v or both disliked the item i. As this is a personalised recommendation the system needs to store the values of the hits and misses for every pair of users but only once as it is symmetrical.

This schema of choosing a single neighbour can be generalised to k neighbours by choosing the top k users that maximise the values drawn from the Beta distribution in the Thompson Sampling algorithm, the neighbourhood of u, $\mathcal{N}_u$. Then each neighbour votes for its favourite item by weighting it with the Thompson Sampling value as in equation 3.12.

$$i_t = \underset{i \in I}{\operatorname{argmax}}[x_{u,v} * r(v,i) | v \in \mathcal{N}_u] \tag{3.12}$$

where $\mathcal{N}_u$ is the top k neighbours of u and $x_{u,v}$ are the values drawn from the Beta distribution in the Thompson Sampling algorithm $x_{u,v} \leftarrow B(\alpha_t(u,v), \beta_t(u,v))$.

4

Experiments

4.1. Experimental Methodology

4.1.1. Experiment description

The main question we are going to answer in this work is “given a context where we want to use a recommender system, which is the best configuration of the system in terms of warm-up algorithm, exploitation algorithm and warm-up duration?” So we define three different variables we explore in the following experiments: warm-up algorithm, exploitation algorithm and warm-up time. To find the best configuration we proceed as follows: for a fixed exploitation algorithm, we vary the warm-up algorithm and evaluate how warm-up affects the exploitation phase for different durations of the warm-up. We aevaluate this process in three different datasets of three different domains to determine which configuration is best for each domain.

4.1.2. Evaluation

The ideal way of conducting this kind of experiments would be to do an online A/B test in a real production environment to evaluate how the different configurations of the recommender system would affect customers. However, as we do not have such a platform we proceed with offline evaluations.

As we want to explore how changing the warm-up algorithm and its duration affects the performance of the recommender system, we will assume the recommender system starts with no previous knowledge about the users and the items. We will acknowledge the nature of the recommendation as an iterative process, and we will divide this process in two phases:

1. **Warm-up**. For this phase we choose a recommender algorithm and a duration in terms of number of epochs. We run the algorithm iteratively until we meet the number of epochs chosen and we save the (user, item, rating) tuples discovered in this phase. If the algorithm chooses a pair (user, item) which is not present in the dataset, we will ignore it. After this phase we will have a subset $T \subset I \times U \times R$ which we will call *train set* from now on.

2. **Exploitation**. We choose another algorithm for this phase (which can be the same as the previous one). This algorithm "learns" from the discovered tuples in the train set T. We will also run this algorithm iteratively, but this time until the whole dataset is discovered by the exploitation algorithm. Each time the system makes a recommendation, we compute recall which we define in equation 4.1. Recall computes the ratio between number of ratings discovered at time t and the total number of ratings to be discovered.

$$recall(t) = \frac{\sum_{k=1}^{t} r(u_k, i_k)}{|\{(u,i) : r(u,i) = 1, u \in U, i \in I\}|} \tag{4.1}$$

where $u_k,\ i_k$ is the pair (user, item) chosen for recommendation at the k-th epoch (i.e. i_k is recommended to u_k at epoch k).

Algorithm 4.1 details the general process we follow in the experiments.

input : U user set, I item set, R *ratings* set, A_w warm-up algorithm, A_e exploitation algorithm, N warm-up epoch
output: $recall$ acumulado

```
recall ← 0;
K ← ∅;
for epoch ∈ {1 ··· N} do
    u ← select_random_user();
    i ← recommend_item(A_w, u);
    r ← get_reward(u, i, R);
    update_knowledge(K, u, i, r);
    remove_rating(R, u, i);
end
while R ≠ ∅ do
    u ← select_random_user();
    i ← recommend_item(A_e, u);
    r ← get_reward(u, i, R);
    update_knowledge(K, u, i, r);
    recall ← update_recall(i, r);
    remove_rating(R, u, i);
end
```

Algorithm 4.1: Warm-up and exploitation for recommendation using multi-armed bandits

4.1.3. Datasets

To conduct the experiments we have selected three different public datasets from different domains. As we are using binary ratings, we have to define a threshold to transform numeric ratings to binary. The datasets are as follows.

1. **CM100K** [Cañamares and Castells, 2018] contains 103.584 anonymous ratings of 1.054 songs. Items with 1 or 2 rating have been treated as 0, that is, the user didn't like the song, while 3 and 4 ratings have been considered as 1 (the user liked the song).

2. **MovieLens1M** [Harper and Konstan., 2015] contains 1.000.209 anonymous ratings of 3 900 films from 6 400 MovieLens system users [movielens, 1997]. Ratings are in a scale of 5 stars so we considered 1 to 3 starts as 0 and ratings from 4 to 5 as 1.
3. **Twitter**: the dataset [Sanz-Cruzado and Castells, 2018] contains a set of interactions (re-tweets and mentions) reflected in the last 200 tweets from 10000 users. In Twitter we can recommend both users and tweets. We have chosen to recommend users based on the interactions. In this new scenario, the items to recommend are the users of the system, so we have to introduce a restriction: if one user u_2 follows another user u_1 but u_1 doesn't reciprocally follow u_2 we are not going to recommend u_1 to follow u_2 back, as it is a very trivial and redundant recommendation. In this dataset there is no 0 rating value because a Twitter user cannot rate negatively another user or tweet.

Table 4.1 summarises the information of the users, items and ratings of the three datasets.

Dataset	Number of users	Number of items	Number of ratings
CM100K	1.000	1.054	103.584
MovieLens1M	6.400	3.900	1.000.209
Twitter	5.000	5.000	43.444

Table 4.1: Summary of the three datasets used for the offline experiments

4.2. CrowdFlower

In this section we will present the results we got after applying the process detailed in algorithm 4.1 to the CM100K dataset. As we mentioned in section 4.1 there are three variables we want to explore in this experiment: the warm-up algorithm, the exploit algorithm and the warm-up size. We will divide this section in three subsections based on the type of the exploitation algorithms. In subsection 4.2.1 we will explore non-personalised algorithms for exploit (random, popularity, average) in subsection 4.2.2 multi-armed bandit algorithms and in subsection 4.2.3 collaborative filtering algorithms.

4.2.1. Non-personalised exploit

Figure 4.1 shows the results of using non-personalised algorithms (random, average and popularity) during the exploitation phase for the CM100K dataset. For each algorithm we can see a set of three plots for three different warm-up durations: 5 %, 10 % and 15 %. To help results visualisation, we have plotted the difference of the recall curve obtained after the chosen warm-up and the recall curve of the exploitation algorithm without warm-up (from now on we will call this curve the "baseline"). Each line represents a different warm-up strategy for the same exploitation algorithm.

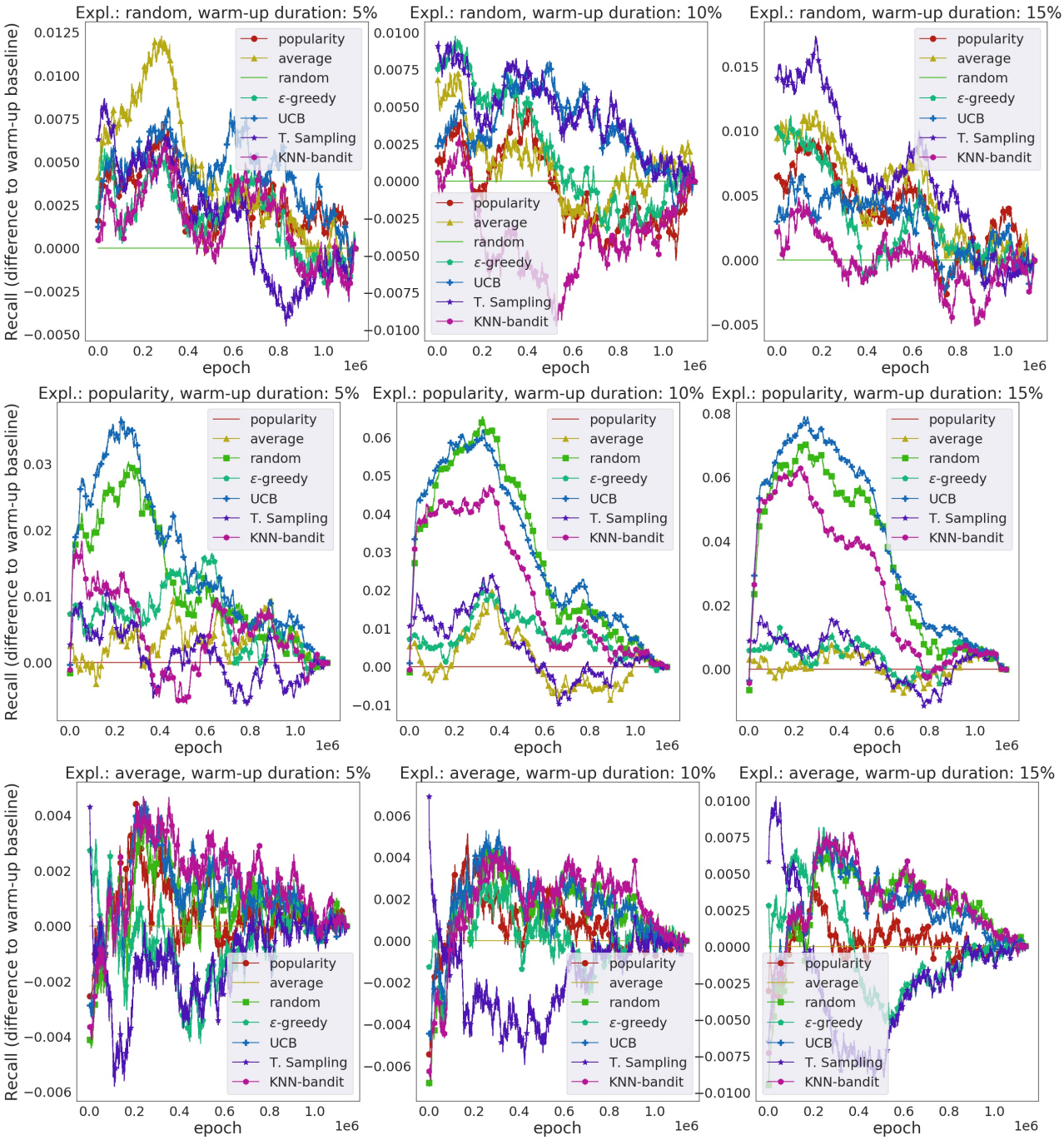

Figure 4.1: Results of non-personalised algorithms in the exploitation phase for CM100K dataset. For the exploitation with both random and average the warm-up phase does not seem to affect the exploitation phase, as there are a lot of noise in the recall curves. For the exploitation with popularity the different warm-up strategies yield different results. The most positive ones are UCB, random and KNN-bandit, as we can see a higher recall when we use these strategies. This indicates that for the first recommendations is worth doing some exploration to determine the most popular and then continue with a greedier approach.

We can see that for both exploitation algorithms random and average there is much noise in the recall curves and the differences in recall with respect to the baseline are too small (about $0{,}01$ for random in all warm-up durations and less than $0{,}01$ for average in all durations). This indicates that the warm-up phase is not affecting the exploitation phase. In the case of a random exploitation, this was expected because in every iteration it chooses a random item regardless of the user and the previous knowledge. In the case of average, as it is the optimal non-personalised strategy for the CM100K dataset [Cañamares and Castells, 2018], it is difficult for a different warm-up strategy to improve optimal results, so they only introduce noise in the recall.

For the popularity exploitation plots we can see a difference in the recall curves depending on the chosen algorithm for the warm-up, the best ones being UCB, random and KNN-bandit for all durations of warm-up. We can see that UCB and random recall curves behave similarly. This can be due to the fact that with few epochs, the exploratory term of UCB is dominant with respect to the exploitative term, so the algorithm acts randomly. This would also explain why both UCB and random warm-up work better than the other strategies for popularity exploitation, because popularity acts greedily, i.e. it finds the best item according to its current knowledge and recommends it until it cannot be recommended anymore (because the same item cannot be recommended to the same user twice), so choosing an algorithm that scores frequently would exhaust the best items in the warm-up phase and therefore popularity would perform worse during the exploitation phase.

4.2.2. Multi-armed bandit exploit

Figure 4.2 shows the results of using bandit algorithms during the exploitation phase for the CM100K dataset as in figure 4.1.

For UCB we can see similar results to those that we got in figure 4.1 for random and average, there is a lot of noise in the recall curves, especially for 5 % and 10 % warm-up. For 15 % warm-up we can see an improvement in the recall curves with popularity, average and ε-greedy as the warm-up algorithms.

For both ε-greedy and Thompson Sampling we observe that using random, UCB and KNN-bandit in the warm-up phase leads to a better recall curves during the exploitation phase.

For KNN-bandit the behaviour is far different than the other bandits algorithms. With 5 % warm-up, UCB improves the exploitation recall, while with 10 % and 15 % warm-up we do not really see an improvement in recall.

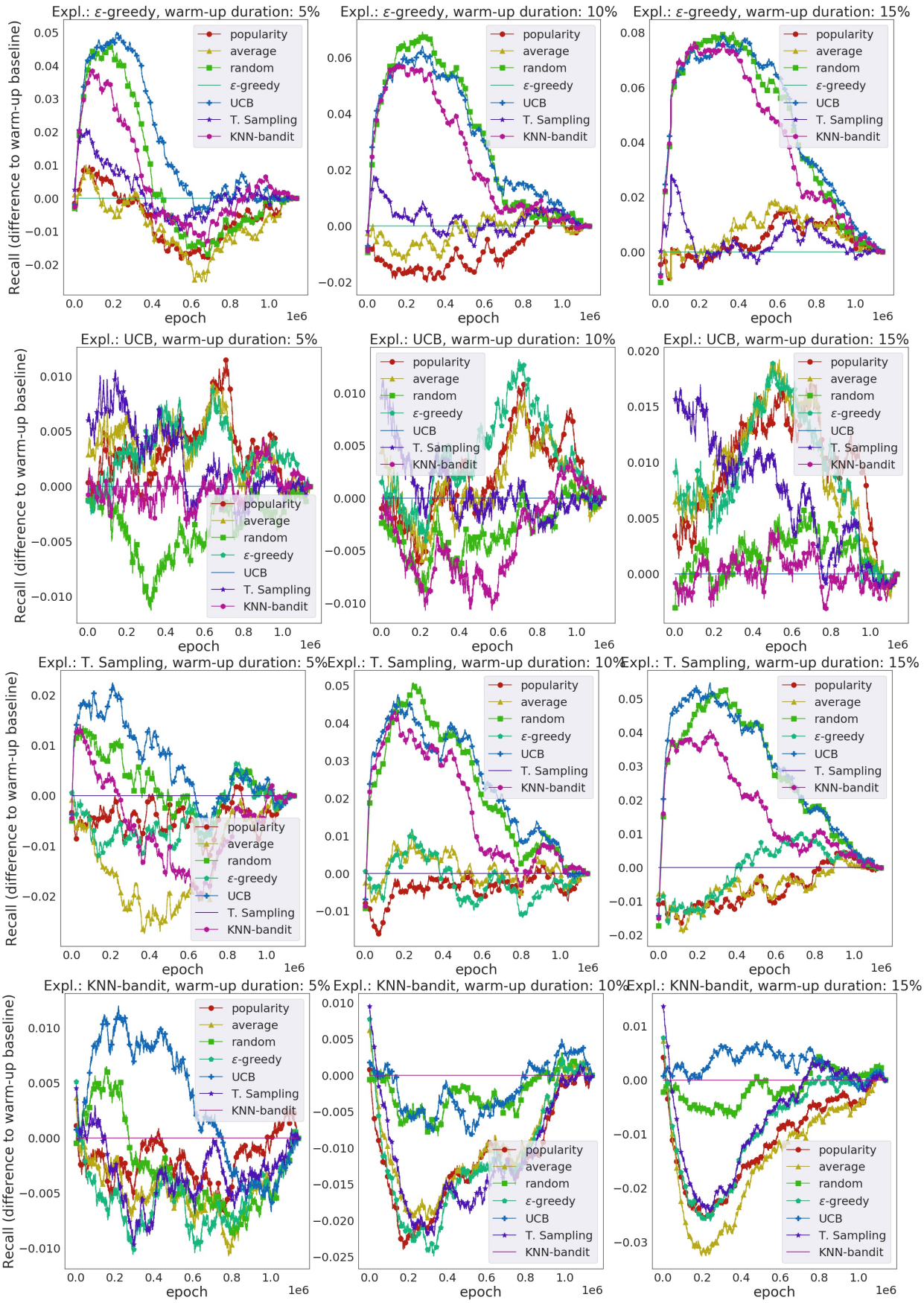

Figure 4.2: Results of bandits algorithms in the exploitation phase for CM100K dataset. For ε-greedy the warm-up with UCB, random and KNN-bandit lead to a better results, which indicates again that it is positive to begin the recommendation with more exploration. For UCB there are a lot of noise in the recall curves and we cannot conclude that the warm-up affects the exploitation. For Thompson Sampling the results are similar to ε-greedy, with a better result in the exploitation when there is a lot of exploration in the warm-up. For KNN-bandits the results are very different from the other exploitation strategies, with no improvement in the exploitation recall with 10 % and 15 % warm-up.

4.2.3. Collaborative filtering exploit

In figure 4.1 we plotted the results of using collaborative filtering algorithms during the exploitation phase for the CM100K dataset as in figure 4.1.

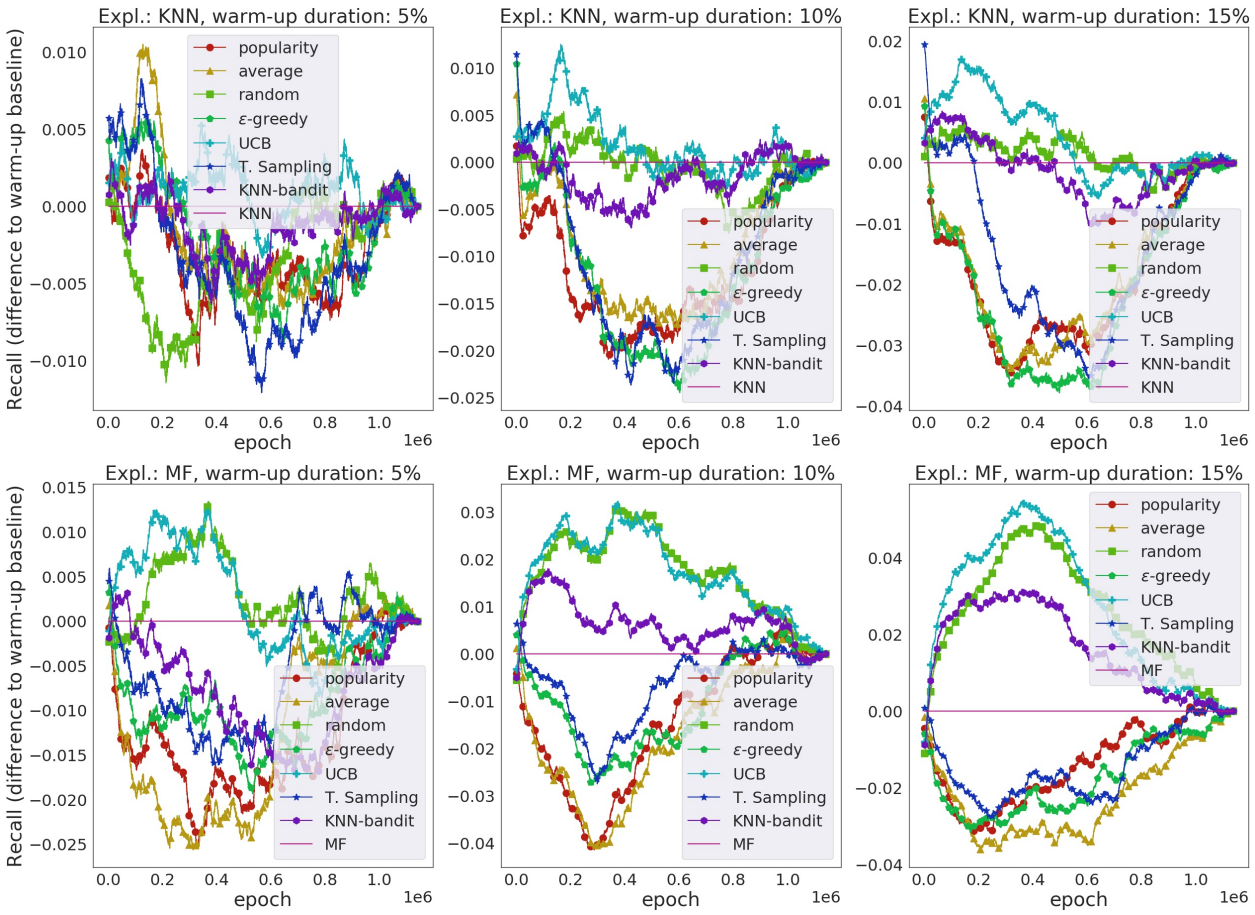

Figure 4.3: Results of collaborative filtering algorithms in the exploitation phase for CM100K dataset. For the exploit with KNN, the only warm-up algorithms that produce a positive impact are UCB and random, but this improvement is less than the one we got with other exploitation strategies (0.02 for UCB compared to 0.05 with ε-greedy in figure 4.2). For Matrix Factorization we can see that the recall improves with more warm-up time, so we can conclude that the warm-up phase is positive for this algorithm.

For user-based KNN we have a similar behaviour to what we got with KNN-bandit in figure 4.2. For 15 % warm-up, we can see that Thompson Sampling gives the better recall at the start of the exploitation phase but it drastically drops in the next epochs. As we are using a fixed test dataset we can explain this behaviour as follows: Thompson Sampling is the algorithm that performs better without any previous knowledge (we also showed this in [López Ramos et al., 2019, page 30]), so it is the algorithm that discovers most positive ratings in the warm-up phase. However when we get to the exploitation phase, as the test dataset is fixed, it is difficult for the exploitation algorithms to hit, because there are a lot of negative ratings left. We can hypothesise that this behaviour could not necessarily happen in a real environment since there are new items and users continuously being added to the system.

For matrix factorization we see a positive impact of the warm-up phase in all cases, the best strate-

gies being random, UCB and KNN-bandit.

4.2.4. Final comparison and wrap-up

The recall curves in figures 4.1, 4.2 and 4.3 are relative to the algorithm used in the exploitation phase as we are subtracting the recall of the baseline to them. In order to fairly compare the different strategies we have to join all of them in a single plot with the same baseline. In figure 4.4 we show the best curves for each exploitation algorithm for all the chosen warm-up sizes and we have subtracted the recall of the random curve to help visualise the differences. Each curve represents a pair [warm-up algorithm]+[exploitation algorithm] with this same notation in the legend.

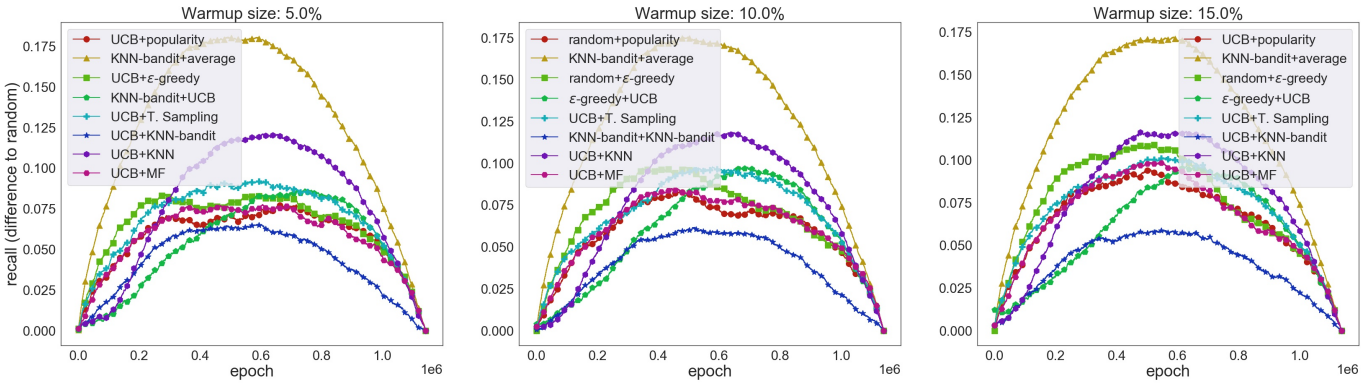

Figure 4.4: Comparison of the best combinations of warm-up and exploitation for CM100K dataset. For the CM100K dataset, the average algorithm is the one that has the best result in terms of recall. Some algorithms like ε-greedy and Thompson Sampling improve with more warm-up time. This improvement occurs with random and UCB (which in the first epochs of recommendation has a similar behaviour than random with most of recommendations being exploratory) so we can conclude that with more exploration on the first recommendations and more exploitation after leads to a positive impact on recall.

We can see that the best strategy for the CM100K dataset is by far the average algorithm. In this algorithm we do not really see an improvement on its performance with more warm-up time. The KNN recall follows the same behaviour.

This is not the case for the ε-greedy algorithm, where we can see its performance improves with more warm-up size. We can see this improvement in ε-greedy occurs with random warm-up so we can interpret this as it is positive for the algorithm to “sacrifise” the hit in the first epochs by doing exploratory random choices and then use this knowledge to improve the performance of the algorithm by doing more exploitation than exploration.

The rest of the exploitation algorithms behave similar to ε-greedy, but with less improvement with more warm-up.

Figure 4.5 compares the values of recall at the end of the warm-up phase for all the algorithms studied. The purpose of this figure is to know which algorithm hitted the most during the warm-up phase regardless of the impact during the exploitation phase.

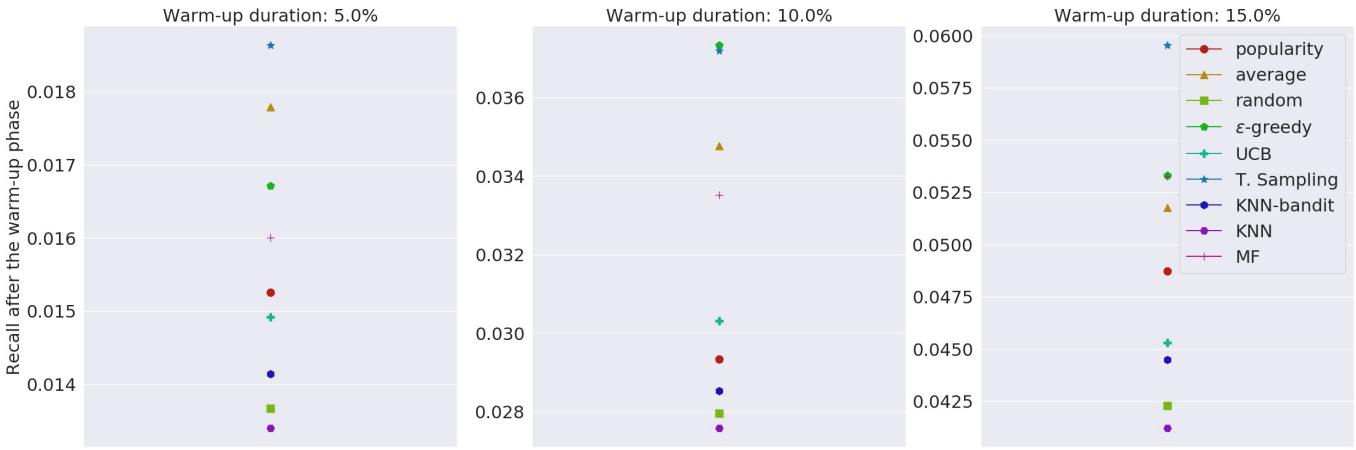

Figure 4.5: Recall at the end of the warm-up phase for CM100K dataset. The top three algorithms during the warm-up phase are Thompson Sampling, ε-greedy and average. For 5 % warm-up duration Thompson Sampling is the best one, however for 10 % warm-up, ε-greedy beats it. For 15 % duration, Thompson Sampling becomes the best algorithm during the warm-up phase and Matrix Factorization improves beating the average algorithm.

We can conclude that for the CM100K algorithm the best approach is the average algorithm and as we have seen in section 4.2.1 for this algorithm the warm-up does not improve the exploitation phase. During the warm-up phase the average algorithm was in the top three algorithms for 5 % and 10 % warm-up duration and it was the forth best for 15 % warm-up. However if we want to quickly start the system and maximise the recall during the warm-up phase, the best algorithm to do so is Thompson Sampling.

4.3. MovieLens

In this section, as in section 4.2, we present the results of applying different warm-up strategies to the MovieLens1M dataset. This section has three different subsections which correspond with the type of algorithms used for the exploitation phase. In subsection 4.3.1 we show the results of the exploitation with non-personalised algorithms (random, average and popularity), in section 4.3.2 with multi-armed bandits algorithms (ε-greedy, UCB, Thompson Sampling and KNN-bandit) and in section 4.3.3 with collaborative filtering algorithms. Finally in section 4.3.4 we compare the results of the best combinations of the past sections and extract conclusions of the experiment.

4.3.1. Non-personalised exploit

Figure 4.6 shows the recall curves of non-personalised algorithms during the exploitation phase after the warm-up ended for three different warm-up duration (5 %, 10 % and 15 % of total ratings in the dataset). Each colour represents a warm-up strategy.

Although the random approach is not a good strategy for the exploitation phase, its plots give an

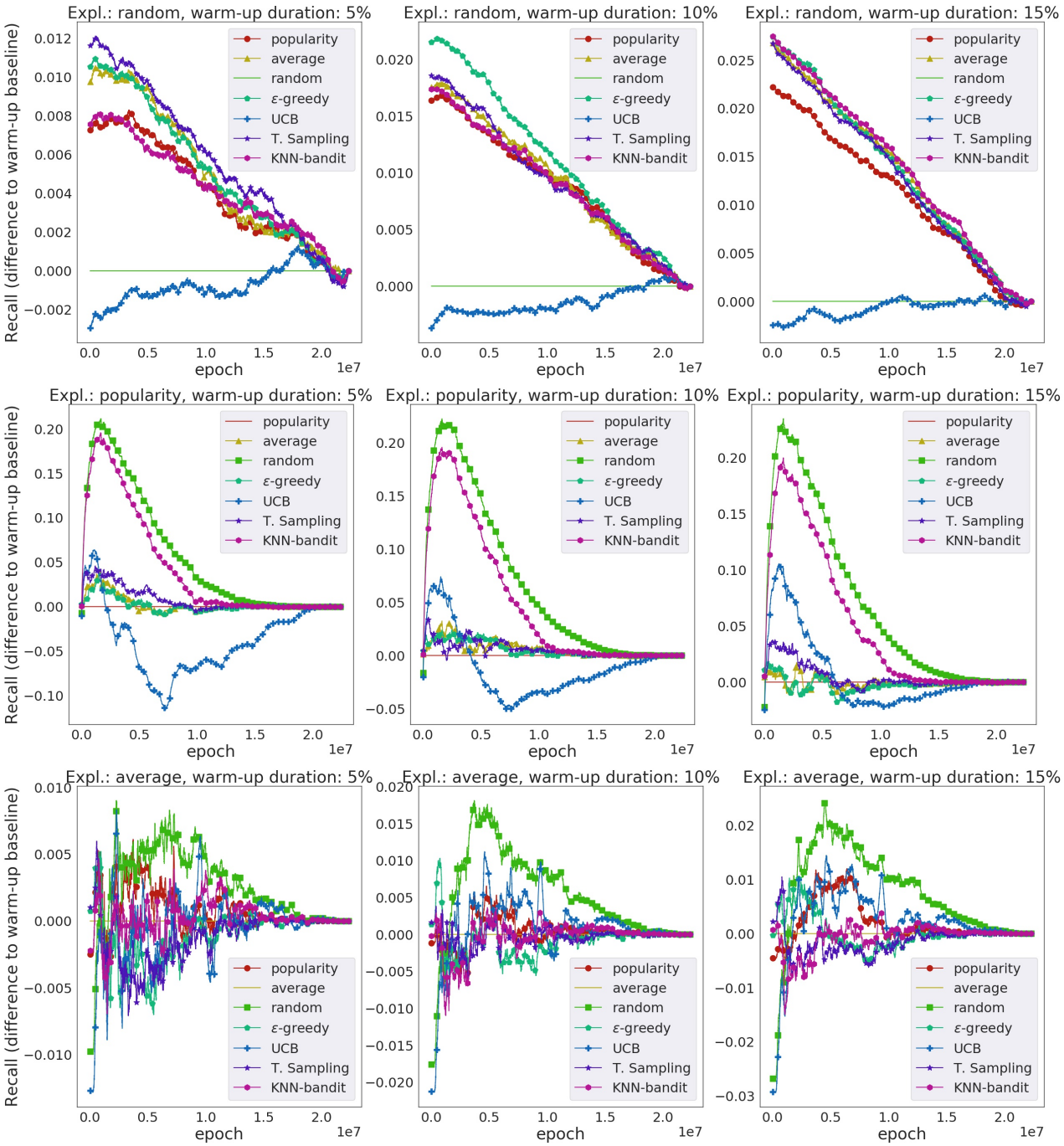

Figure 4.6: Results of non-personalised algorithms in the exploitation phase for MovieLens1M dataset. The random plots help evaluate which algorithm performed better during the warm-up phase because the recall at the start of this plots is the recall at the end of the warm-up phase. In the popularity plots we can see that the random warm-up is the strategy that gives the best recall curve for all the warm-up sizes. For the average plots, again the random warm-up is the one that performed better but with less difference with the other warm-up strategies. This indicates that more exploration in the first stages of the recommender system help improve the performance of the exploitation algorithms.

insight on what are the strategies that performed better during the warm-up phase because at the start of these plots we can see how the warm-up phase ended. For 5 % warm-up, we can see that the Thompson Sampling algorithm is the one that starts with the highest recall, for 10 % the best algorithm for the warm-up was ε-greedy and for 15 % the best was KNN-bandit. These plots start with the recall at the end of the warm-up phase and then monotonically decrease because the random strategy is not going to improve the warm-up algorithms.

For the exploitation with the popularity algorithm the random warm-up is the strategy that yields the best recall curve, followed closely by the KNN-bandit algorithm. The rest of the warm-up strategies does not seem to improve the recall of the popularity algorithm during the exploitation phase. This strengthens the idea that for the warm-up phase is better to use more exploratory strategies and during the exploitation phase more greedy approaches (the popularity algorithm is the greediest possible strategy, as it always recommends the most popular item to all the users).

The average plots are very similar to what we got in figure 4.1 for the average algorithm: the warm-up does not seem to affect the exploitation phase as there are a lot of noise in the curves. However once we increase the warm-up duration we can see an improvement in the recall curves, specially for the random warm-up. As the random algorithm fails a lot because it does not account the information learnt in past recommendations, it discards those unpopular items during the warm-up phase and thus during the exploitation phase the average algorithm performs better.

4.3.2. Multi-armed bandit exploit

Figure 4.7 shows the recall curves during the exploitation phase using multi-armed bandits algorithms. Again each curve represent a different warm-up strategy with the same exploitation algorithm.

In the ε-greedy exploitation, we observe that the warm-up algorithm that improves the baseline recall the most is the KNN-bandit for all warm-up sizes. However this improvement is not as significant as other that we got in the past plots (about $+0{,}05$ with respect to the baseline for ε-greedy compared to $+0{,}2$ for the popularity exploitation in figure 4.6). Most of the warm-up strategies helped improve the recall during the exploitation phase, specially in the first epochs of the exploitation, except the UCB warm-up which underperformed the baseline with a decrease of more than $0{,}15$ in the recall.

For the UCB plots, we observed that almost all the warm-up strategies improved the exploitation recall and this improvement continues with more warm-up time, changin from about $+0{,}02$ recall with 5 % warm-up to $+0{,}05$ for 15 % warm-up. As we proved in [López Ramos et al., 2019], the UCB algorithm is not a good strategy for the MovieLens1M dataset, so another algorithm during the warm-up phase helps the system to produce more useful recommendations than the UCB algorithm. However when the system reaches 15 million epochs of recommendatios, there is a drastic decrease in the recall. This can indicate that the UCB algorithm performs better in the last epochs of the recommendation, when

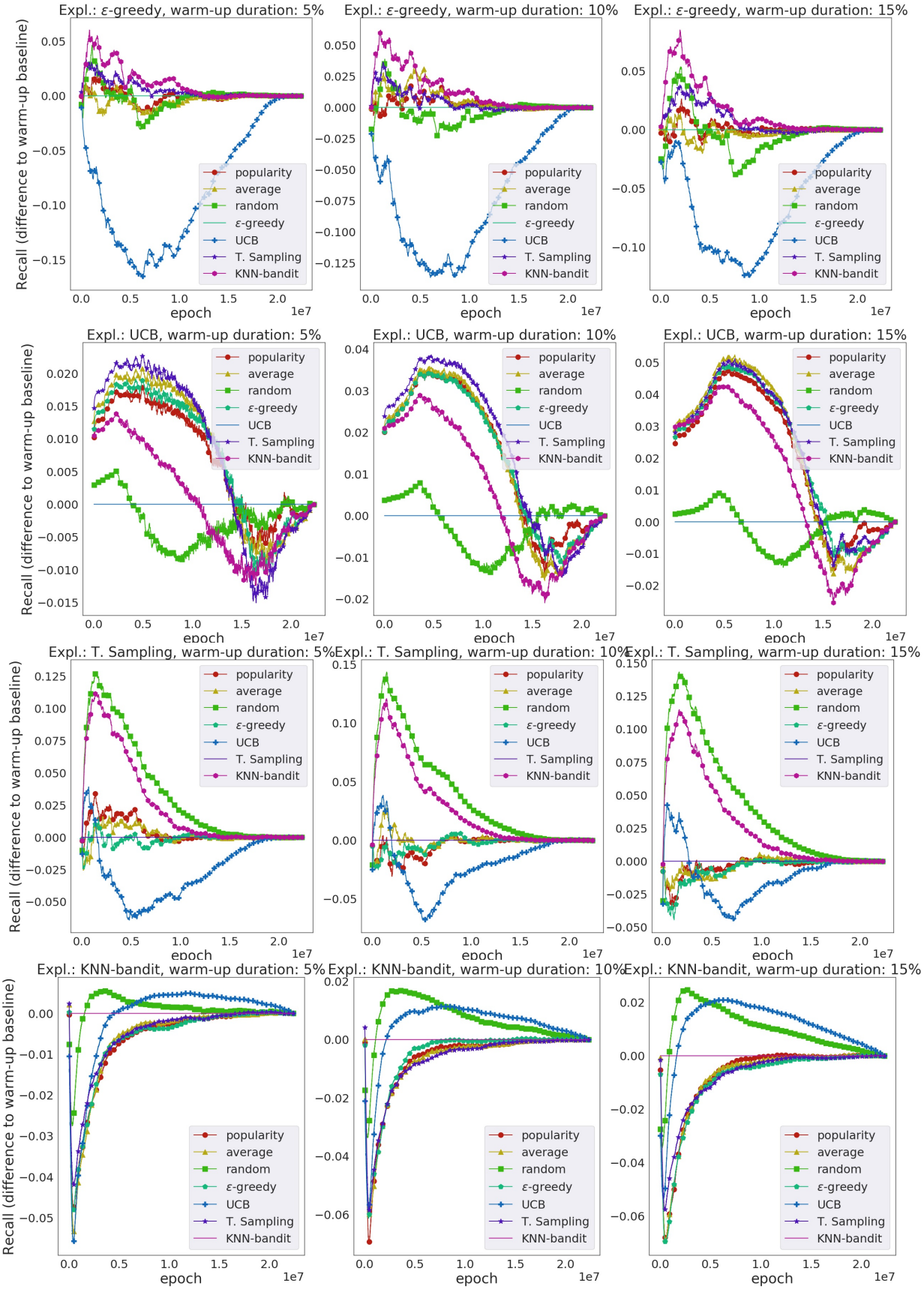

Figure 4.7: Results of bandits algorithms in the exploitation phase for MovieLens1M dataset. In the ε-greedy plots, the KNN-bandit algorithm is the warm-up strategy that helps the exploitation phase the most, but this improvement is not as big as others we got for non-personalised exploitations in figure 4.6. As UCB is not the best strategy for the MovieLens dataset [López Ramos et al., 2019], almost every warm-up strategy improves the performance of the recommender during the exploitation phase. For Thompson Sampling, the random warm-up is the strategy that improves the exploitation the most, indicating that more exploration in the first epochs is positive for the later performance of the recommender. The situation we observe for the KNN-bandit algorithm is much different from the other bandit algorithms, with a drastic drop in the recall in the first epochs of the exploitation for all chosen warm-up algorithms.

the system has much more information about the user and the items, so maybe with more warm-up than 15 % we can obtain a good performance with the UCB algorithm.

The Thompson Sampling plots are very simmilar to the ones that we got in figure 4.6 for popularity. The random and the KNN-bandit warm-ups are the ones that help the best the Thompson Sampling algorithm during the exploitation phase. This reiterates that in the first epochs of the recommendation is worth doing more exploitation to help improve the performace of the algorithms in the next epochs.

The behaviour of the recall curves for the KNN-bandit exploits are very different from the other exploitation strategies. For all the warm-up strategies we see a high decrease in the recall at the beginning of the exploitation phase and then a drastic increase. More warm-up time does not solve this problem because in the 10 % and 15 % warm-up plots we can see that this decrease becomes worse. Some of the warm-up strategies (random and UCB) improve the baseline recall after this decrease but we can assume that this improvement (about $+0{,}02$) is not worth after the drastic decrease (more than $-0{,}06$) for the KNN-bandit algorithm.

4.3.3. Collaborative filtering exploit

Figure 4.8 shows the results for the KNN and Matrix Factorization algorithms during the exploitation phase. Each line represents the difference of the recall for each chosen warm-up algorithm and the recall of the baseline algorithm, which is the KNN or Matrix Factorization without warm-up i.e., the same collaborative filtering algorithm during the warm-up phase.

For KNN, all the chosen warm-up strategies improved the exploitation recall in the first epochs of the exploitation phase, however only random, KNN-bandit and UCB maintained this improvement during the rest of the exploitation while in the other algorithms the recall dropped and underperformed the baseline. Although in the first epochs the random warm-up is the best strategy, the UCB warm-up helps the KNN algorithm to perform better during the rest recommendation cycles.

For the Matrix Factorization the behaviour is similar to the KNN case. The exploitation with random, KNN and UCB in the warm-up phase produces the best recall curves. For the rest of the warm-up strategies, the warm-up does not help improve the exploitation recall.

4.3.4. Final comparison and wrap-up

In figure 4.9 we compared the best strategies from figures 4.6, 4.7 and 4.8 in the same plot, but this time instead of subtracting the baseline recall, we subtracted the random recall to help visualise the curves.

For the MovieLens1M dataset the algorithms that performed the best during the exploitation phase are the collaborative filtering (Matrix Factorization and KNN), Thompson Sampling and popularity. When

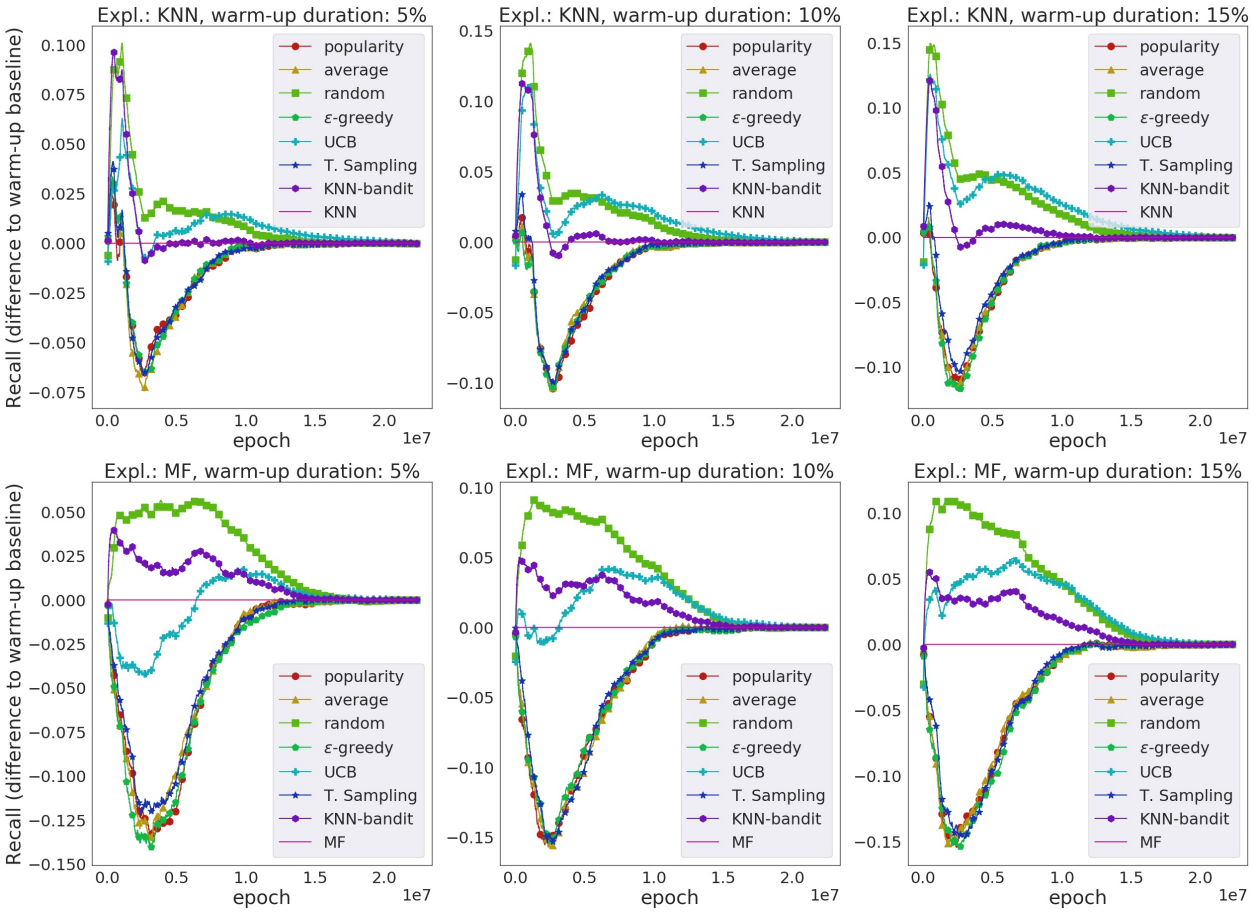

Figure 4.8: Results of collaborative filtering algorithms in the exploitation phase for MovieLens1M dataset. Both sets of plots shows similar results for the same warm-up strategies. In the KNN plots, we can see a big increase in the first epochs for all the warm-up strategies, however only UCB, KNN-bandit and random maintain this improvement with respect to the baseline in the succesive epochs. For Matrix Factorization again random, UCB and KNN-bandit are the best warm-up strategies. The rest of the warm-up algorithms worsen the exploitation recall.

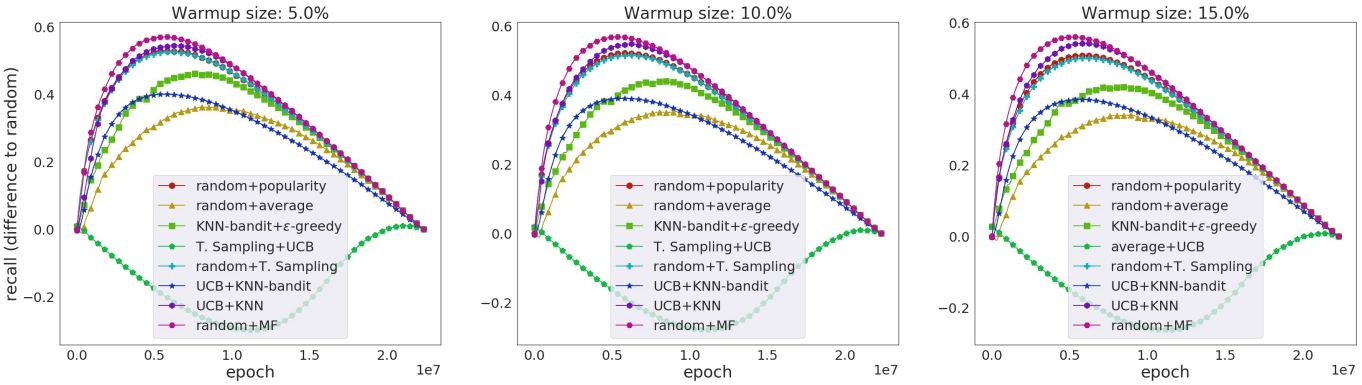

Figure 4.9: Comparison of the best combinations of warm-up and exploitation for MovieLens1M dataset. In the MovieLens1M dataset the best strategies are the collaborative filtering ones (KNN and Matrix Factorization) followed closely by Thompson Sampling and popularity. With more warm-up we can see more differences between the top recall curves: while in the 5 % plot, the top 4 curves are very close to each other, in the 15 % plot the performance of the collaborative filtering algorithms improved more than Thompson Sampling and popularity.

we increase the warm-up duration, we observe an increase in the recall of the collaborative filtering algorithms, whose recall lines separate from the other two (Thompson Sampling and popularity) top strategies. As we saw in figure 4.8 the warm-up positively impacts the recall during the exploitation phase, specially for the KNN algorithm (more than $+0{,}1$ recall with respect to the baseline). So we have proven that in the first epochs of the recommendation we should adopt an exploratory strategy (like random) and then change the algorithm to a more classic one (like collaborative filtering) to maximise the hit of the recommendations.

Figure 4.10 shows the value of recall after the warm-up phase for all the algorithms studied.

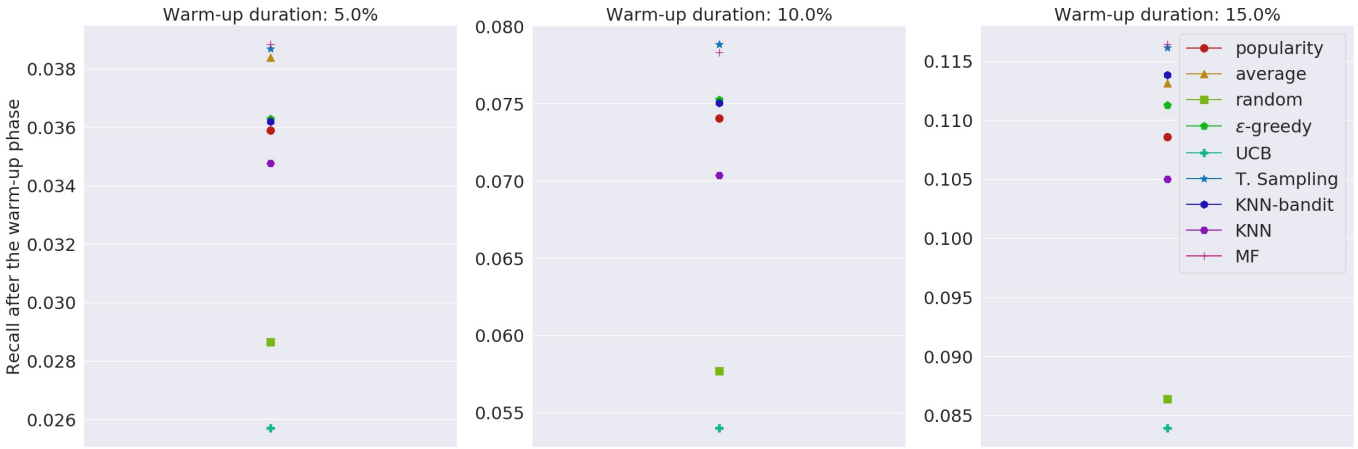

Figure 4.10: Recall at the end of the warm-up phase for MovieLens1M dataset. For 5 % warm-up Matrix Factorization (MF), Thompson Sampling and average are the best strategies in terms of recall. For 10 % and 15 % warm-up MF and Thompson Sampling distance from the rest algorithms.

We can conclude that both collaborative filtering approaches (KNN and MF) are the algorithms that perform best for the MovieLens1M dataset. For these algorithm we saw an increase on the exploitation recall with random, UCB and KNN-bandit as the warm-up algorithms and this increase is higher with more warm-up duration. With 15 % warm-up KNN-bandit was the third best algorithm after the warm-up phase ended in terms of recall so KNN-bandit is a good strategy to start the system with and then change it to a collaborative filtering approach to improve the recall in the long term.

4.4. Twitter

In this section we show the results of different combinations of warm-up and exploitation strategies for Twitter dataset. Unlike the other two datasets, CM100K and MovieLens, in Twitter dataset we recommend contacts so we must apply a new restriction: if a user u follows another user v we do not recommend v to follow u back. Section 4.4.1 shows the results of applying non-personalised approaches (random, average and popularity) in the exploitation phase, section 4.4.2 multi-armed bandits algorithms (ε-greedy, UCB, Thompson Sampling and KNN-bandit) and section 4.4.3 collaborative filtering algorithms (KNN and Matrix Factorization).

4.4.1. Non-personalised exploit

Figure 4.11 shows the recall curves of non-personalised algorithms (random, popularity and average) during the exploitation phase for the Twitter dataset. Each curve represent the same algorithm in the exploitation phase with a different warm-up strategy.

For the random exploitation we can see that for the 5 % and 10 % warm-up the recall curves have a drastic increase in the recall at the beginning of the plots and then the recall progressively decreases. For the 15 % curves there is no significant improvement in recall and we see a lot of noise. The maximum recall decreases with more warm-up duration from about $0{,}02$ in the 5 % warm-up to $0{,}008$ in the 15 % warm-up. The random approach recommends an item randomly regardless of the knowledge collected during the warm-up phase so, as we have the restriction that an item cannot be recommended to the same user more than once, the strategies that perform best during the warm-up phase exhaust the most popular items during the warm-up phase leaving the worst items to be picked by the random approach with a higher probability during the exploitation phase.

In the popularity exploit the warm-up strategies that improves the recall the most are the UCB, random and KNN-bandit algorithms. All the rest of warm-up strategies improve the recall curves but less than the three mentioned before. For the random warm-up we can see that with more warm-up duration the recall during the exploitation phase improves. As the popularity algorithm recommends the item with more positive ratings, the random warm-up helps discard those items that has the least number of interactions (because an item that has already been recommended to an user cannot be recommended to the same user again). For both the UCB and KNN-bandit warm-up with more warm-up duration the recall decreases.

The average plots are very similar to what we observed in CM100K (figure 4.1) and in MovieLens1M datasets (figure 4.6): the warm-up does not affect the performance during the exploitation phase because the recall curves are very noisy.

4.4.2. Multi-armed bandit exploit

Figure 4.12 shows the results of using multi-armed bandits algorithm during the exploitation phase for the Twitter dataset. As in previous figures, each line represents the recall after different warm-up strategies with the same exploitation algorithm.

In the ε-greedy plots we can see that the KNN-bandit, UCB and random are the warm-up strategies that improve the most the recall during the exploitation phase. The rest of the recall curves are below the baseline (ε-greedy without warm-up) meaning these strategies are not a good approach for the warm-up of the ε-greedy algorithm. With more warm-up duration we can see that the recall decreases specially for UCB whose recall curve is below the baseline in both 10 % and 15 % warm-up duration.

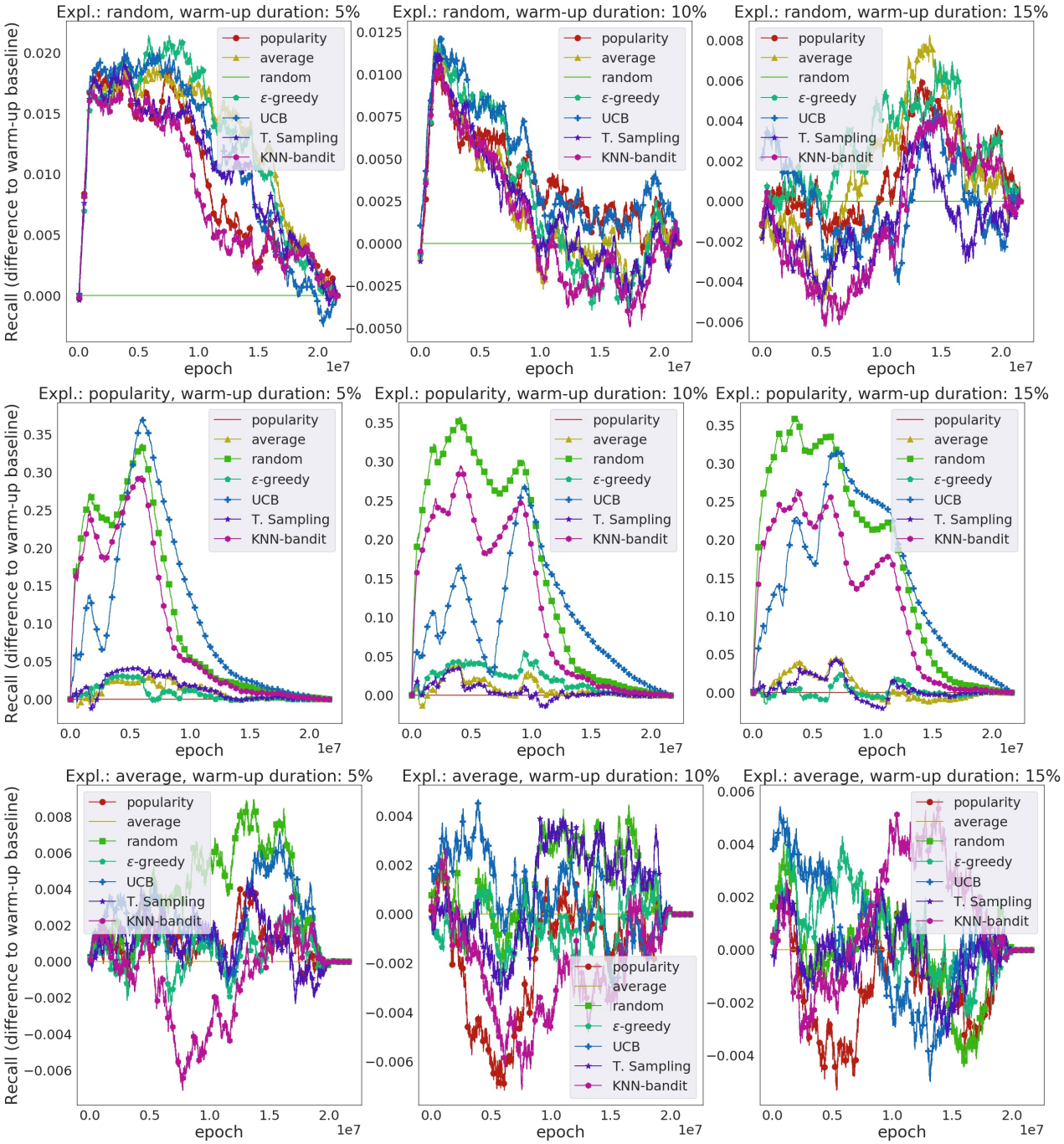

Figure 4.11: Results of non-personalised algorithms in the exploitation phase for Twitter dataset. As in the previous datasets (CM100K and MovieLens1M) the only exploitation algorithm that is affected by the warm-up is the popularity. In the popularity plots we can see that the warm-up strategies that perform best are the UCB, random and KNN-bandit algorithms. These recall curves improves with more warm-up duration only for the random warm-up and it decreases for the UCB and KNN-bandit warm-ups. For the random and average plots we observe a lot of noise in the curves so we can conclude that the warm-up does not affect these exploitation strategies.

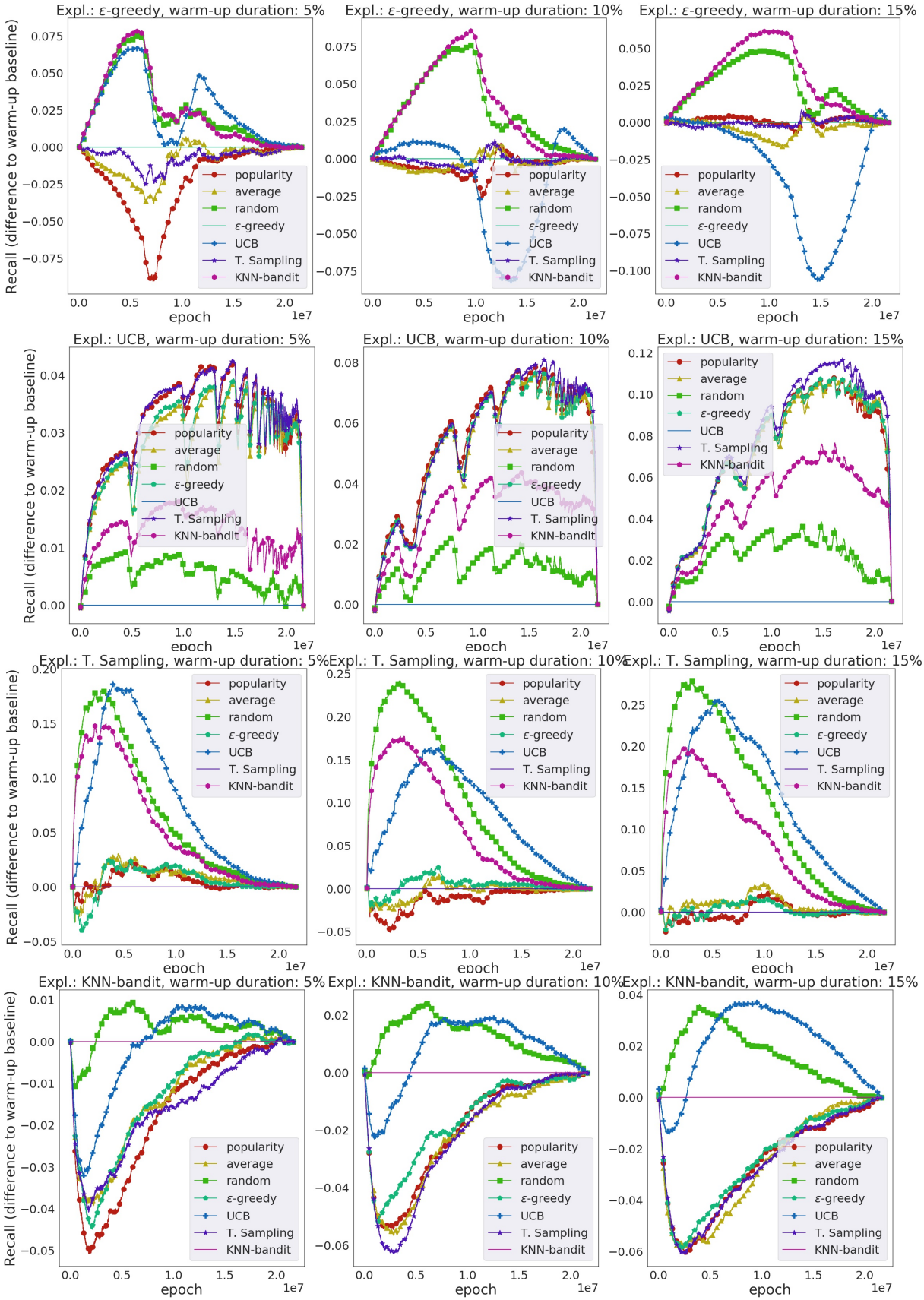

Figure 4.12: Results of bandits algorithms in the exploitation phase for Twitter dataset. For ε-greedy and Thompson Sampling we see that the warm-up strategies that yield the best recall are UCB, random and KNN-bandit. The rest of the warm-up strategies maintain or worsen the baseline recall (exploitation algorithm without warm-up). For UCB and KNN-bandit the results is much different. In Thompson Sampling the recall increases with more warm-up duration while in ε-greedy the opposite happens. In the case of UCB all warm-up strategies increase the recall during the exploitation phase, inverting the behaviour of the previously mentioned algorithms. For KNN-bandit only the random and UCB strategies improve the recall in the exploitation phase with a period of worse recall in the first epochs of the exploitation, specially in the UCB warm-up. In this plots we can see that more warm-up duration improves the performance during the exploitation phase.

The UCB algorithm in its first epochs act exploratory as random does in all the iterations. This means that the ε-greedy algorithm benefits form an exploratory warm-up in the first epochs but it is not a good idea to extend this exploratory behaviour beyond the 5 % warm-up because this causes the recall to drop.

In the UCB plots we observe the opposite behaviour as ε-greedy, the UCB (baseline), random and KNN-bandit are the warm-up approaches that achieve the least recall during the exploitation phase. The rest of the algorithm impact the recall similarly. With more warm-up duration the recall during the exploitation phase keeps improving. As we discussed in section 3.2.3, the UCB algorithm needs some epochs to initialise the value of the uncertainty at the beginning of the recommendation loop. This initialisation can be be replaced by the warm-up phase with better result as shown by the UCB plots.

For Thompson Sampling the results are similar to ε-greedy. The best warm-up strategies are again the random, UCB and KNN-bandit algorithms. This time the other warm-up strategies (average, popularity and ε-greedy) do not negatively affect the recall during the exploitation phase but are similar to the baseline (Thompson Sampling without warm-up). This time, unlike what happened with ε-greedy, the recall improves with more warm-up duration and the recall after UCB warm-up stays above the baseline. We conclude that Thompson Sampling is another algorithm that benefits from a first exploratory phase and this phase can extend above the 15 % of the epochs.

In the KNN-bandit plots the best warm-up strategies are random and UCB while the rest of the algorithms are below the baseline recall. For UCB at the beginning of the exploitation phase we observe a decrease in the recall, but after this drop the recall grows much faster. This drop in the recall becomes less drastic with more warm-up duration so we could explore if warm-up beyond the 15 % helps even more the exploitation recall.

4.4.3. Collaborative filtering exploit

In figure 4.13 we plotted the results of collaborative filtering algorithms (KNN and MF) during the exploitation phase after the warm-up. Each curve corresponds to a different warm-up stratey.

In the KNN plots, we see that the warm-up approaches that improve the exploitation recall the most are UCB, random and KNN-bandit. This situation varies a lot depending on the warm-up duration. For 5 % warm-up only the UCB algorithm achieves an improvement during the exploitation phase and this improvement is less than $0{,}02$ with respect to the baseline. For 10 % and 15 % all random, UCB and KNN-bandit improve the exploitation recall and this improvement gets to $+0{,}15$ with respect to the baseline. We can also observe a drop in the recall curves after the warm-up with UCB in the 5 % and 10 % plots that disappears with 15 % plots. We conclude that the KNN algorithm benefits a lot from exploratory warm-up strategies. The rest of the warm-up algorithms performed worse than the baseline.

In the MF plots, again random, UCB and KNN-bandits are the only approaches that help improving

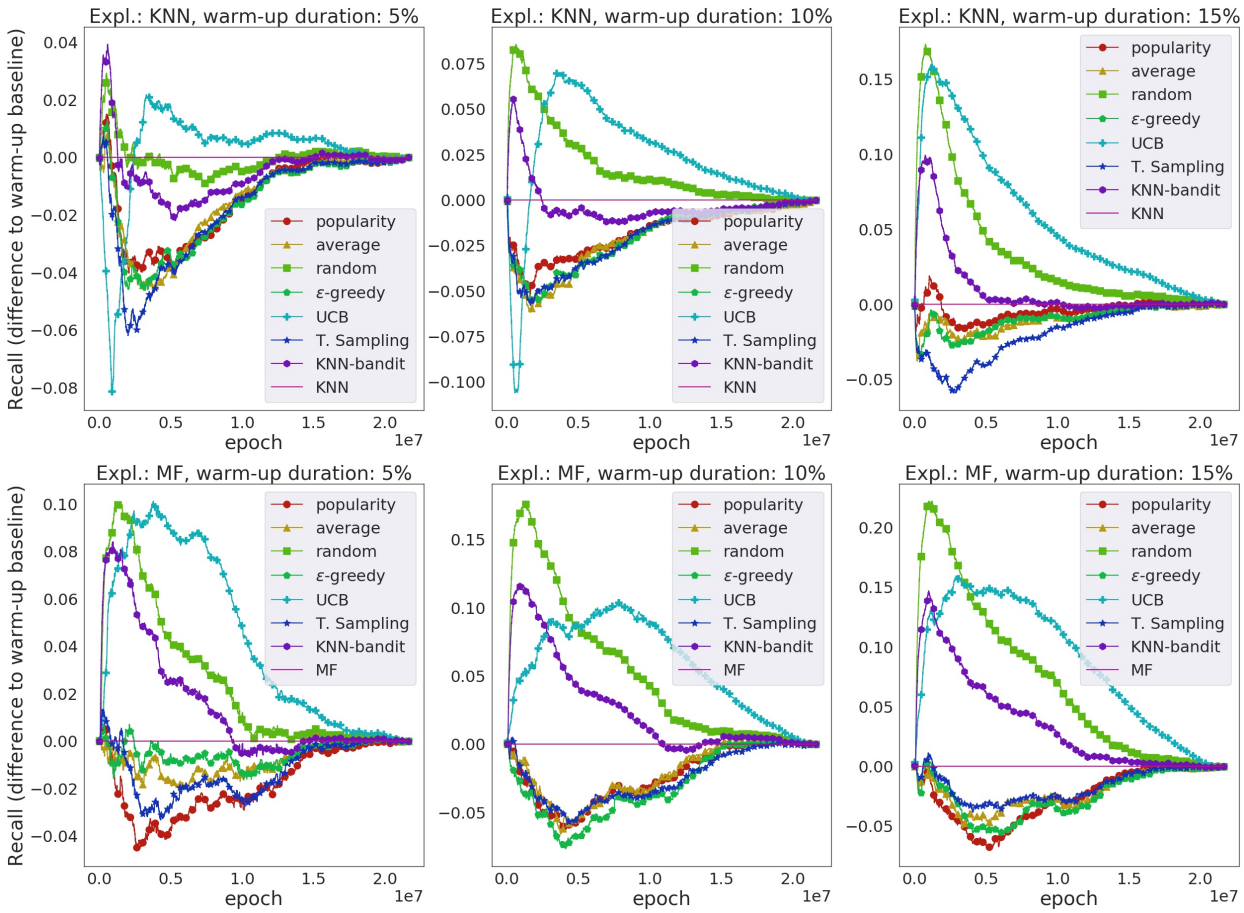

Figure 4.13: Results of collaborative filtering algorithms in the exploitation phase for Twitter dataset. For KNN, the best warm-up strategies are UCB, KNN-bandit and random. However for 5 % warm-up we can see an improvement in the recall only with UCB and this improvement occurs after a drop in the recall during the first epochs. For MF the same algorithms (random, UCB and KNN-bandit) are the ones that improve the exploitation recall the most. In this case there is not a drop in the exploitation recall in the first iterations. For both exploitation strategies, the rest of the warm-up approaches (popularity, average, ε-greedy and Thompson Sampling) impacted the exploitation recall negatively.

exploitation phase. The recall keeps improving with more warm-up duration but less than the previous KNN plots: from $+0{,}02$ to $+0{,}15$ in the KNN exploitation and from $+0{,}1$ to $+0{,}2$ in the MF exploitation.

4.4.4. Final comparison and wrap-up

Figure 4.14 shows the best strategies from figures 4.11, 4.12 and 4.13. To help compare the recall curves we subtracted the random recall to all of the rest algorithms. Each line represent a combination of [warm-up algorithm]+[exploitation algorithm] (notation in the legend) and show the recall only during the exploitation phase.

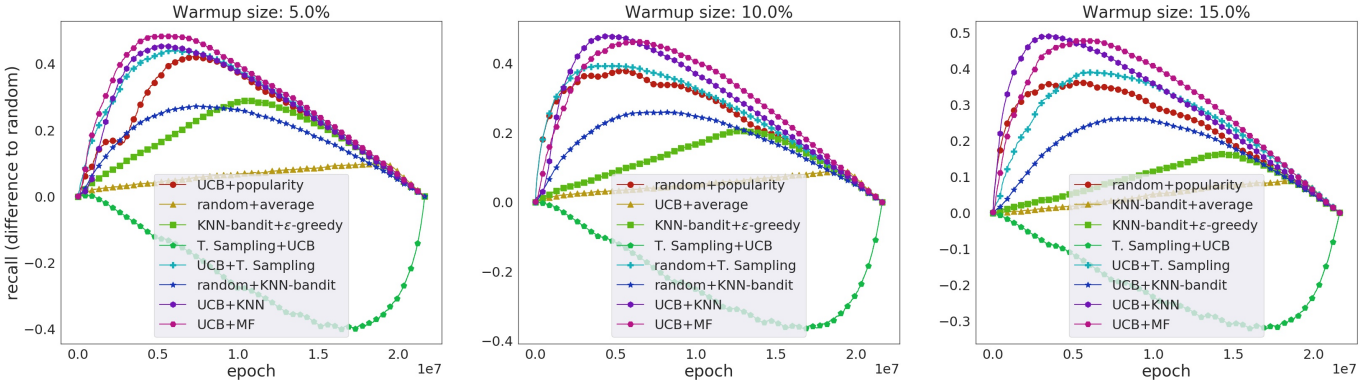

Figure 4.14: Comparison of the best combinations of warm-up and exploitation for Twitter dataset. For 5 % warm-up the best algorithm is the Matrix Factorization, however with more warm-up we can see that the KNN beats the MF algorithm in the first epochs of the exploitation. For the Twitter dataset again the collaborative filtering approaches win but are followed closely by the Thompson Sampling and popularity algorithms.

As in MovieLens dataset (figure 4.9), for Twitter dataset the collaborative filtering approaches are the ones that yield the best results during the exploitation phase. For 5 % warm-up we can see that the best approach is the Matrix Factorization algorithm, however with more warm-up duration the KNN algorithm beats the Matrix Factorization in the firsts epochs of the exploitation phase. Although UCB is most of the times the best strategy during the warm-up phase, during the exploitation phase is the worst one. For popularity and ε-greedy, in the 5 % warm-up plots we can see that these strategies are close to the collaborative filtering, however with more warm-up duration the collaborative filtering algorithms improve a lot more than popularity and ε-greedy, which improve little with more warm-up duration.

Figure 4.15 shows the value of recall after the warm-up phase for all the algorithms studied in the Twitter dataset.

We can conclude that for the Twitter dataset the collaborative filtering strategies are the best ones for the exploitation phase and this algorithms benefits for the warm-up with multi-armed bandit UCB strategy. UCB was also the algorithm that performed best during the warm-up phase, so a good strategy for twitter dataset would be to start the system using UCB and then changing it to a collaborative filtering approach.

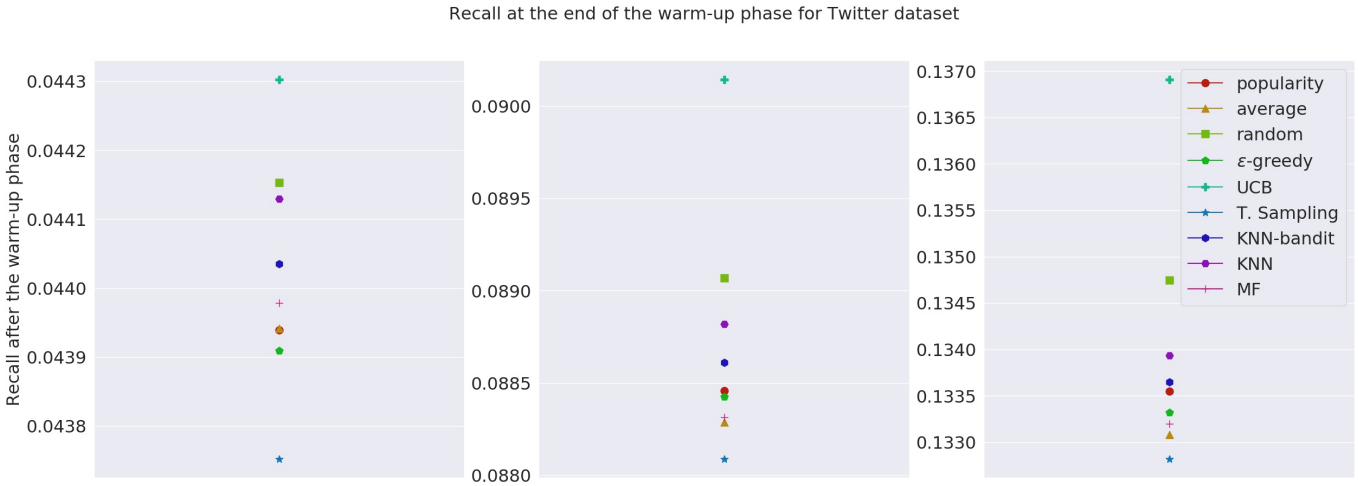

Figure 4.15: Recall at the end of the warm-up phase for Twitter dataset. For all the warm-up duration the best algorithm during the warm-up phase was UCB.

5

Conclusions

In this chapter we conclude the document by presenting in the first place in section 5.1 a summary of the tasks covered in this project and its contributions and last in section 5.2 a list of extensions to improve the developed work.

5.1. Summary and contributions

In this work we have studied the cold-start problem in recommender systems and we have studied multi-armed bandits algorithms to tackle it. As multi-armed bandit algorithms we have chosen ε-greedy, Upper Confidence Bounds and Thompson Sampling, which perform non-personalised recommendation, and KNN-bandit, which is based in the collaborative filtering algorithm K-Nearest Neighbours and thus it provides personalised recommendations.

We have consider the recommendation task as an iterative process with a constant interaction between the user and the system. To study the cold start problem, we have divided the recommendation loop in two different stages: the warm-up phase and the exploitation phase. During the warm-up phase the system starts with no previous knowledge about the users nor the items and it focuses on learning as much information as possible by recommending items to the users and observing their responses. On the exploitation phase, the system gathers all the information acquired during the warm-up phase and focuses on trying to hit the recommendations. In this context of warm-up and exploitation we have combined different recommendation strategies for the two phases. As warm-up strategies we have used both multi-armed bandits algorithms and classical non-personalised approaches (random, average and popularity). In the exploitation phase we added collaborative filtering algorithms to the latter.

To evaluate the performance of the combination of the chosen warm-up and exploitation algorithms we have conducted offline experiments, varying the warm-up duration from 5 %, 10 % and 15 % of the total of ratings considered. For those experiments we have taken three datasets form different domains CM100K (music), MovieLens1M (films) and Twitter (social network) and emulated the recommendation loop. The recall curves show that the warm-up strategies that improve the most the exploitation phase are random, UCB and KNN-bandit. The random approach is purely exploratory and UCB is more

exploratory in its first epochs. So we can conclude that a more exploratory warm-up leads to a better performance in the long term during the exploitation phase. For CM100K dataset the best strategy by far during the exploitation phase was the average approach, which was not affected by the warm-up ($+0{,}01$ recall in the best case with respect to the average algorithm without the warm-up phase). For MovieLens1M and Twitter, collaborative filtering algorithms (Matrix Factorization and KNN) were the ones that performed better during the exploitation phase. Both algorithms benefited from the warm-up phase and the exploitation recall increased with more warm-up duration. In MovieLens1M the exploitation with KNN improved from $+0{,}1$ recall with respect to the baseline (KNN with no warm-up) with 5 % warm-up to $+0{,}15$ with 15 % warm-up, the exploitation with Matrix Factorization improved from $+0{,}05$ with 5 % warm-up to $+0{,}1$ with 15 % warm-up. In Twitter dataset the KNN algorithm improved from $+0{,}02$ recall with 5 % warm-up to $+0{,}15$ with 15 % warm-up and Matrix Factorization from $+0{,}1$ with 5 % warm-up to $+0{,}15$ with 15 % warm-up.

Although the most exploratory strategies are the ones that improved the most the performance of the algorithms during the exploitation phase, if the system owner is interested in hitting the recommendations as soon as possible it would be best to choose another strategy. To study this we have compared the recall after the warm-up phase finished. The results show that for CM100K, Thompson Sampling and ε-greedy are the algorithms that hit the most during the warm-up phase, for MovieLens1M the best are Thompson Sampling and Matrix Facotirzation and for Twitter, the best algorithm during the warm-up phase was UCB. So we can conclude that Thompson Sampling easily and quickly adapts to the context.

5.2. Future work

In this work we have used offline experiments simulating the recommendation loop in order to test which are the best algorithms combinations for the warm-up and exploitation phases. Although offline evaluation is a fast way to check an hypothesis, its results may not correspond with the results of a real environment. In this context of simulating the recommendation loop, we are using a limited set of ratings given by the users to check if the recommendation was a hit. In the case of an unknown user, item pair, we ignore the recommendation and do not compute it in the recall curve. In a real environment such situation of an unknown rating would not happen if the user answers to every recommendation, so the results may differ to what we got with an offline evaluation.

In this project we have evaluated the warm-up problem when the system is new, but we have not tackled the new user or new item cold-start problems. This scenarios are difficult to evaluate in an offline experiment and we would need an online experimentation platform to test them.

The results of the recall curves represent a single execution of the recommendation loop. These results may differ with different simulations due to the non-deterministic character of some of the algorithms. We should run several executions of the recommendation loop and compute the mean of the

recall curve and get confidence intervals for each point in order to validate the results.

Most of the tested algorithms have hyper-parameters which we have not tunned in this work. Tuning the hyper-parameters can lead to different results improving or worsen the recall curves and changing the optimal combination of warm-up and exploitation strategies.

Bibliography

[Auer et al., 2002] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

[Cañamares and Castells, 2018] Cañamares, R. and Castells, P. (2018). Should I Follow the Crowd? A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. In *41st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2018)*, pages 415–424, Ann Arbor, Michigan, USA.

[Chapelle and Li, 2011] Chapelle, O. and Li, L. (2011). An empirical evaluation of thompson sampling. In *25th Conference on Neural Information Processing Systems (NIPS 2011)*, pages 2249–2257, Granada, Spain.

[Gentile et al., 2014] Gentile, C., Li, S., and Zappella, G. (2014). Online clustering of bandits. In *31st International Conference on Machine Learning (ICML 2014)*, pages 757–765, Beijing, China.

[Harper and Konstan., 2015] Harper, F. M. and Konstan., J. A. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19*.

[Hill et al., 2017] Hill, D. N., Nassif, H., Liu, Y., Iyer, A., and Vishwanathan, S. (2017). An efficient bandit algorithm for realtime multivariate optimization. In *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2017)*, pages 1813–1821, Halifax, NS, Canada,.

[Kawale et al., 2015] Kawale, J., Bui, H. H., Kveton, B., Tran-Thanh, L., and Chawla, S. (2015). Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1297–1305. Curran Associates, Inc.

[Kohavi and Longbotham, 2017] Kohavi, R. and Longbotham, R. (2017). *Online Controlled Experiments and A/B Testing*, pages 922–929.

[Li et al., 2010] Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *19th international conference on World wide web (WWW 2010)*, pages 661–670, Raleigh, North Carolina, USA.

[Lika et al., 2014] Lika, B., Kolomvatsos, K., and Hadjiefthymiades, S. (2014). Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065 – 2073.

[López Ramos et al., 2019] López Ramos, E., Sanz-Cruzado Puig, J., and Castells, P. (2019). *Bandidos multi-brazo en sistemas de recomendación*. UAM. Departamento de Ingeniería Informática.

[movielens, 1997] movielens (1997). Sitio web MovieLens.

[Paterek, 2007] Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8.

[Pereira and Varma, 2016] Pereira, N. and Varma, S. (2016). Survey on content based recommendation system.

[Puthiya Parambath and Chawla, 2020] Puthiya Parambath, S. A. and Chawla, S. (2020). Simple and effective neural-free soft-cluster embeddings for item cold-start recommendations. *Data Mining and Knowledge Discovery*.

[Rashid et al., 2008] Rashid, A. M., Karypis, G., and Riedl, J. (2008). Learning preferences of new users in recommender systems: an information theoretic approach. *Acm Sigkdd Explorations Newsletter*, 10(2):90–100.

[Ricci et al., 2011] Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B. (2011). *Recommender systems handbook*. Springer, New York; London.

[Rohani et al., 2014] Rohani, V. A., Kasirun, Z. M., Kumar, S., and Shamshirband, S. (2014). An Effective Recommender Algorithm for Cold-Start Problem in Academic Social Networks. *Mathematical Problems in Engineering*, 2014.

[Safoury and Salah, 2013] Safoury, L. and Salah, A. (2013). Exploiting user demographic attributes for solving cold-start problem in recommender system. *Lecture Notes on Software Engineering*, 1(3):303–307.

[Sanz-Cruzado and Castells, 2018] Sanz-Cruzado, J. and Castells, P. (2018). Contact recommendations in social networks. In Berkovsky, S., Cantador, I., and Tikk, D., editors, *Collaborative Recommendations: Algorithms, Practical Challenges and Applications*, pages 519–570. World Scientific Publishing.

[Sanz-Cruzado et al., 2019] Sanz-Cruzado, J., Castells, P., and López, E. (2019). A simple multi-armed nearest-neighbor bandit for interactive recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 358–362, New York, NY, USA. Association for Computing Machinery.

[Scott, 2015] Scott, S. L. (2015). Multi-armed bandit experiments in the online service economy. *Applied Stochastic Models in Business and Industry*, 31(1):37–45.

[Shani et al., 2005] Shani, G., Heckerman, D., and Brafman, R. I. (2005). An mdp-based recommender system. pages 453–460, Edmonton, Alberta, Canada.

[Sun et al., 2013] Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., and Zha, H. (2013). Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, New York, NY, USA. Association for Computing Machinery.

[Sutton and Barto, 2018] Sutton, R. and Barto, A. (2018). Reinforcement learning: An introduction (2nd ed). *MIT Press, Cambridge, MA, USA*.

[Vargas et al., 2016] Vargas, S., Sanz-Cruzado, J., Rossetti, M., and Ezzat, M. (2016). RankSys: Java 8 Recommender Systems framework for novelty, diversity and much more.

[Wang et al., 2018] Wang, Q., Zeng, C., Zhou, W., Li, T., Iyengar, S. S., Shwartz, L., and Grabarnik, G. (2018). Online interactive collaborative filtering using multi-armed bandit with dependent arms. *IEEE Transactions on Knowledge and Data Engineering*.

[Zhao et al., 2013] Zhao, X., Zhang, W., and Wang, J. (2013). Interactive collaborative filtering. In

Proceedings of the 22nd ACM international conference on Conference on information; knowledge management, CIKM '13, pages 1411–1420, New York, NY, USA. ACM.