UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Popularity Bias in Recommender Systems

Author:
Lorenzo Andrés VIGO DEL ROSSO

Supervisor:
Paula GÓMEZ DURAN

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2022

Contents

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc in Fundamental Principles of Data Science

Popularity Bias in Recommender Systems

by Lorenzo Andrés VIGO DEL ROSSO

The use of Recommender Systems has increased prominently in the recent past and research in this area has obtained great advances in terms of accuracy in recommendations and in the application of these systems on diverse data models. However, new concerns arise about the generated results. One does not only want the results to be accurate but also fair. Biases are one of the principal enemies of fairness. In this work, we will focus on popularity bias which causes that only popular items are recommended to the users, damaging diversity in the generated outputs. Popularity bias can also have negative effects in society due to the generation of filter bubbles and echo chambers. We will explain bias mitigation and leveraging techniques. Also, we will perform an analysis to characterize a dataset and to detect the bias amplification caused by several Recommender Systems. In order to achieve these goals, metrics and concepts will be defined during the analysis, giving special attention to the one named *Bias Effect*.

Chapter 1

Introduction

1.1 Recommender Sytems and Popularity Bias

Recommender Systems are a powerful tool that assists users in finding new items they might be interested in. Information about the past is presented in the shape of interactions or ratings between user and items, accompanied by properties of the users and items themselves and other kinds of relevant information such as interaction context. With these data, *Recommender Systems* are capable of inferring user interests (Aggarwal et al., 2016) and generating relevant recommendations for every user.

A basic model for a Recommender System may have good results in terms of accuracy. However, some limitations of these systems can affect negatively on the effectivity and on the quality of the recommendations. For instance, **Collaborative Filtering** Recommender Systems [1] suffer from the known as **cold-start problem**: during the early stages, when there are not enough observed ratings by the users, the system does not have the data needed to generate accurate recommendations.

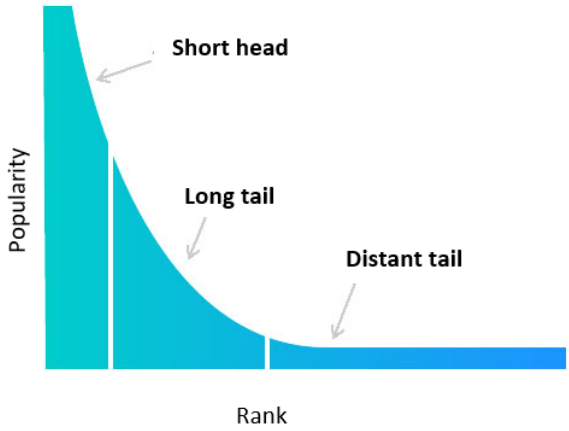

FIGURE 1.1: The long-tail of item popularity (Abdollahpouri, Burke, and Mobasher, 2019)

In this work, we are more concerned about the **long-tail problem** which is linked to the origin of *popularity bias*. Collaborative Filtering Recommender Systems tend to highlight popular items (those with a higher amount of ratings), while neglecting

[1] *Collaborative Filtering* refers to the sole use of ratings from multiple users in a collaborative way to predict missing rating (Aggarwal et al., 2016).

long-tail items (Abdollahpouri, Burke, and Mobasher, 2019).

It is natural to assume that items will not be equally consumed, but rather follow a Power Law distribution (Adamic and Huberman, 2000). This allows us to classify items in terms of their popularity. Following the division shown in 1.1, three item classes can be identified. *Short head items* are the most popular items: for example, in movie recommendation, blockbuster movies. *Long tail items* are those items that fail to be included in recommendations, while *distant tail items* are those items with such few ratings that any rating comparison involving them would be unreliable (Abdollahpouri, Burke, and Mobasher, 2019).

On one hand, short head items are by definition well-known and they generally will please a large number of users. On the other hand, recommending only popular items hinders item discovery and ignores the interests of users with niche tastes. It is desirable to take care of *niche users*, as the short head presents a higher density of popularity and interactions while the long and distant tail count with a larger aggregated share of the pie.

In general, recommender models trained on datasets that follow the long-tail distribution do not only imprint this distribution in their recommendations, but also sharpen it (Zhang et al., 2021). This effect is known as **popularity bias**.

1.2 Impact of Popularity Bias in Recommendations

For several reasons, popularity bias should be considered and mitigated in the recommendation process. One of the motives was presented in the previous section and is known as the **Matthew effect** (Liu et al., 2019): long-tail items do not have enough exposure to be recommended, leading to recommendations that are poor in diversity. Users that are already familiar with short head items will not find it useful to be constantly recommended more popular items. They will eventually discover those items through advertisements, social media or word-of-mouth, so it is against their needs to overexpose them to popular products or media.

One could also argue that exposing users exclusively to well-accepted items drives them to adapt a *herd mentality* (Zhang et al., 2021). It may not be desirable to push up a herd-like mentality in our society because it would provoke a shortage of personality and free will in its members and a lack of variety in art and culture.

In addition, popularity bias can provoke other concerning issues depending on the dataset being used. *Echo chambers* and *filter bubbles* are side effects of popularity bias that can have more negative impacts on society. **Echo chamber** describes the generation of social communities who share similar opinions within the group. **Filter bubble** is the phenomenon of an overly narrow set of recommenders isolating users in information echo chambers (Ge et al., 2020).

These two concepts are not isolated from each other since filter bubbles are a potential cause of echo chambers. These effects should be taken care of because of the ability of Recommender Systems to influence on user tastes, interests and beliefs. User interests may be reinforced due to repeated exposure to similar items or categories of items. This phenomenon might be dangerous in a political context, for

example, as it is not viable to simply classify items into those that portray positive or negative views.

Consequently, the effects of popularity bias should be analyzed and fought against not only in movie or music recommendation, but also in more sensitive contexts for the users, such as in information and news recommendations and in social networks.

1.3 Scope

In this work, we will firstly explore the State of the Art in popularity bias mitigation and leveraging. Several techniques will be presented alongside with sample methodologies that apply them.

Our contributions are focused on the detection of popularity bias within recommendations previous to the application of the mentioned methodologies. Particularly, we will work with the MovieLens-1M dataset[2] and with the following Recommender Systems: Random, Itempop, FM and FM-GCE. More details about them will be given in further chapters.

Popularity bias will be studied and analyzed after aggregating item and recommendation distributions in terms of movie genres. This procedure is extendable to other kind of datasets: hashtags or topic detection in tweets, topic in articles and news or genre in music and literature. It is more interesting to analyze how popularity bias is present in several groups of items rather than in each item single-handedly because it gives us a broader insight of user behavior.

To achieve our goal, we are required to define descriptive metrics that help us understand the presence and effects of popularity bias in the recommendations generated by each one of the models.

As future work, we aspire to apply the explained popularity bias mitigating methodologies on our models and check how effective they are through our metrics. Also, validating the utility of our metrics by extending the analysis to other datasets and models is more than desired.

[2] Available here: https://grouplens.org/datasets/movielens/1m/

Chapter 2

State of the Art

Several methodologies have been defined in existing work that help achieving Popularity Bias-Aware Recommender Systems. Some of the techniques apply *unbiased learning* and *ranking adjustment* in order to mitigate the effects of popularity bias.

2.1 Inverse Propensity Score (IPS)

Inverse Propensity Scoring consists in performing a reweighting in the interaction samples that will be used for the model training. The intuitive idea is to give those samples involving popular items a lower weight to compensate the long tail distribution in the dataset (Gruson et al., 2019): each sample will be reweighted through the inverse of the propensity score of the corresponding item.

There are theoretical foundations that show that an IPS loss is an unbiased estimator that eliminates the popularity bias. Nonetheless, the requirement of computing the *true* propensity scores adds a potentially high complexity to the process. Even though, there are approaches available that successfully compute adequate approximations. Also, IPS estimators suffer from a high variance problem (Gangwar and Jain, 2021).

2.2 Causal Embedding

The idea behind *Causal Embeddings* is to use bias-free data from a uniform distribution to make the model learn an unbiased embedding. Basically, users will be shown random items, automatically discarding item popularity.

The user experience may get damaged by receiving random recommendations but this risk can be controlled by balancing the number of recommendations that are generated randomly (Bonner and Vasile, 2018).

2.3 Ranking Adjustment

After recommendations are computed, *Ranking Adjustment* approaches purposely lower the ranking scores of popular items. Two examples of Ranking Adjustment methodologies are the following:

- In (Abdollahpouri, Burke, and Mobasher, 2017), a **regularization factor** in the objective loss function that fights for a 50/50 ratio between short head and long tail items is proposed. The concepts of *intra-list diversity* and the metric *Average*

Percentage of Tail items (APT) are involved in the process and are introduced in the cited work.

- **Popularity Compensation** (PC) debiasing algorithm (Zhu et al., 2021) is a methodology to correct the predicted scores of each item for a user. The algorithm takes care of considering both popularity and user tastes in a way in which the items that benefit the most from the compensation are those unpopular items that match the user tastes.

Based on *Result diversification*, we also have **Explicit Query Aspect Diversification** (xQuAD), which selects items iteratively by estimating how well a given document satisfies an uncovered aspect in an under-specified query (Abdollahpouri, Burke, and Mobasher, 2019). A personalization factor determined by each user's historical interest in long-tail items is also introduced. xQuAD re-ranks a recommendation list R for a given user u as a new list S according to the following criterion:

$$P(v|u) + \lambda P(v, S'|u) \tag{2.1}$$

The first term represents the likelihood of u being interested in an item v, while the second one is the likelihood of u being interested in v as an item that is still not included in the new generated list, S. The first term is in charge of the ranking accuracy and the latter is responsible for the diversity in recommendations between two categories: short head (Γ) and long tail (Γ').

$P(v, S'|u)$ can be reformulated through the likelihood of v being a part of each one of the categories (d) and the likelihood of the user being interested on an item depending on its category (Santos, Macdonald, and Ounis, 2010).

$$P(v, S'|u) = \sum_{d \in \{\Gamma, \Gamma'\}} P(v, S'|d) P(d|u) \tag{2.2}$$

There are modifications of xQuAD available, such as Binary xQuAD and Smooth xQuAD. The differences reside in how the first of the two likelihoods is computed.

2.4 Causal Intervention

Causal intervention allows us to leverage the popularity bias instead of eliminating it. This philosophy aligns with the idea that popularity biases have their own upsides: some items have higher popularity because of their quality. Therefore, blindly eliminating the popularity bias is getting rid of important signals implied in the data (Zhang et al., 2021).

By defining a *causal graph* (Pearl, 2009), the effect of item popularity on interaction probability can be explained. Causal graphs model causality by displaying directed edges $X \rightarrow Y$ between two variables X and Y if and only if X is a direct cause of Y.

Let U, I and Z be the user, item and item probability nodes respectively. Also, consider C as the interaction probability. It must be reasoned that Z affects both I and C. A high amount of users show **conformity**, considering popularity as a positive attribute when choosing an item to interact with. Also, a user can only interact with an item if they are exposed to it[1]. For these reasons, $Z \rightarrow C$. Then, since recommenders usually inherit the bias from the data, the popularity affects on the exposure of the items: $Z \rightarrow I$.

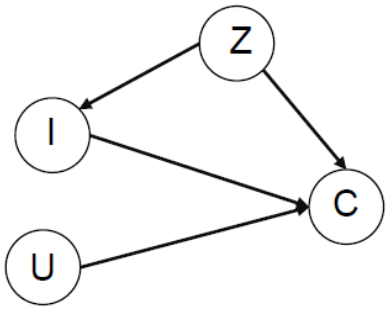

FIGURE 2.1: Causal graph that considers popularity in the recommendation process (Pearl, 2009)

Therefore, item popularity Z is a common cause for I and C. For this reason, Z acts as a *confounder* between the two variables.

Confounders are mostly described as the origin of 'mixing' or 'blurring' of effects (Jager et al., 2008). It is a term that is generally used in epidemiology and it refers to mistakenly measuring the effect of *confounding* variables. There are several classical structures in causal graphs that lead to confounding and the one shaped by the nodes Z, I and C is one of them (VanderWeele, 2019).

In this scenario, Z affects C through two different paths. First, in the desired path $Z \rightarrow C$. Second, in the path $Z \rightarrow I \rightarrow C$ that represents the bad effect of *bias amplification*. This causal path causes the unwanted recommendations that are based on popularity and that ignore the user interests. Causal intervention allows us to *intervene* in the graph by using the tools provided by *do-calculus* (Pearl, 2012). The goal is to get rid of the connection $Z \rightarrow I$ along with the negative effects of popularity bias.

Do-calculus is performed within a framework named **Popularity-bias Deconfounding and Adjusting** (PDA). It is applied during training and inference. By performing $P(C|do(U, I))$, the impact of the parents of U and I disappears. In this case, we are only getting rid of the effect of the path $Z \rightarrow I$ since U has no parents in the graph. The trick is to find an estimation of $P(C|do(U, I))$ in terms of probabilities in the original causal graph. This estimation should be found by using intermediate probabilities that stem from the intervened causal graph.

[1] In a probabilistic approach, consider Y to be a random variable that models the probability of a user u interacting with an item i. Also, consider a random variable O that models the probability of u being exposed to i. $Y_{ui} = 1$ is only possible when $O_{ui} = 1$.

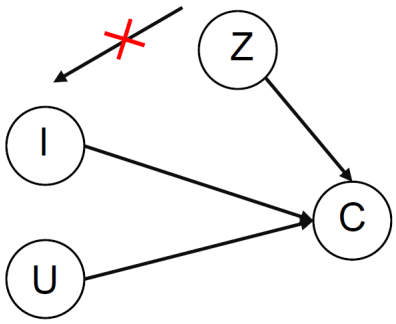

FIGURE 2.2: Desired causal graph after causal intervention (Pearl, 2009).

$$\begin{aligned} P(C|do(U,I)) &= P_{G'}(C|U,I) \\ &= \sum_z P_{G'}(C|U,I,z)P_{G'}(z|U,I) \\ &= \sum_z P_{G'}(C|U,I,z)P_{G'}(z) \\ &= \sum_z P(C|U,I,z)P(z) \end{aligned}$$

Proof and reasoning behind this mathematical development can be found in (Zhang et al., 2021). The term $P(C|U,I,z)$ can be estimated with any user-item matching model (such as Matrix Factorization (Bokde, Girase, and Mukhopadhyay, 2015)), an ELU' (Clevert, Unterthiner, and Hochreiter, 2015) and a smoothing parameter for item popularity, which allows leveraging or eliminating its effect. Finally, with this estimation, the value of $P(C|do(U,I))$ is computable, and therefore, the probability of a user interacting with an item after leveraging the impact of popularity bias.

2.5 Popularity Drift

It is intuitive to think that item popularity is dynamic and changes over time. An item might be very popular and receive numerous interactions at a certain point in time and then become less relevant. For that reason, it is logical to assume that the impact of popularity could also be dynamic over time. In (Zhang et al., 2021), a metric to quantify this effect is defined. Suppose the interaction dataset is divided in T stages over time, each one of which is represented by a $|I|$ dimensional vector where I is the bag of available items: $[m^t_1, ..., m^t_{|I|}]$. Each entry of the vector should represent the popularity of each item during the stage. The **Drift of Popularity** (DP) is defined as:

$$DP(t,s) = JSD([m^t_1, ..m^t_{|I|}], [m^s_1, ..m^s_{|I|}]) \tag{2.3}$$

JSD denotes the Jensen-Shannon Divergence, which is key to measure the similarity between two stages (Endres and Schindelin, 2003). A higher value denotes a higher popularity drift.

The number of interactions can only increase with time. Therefore, with conventional Recommender Systems, an item can only increase in popularity during time. In addition, those Recommender Systems that propagate popularity bias will provoke that popular items increase in popularity constantly. For this reason, it might be interesting to consider weighting the interactions in the train set in terms of their antiquity.

Chapter 3

Dataset Analysis for Popularity Bias Identification

3.1 Dataset Introduction and Preprocessing

For our analysis, the dataset used will be MovieLens-1M[1]. It provides a set of real interactions among over 6000 users and 4000 movies. Both users and movies are complemented with information about them. In the case of users, information such as the user age, gender and profession is included. Meanwhile for items, one can see the title, genres and year of release. We filtered out those users and movies which could be classified as distant tail entities or, in other words, those without enough interactions (our personal choice: 20 and 5 interactions respectively).

We will not only consider the popularity bias in terms of item popularity, but also based on genre popularity. MovieLens 1M provides more than one genre label for each movie. This property made it more difficult to reach to conclusions as a specific unpopular genre could be seen benefiting from the popularity bias from a popular genre because some items would share both genres. For this reason, we had to preprocess the dataset and only keep the least popular genre for each item. We decided to keep the least common genre because we assume that said property provides the most relevant information to tell items apart, even though it may not be the most accurate description for some movies.

The dataset is split into a training and a testing set following the Time-Aware Leave One Out strategy per user. That is, the last interaction from each user is kept as a test sample. This practice is usual in Recommender Systems and, in our case, it is possible because each interaction comes along with a timestamp. In further experiments, we will increase the number of interactions that are kept as test samples to analyze the effect of Popularity Drift. In this context, we will refer to the data splitting technique as *Leave N Out strategy*.

Also, the test set for each user will not include the movies the user already interacted with. Consequently, the test set will be different for each user but no user will be recommended a movie they have already watched.

Figures 3.1 and 3.2 show the distribution of items in terms of genre before and after (respectively) keeping only the least popular genre for each item. Take into account that it is more than logical for all percentages to sum up over 100% when each

[1] Available here: https://grouplens.org/datasets/movielens/1m/

item has more than one genre.

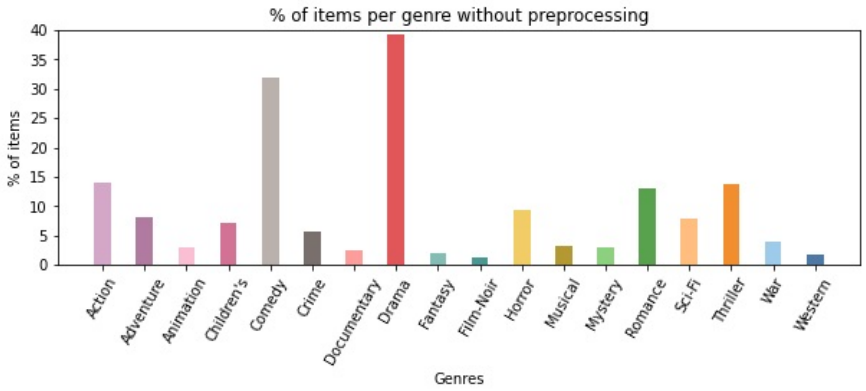

FIGURE 3.1: Percentage of items per genre with more than one genre per item.

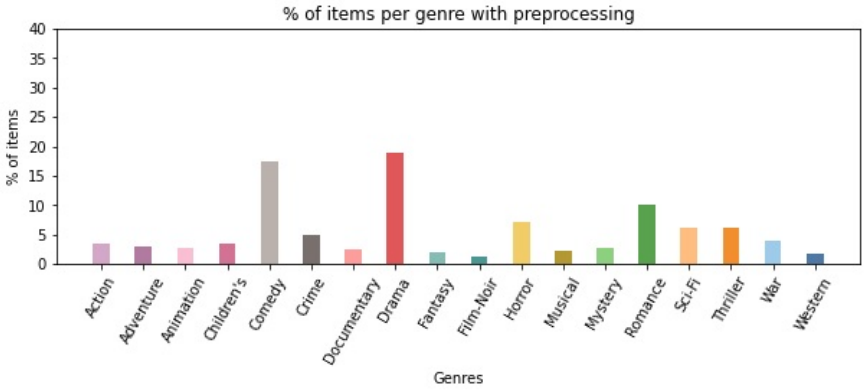

FIGURE 3.2: Percentage of items per genre with only one genre per item.

Figure 3.3 shows the number of training set interactions by item, which have been sorted depending on their popularity. Also, each item has its bar colored according to its genre. In this visualization, it can be checked that our training set follows the long tail distribution. Also, in further observations, it will be visible that certain genres show higher density of items in specific zones of the distribution.

3.2 Model Introduction

In order to compute recommendations, a recommender model is required. In our research, we will use four different models so that it can be compared how each one of them imprints the bias into their recommendations.

- **Factorization Machine** (FM): Collaborative Filtering Recommender System, more information can be found in (Rendle, 2010).
- **Factorization Machine with Graph Convolutional Embeddings** (FM-GCE): Enriched version of Factorization Machines that uses Graph Convolutional Networks (GCN) instead of the regular Embedding Layer (Gómez Duran et al., 2021).

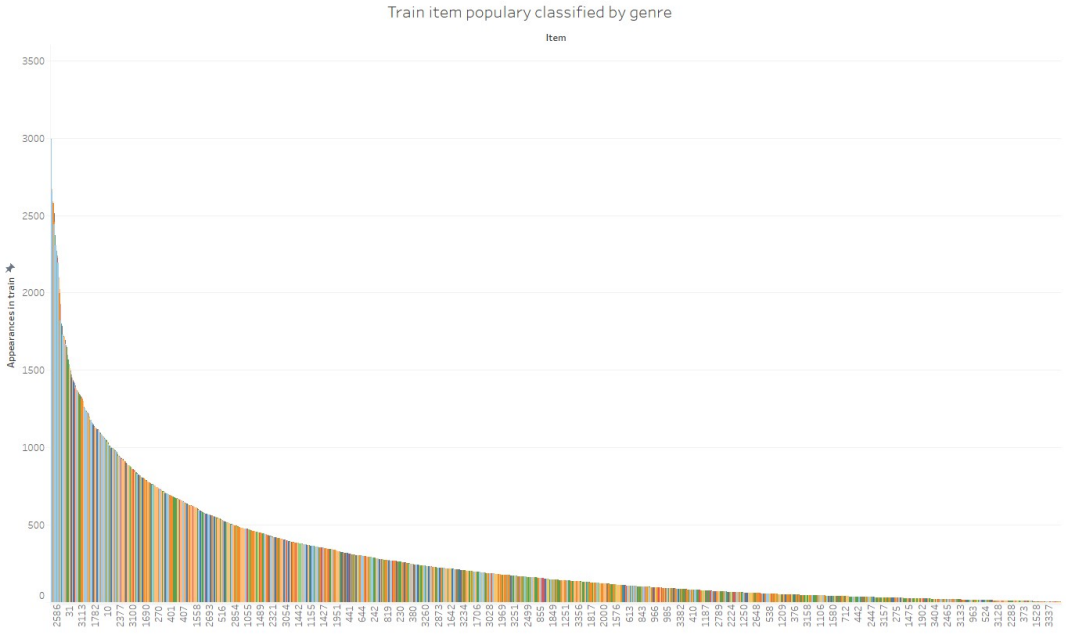

FIGURE 3.3: Items sorted by train popularity and colored according to genre.

- **Itempop**: Given an integer k, Itempop always recommends the k most popular items to each user. Take into account that in our split dataset, a user can not be recommended a movie they have already interacted with. Therefore, these k most popular items will not be the same for each user.
- **Random**: This model assumes that users are random and are therefore unbiased. The recommendations will be k random available items, where k is an integer.

3.3 Interaction Dataset Characterization

Users are agents that take decisions consciously. These decisions are rational to some extent, but are also inadvertently influenced by other factors such as the popularity bias. If users were completely unbiased and irrational, they would choose the items they interact with by random. This the main assumption of the Random model, as explained in the previous section.

The Random model portrays this assumption into its computed recommendations: each user will be recommended items in a random way. Therefore, disregarding minor differences caused by how the test set is generated, if we aggregate the recommendations in terms of the item genre, it will be noticeable that the recommendation distribution will match the distribution displayed in 3.2. This comparison can be visualized in 3.4.

However, as we stated, users are not random. For this reason, the interactions in the training set and the recommendations follow different distributions, as it can be seen in 3.5. This difference is a univocal characterization of our interaction dataset.

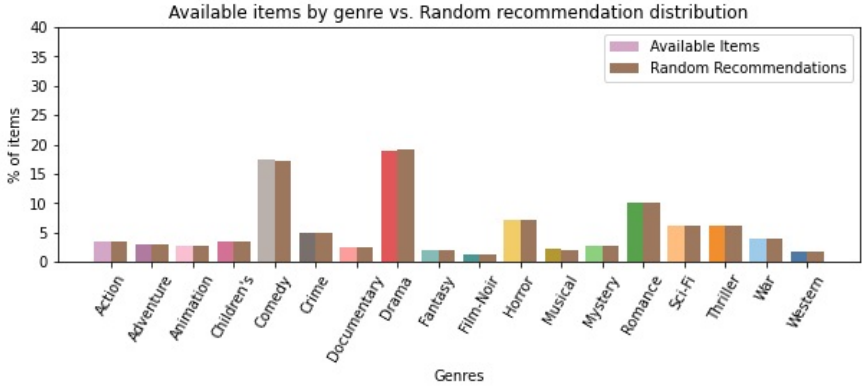

FIGURE 3.4: Comparison between the distribution of the available items in terms of the genre vs. the same distribution for the random recommendations.

From this phenomenon we can conclude that our users are not random, but conscious.

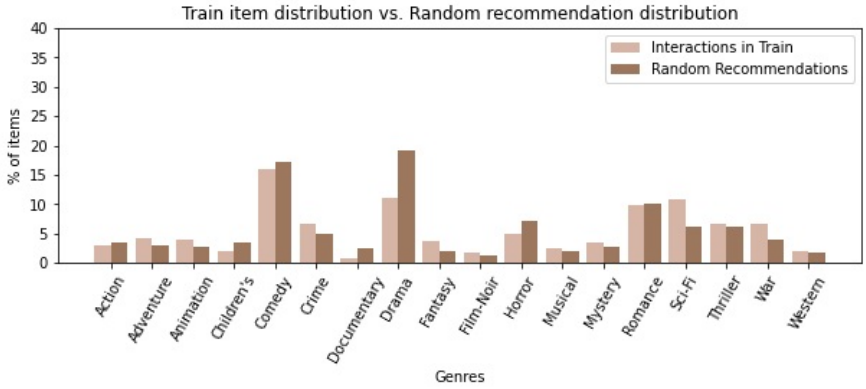

FIGURE 3.5: Comparison between the distribution of the interactions in the training set in terms of the genre vs. the distribution for the random recommendations in terms of genre.

3.4 Popularity Bias in Interaction Train Set

The **popularity of an item** can be defined as the normalized amount of interactions per item in which the given item is involved minus the expected popularity of an item. This expectancy can be approximated by assuming that the users are random, therefore, for an item i:

$$popularity(i) = \left(\frac{\#\{interactions\ with\ i\}}{\#\{interactions\}} - \frac{1}{\#\{items\}} \right) \#\{items\} \tag{3.1}$$

If this measure was visualized, the result would be identical to 3.3 but with a different range in the vertical axis. A notion of how distant from being random the

users are can be computed after defining the popularity. We will name it **consciousness of the users** and it will consist in the squared root of the sum of the squared item popularities. This measure should not be computed after the genre aggregation. For a set I of items,

$$consciousness(I) = \sqrt{\sum_{i \in I} popularity(i)^2} \tag{3.2}$$

The closer to 0 the consciousness is, the more randomly the users have been acting. In the experiment with the used dataset, that is, with $I = \#\{items\ in\ MovieLens\text{-}1M\}$:

$$consciousness(I) \approx 6117,47$$

It is visible that the genres with the most amount of items are those with the highest amount of interactions in train. That is, the more items a genre has available, the more popular this genre tends to be. Nonetheless, such is not the only factor to be taken into account. Drama is the most common genre among items but is less popular among users than Comedy. In fact, the aggregated popularity of Drama is almost the same as the one of Sci-Fi, which is a genre with much less available items. One can conclude then that on average the Sci-Fi items are more popular than those classified as Drama.

However, a genre does not need to be more popular on average to benefit from popularity bias. A genre *benefits* from popularity bias when it gets more recommended than it is expected. The expected amount of recommendations follows the random recommendation distribution. For this definition to be applied, the recommender should suffer from said bias. In particular, this definition is not applicable in the context of the Random recommender in 3.5. In order to analyze how genres might benefit from popularity bias, we will introduce the distribution of the recommendations generated by Itempop.

Itempop@k, as defined in previous sections, returns the top k most popular items available for each user as recommendations. Assuming $k = 1$, the initial thought would be that the same item will be recommended to all users. However, due to the structure of our test set, where users can not be recommended items they have already interacted with, the distribution of recommendations for Itempop@1 is the one displayed in 3.6.

Instead of having only one single item recommended, it is perceivable that several genres contain an item that has been recommended to some users. Documentary and Drama both have the most popular items that have generally not been consumed by the users. The amount of genres with no recommendations will decrease if we increase the value of k.

From now on, we will be working with the recommendations computed by Itempop@31, which distribution can be checked in figure 3.8.

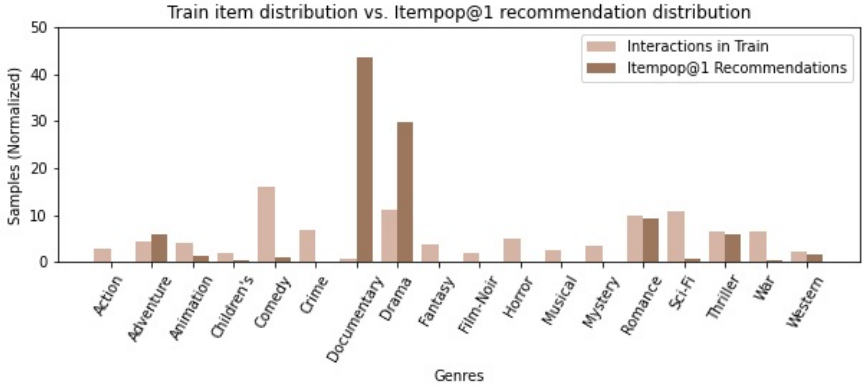

FIGURE 3.6: Comparison between the distribution of the interactions in the training set in terms of the genre vs. the distribution for the Itempop@1 recommendations in terms of genre.

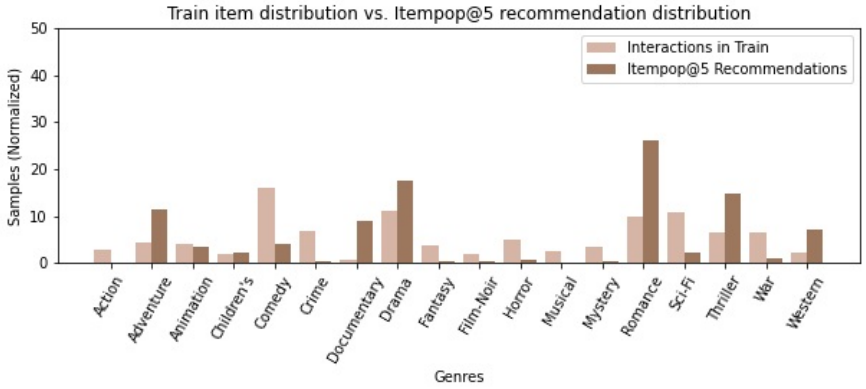

FIGURE 3.7: Same comparison as in 3.6 with a higher value of k: 5

It is trivial that the Itempop@31 is popularity biased. Therefore, it can be said that Drama and Romance are benefiting from popularity bias in its recommendations. Meanwhile, Sci-Fi does not benefit from bias despite being one of the most popular genres on average. In order to justify this scenario, it is needed to define some notions concerning the position of the items of each genre in the item popularity ranking (3.3).

First, the **aggregated popularity bias of a genre** g of a given dataset D for a popularity biased model M can be defined as:

$$agg\_genre\_pop_M(g) = \frac{\#\{recommendations\ within\ g\}}{\#\{recommendations\}} - \frac{\#\{interactions\ within\ g\}}{\#\{interactions\}} \tag{3.3}$$

The larger and positive this value is, the more g benefits from the popularity bias. For example, in our escenario with $g = Drama$:

$$agg\_genre\_pop_{Itempop@31}(g) \approx 4.85$$

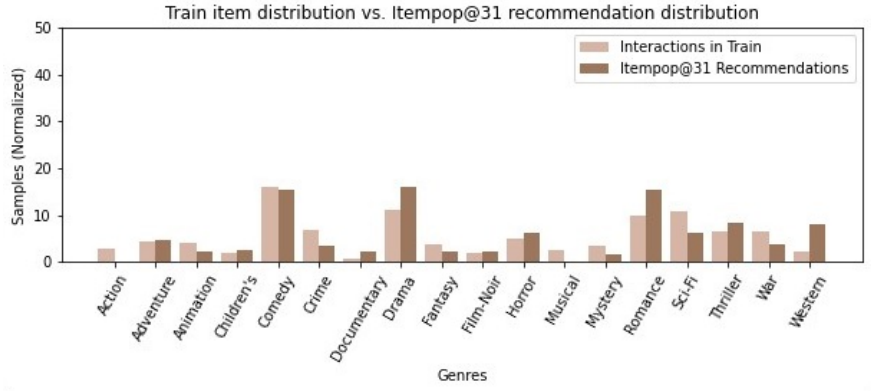

FIGURE 3.8: Comparison between the distribution of the interactions in the training set in terms of the genre vs. the distribution for the Itempop@31 recommendations in terms of genre.

Considering the aggregated popularity bias of all genres at once can give us an idea of the **bias effect** on our recommendations. Given a categorization of the available items G (in our case, genres):

$$bias\_effect_{M,eucl}() = \sqrt{\sum_{g \in G} agg\_genre\_pop_M(g)^2} \tag{3.4}$$

In our case of study:

$$bias\_effect_{Itempop@31,eucl}() \approx 12.64 \tag{3.5}$$

Using the Euclidean distance for such kind of metric might be misleading as completely opposite distributions in recommendations can return the same numerical value. For this reason, it may be interesting to compute the same metric with respect to the cosine similarity. Analogously, *bias effect* can be computed in terms of any desired distance or similarity.

To use similarities, the following vectors should be defined. In Python notation, for a model M:

$$r_M = \left[\frac{\#\{recommendations\ within\ g\}}{\#\{recommendations\}} \ for\ g\ in\ G \right]$$
$$d = \left[\frac{\#\{interactions\ within\ g\}}{\#\{interactions\}} \ for\ g\ in\ G \right]$$

Then, to compute the bias effects in terms of the cosine similarity:

$$bias\_effect_{M,cos}() = cosine\_similarity(r_M, d) \tag{3.6}$$

The bias effect will move in different ranges of values depending on the distance or similarity applied. This should be taken into account before comparing the bias effects if different computations are used. In our current example:

$$bias\_effect_{Itempop@31,cos}() \approx 0.0829 \tag{3.7}$$

Evidently, this value is very different from the one previously obtained even though both values represent the same reality.

Next, one can visualize where the items of a specific genre fall in the popularity ranking by filtering in 3.3. If this query is performed for Drama, Romance and Sci-Fi respectively, the following visualizations are generated: 3.9, 3.10 and 3.11.

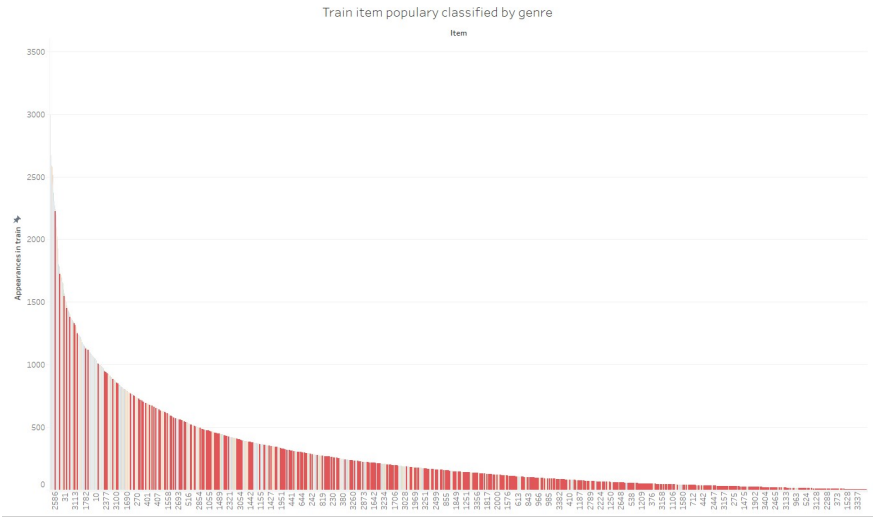

FIGURE 3.9: Drama item placing in item popularity ranking.

It is surprising to see that Sci-Fi has the greatest density of items at the very beginning of the graph while Drama and Romance benefit more from the popularity bias than Sci-Fi does.

In order to intuitively justify this, it is convenient to interpret that Itempop@k works in the following way. First, declare an empty array of length k for each user, where their recommendations will be stored. The items are visited in descending popularity order and added to all the arrays that are not already full and do not correspond to users that have already interacted with the item. Fixing $k = 31$, let's focus in the top 31 popular items. All of those items would be recommended to all the users if it was valid to do so. However, the most popular item is a valid recommendation to less users than the 31^{st} most popular item. For that reason, the probability for an item to be recommended increases in the very beginning of the

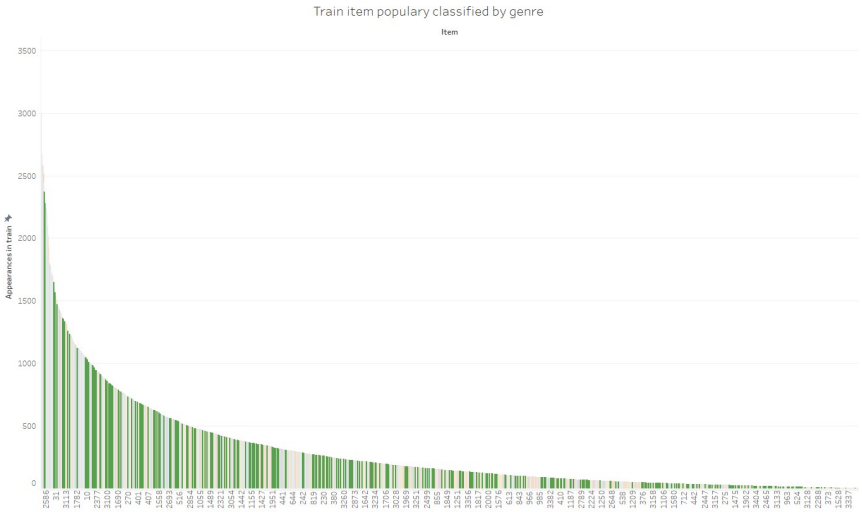

FIGURE 3.10: Romance item placing in item popularity ranking.

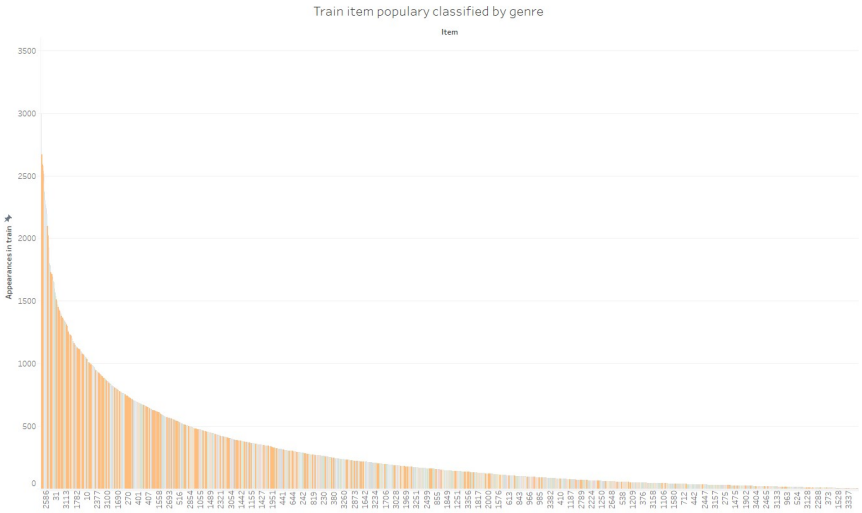

FIGURE 3.11: Sci-Fi item placing in item popularity ranking.

graph.

This effect drags on for some more items depending on the amount of users that have interacted with several items among the most popular items. Then, when most user arrays start getting full, the probability massively decreases towards 0 until all arrays are completed.

To sum up, popularity bias in our dataset with Itempop benefits those genres that accumulate the most items in the final section of the short head of our popularity item graph, but not those genres that do so in the very beginning of the same graph. There are some concepts that can be interesting in order to study this effect deeper.

As it important to know where the most items of a genre are concentrated in the long tail distribution, computing the density of the appearances in the distribution can help reach to conclusions. This measure can be visualized by reorganizing the

popularity ranked long tail distribution in 3.3 in bins and then applying a **Gaussian Kernel Density Estimation**.

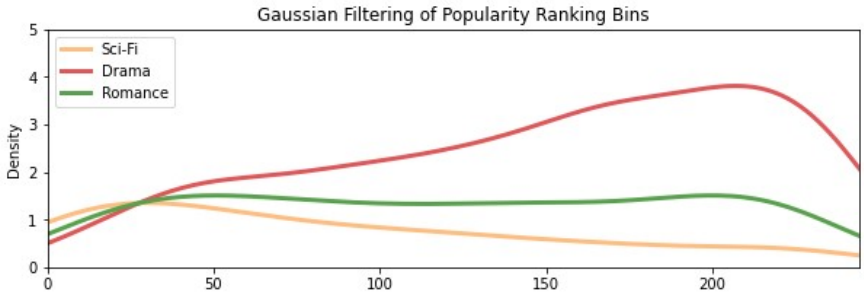

FIGURE 3.12: Gaussian Filtered Density of Binned Frequency of Genres in Popularity Ranking

In figure 3.12, it can be seen that Sci-Fi genre presents more density at the very beginning of the ranking, but it is soon surpassed by both Drama and Romance.

Numerically, density can be estimated by calculating the normalized amount of items of a genre r between two positions in the ranking j, k. The genres that will benefit the most from popularity bias in Itempop are those with a higher density in the final zone of the short head. Given a function $rank(i)$ that returns the popularity ranking position of an item i,

$$density(g, j, k) = \frac{\#\{items\ i\ of\ genre\ g \mid j \leq rank(i) \leq k\}}{|k - j|} \tag{3.8}$$

3.5 Popularity Bias Aggravation by Collaborative Filtering Recommender Systems

It is interesting to use Itempop to analyze the inherent popularity bias found in the dataset, as this recommender directly imprints the popularity of items and genres into its recommendations. However, it is crucial to analyze how Collaborative Filtering Recommender Systems can aggravate the implicit bias that stems from interaction data. To perform this study, we will employ a Factorization Machine as model. We will also analyze how adding a Graph Convolutional Embedding to the Factorization Machine model (FM-GCE) impacts the bias effect in the recommendations.

It is noticeable at first that, opposed to those results generated by Itempop@31, the recommendations are distributed along all available genres. Most popular genres still benefit from bias, including Comedy in this occasion. Sci-Fi keeps on being the main exception. Numerically, the bias effect is reduced to two thirds of its value with Itempop.

$$bias\_effect_{FM,eucl}() \approx 8.29 \tag{3.9}$$

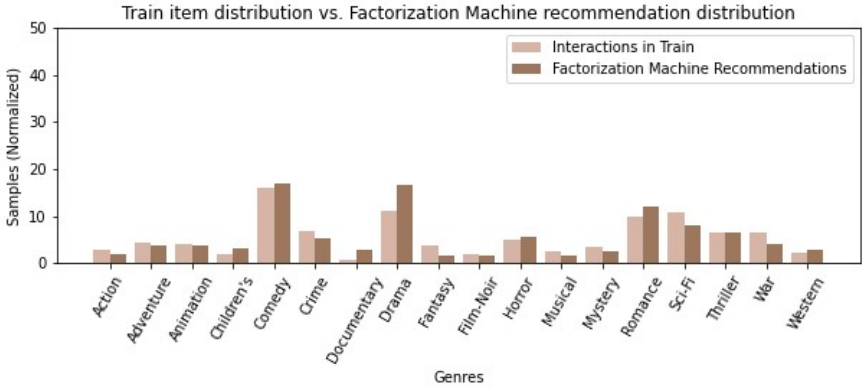

FIGURE 3.13: Comparison between the distribution of the interactions in the training set in terms of the genre vs. the distribution for the Factorization Machine recommendations in terms of genre.

By computing the bias effect with cosine similarity, a value very closer to zero is obtained. In following sections, a table with all results will be displayed for easier comparisons.

$$bias\_effect_{FM,cos}() \approx 0.0361 \tag{3.10}$$

The impact of adding Graph Convolutional Embeddings into Factorization Machines can also be analyzed.

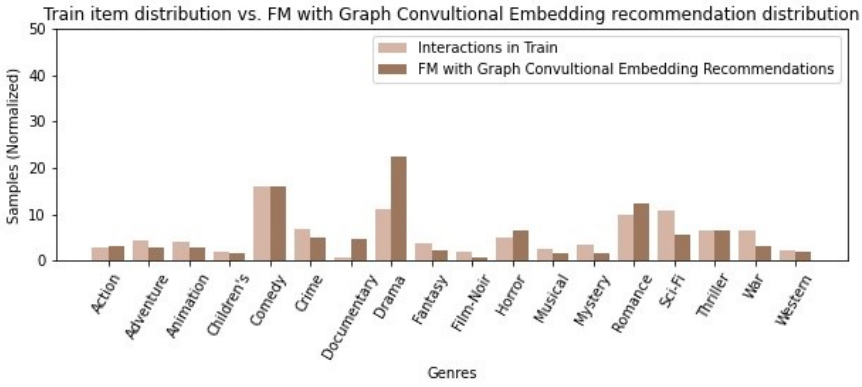

FIGURE 3.14: Comparison between the distribution of the interactions in the training set in terms of the genre vs. the distribution for the FM with GCE recommendations in terms of genre.

All genres are still considered within the recommendations. However, Drama gets a higher benefit from popularity bias than before, surpassing Comedy as most recommended genre. Sci-Fi suffers more than before. Finally, most minority genres are more damaged, except for Documentaries. In fact, the bias effect provoked by Factorization Machines with Graph Convolutional Embeddings is worse than

the one obtained by the original Factorization Machine, without surpassing Itempop@31:

$$bias\_effect_{FM-GCE}() \approx 8.36 \tag{3.11}$$

As expected, the same conclusion can be reached by computing the bias effect with cosine similarity.

$$bias\_effect_{FM-GCE}() \approx 0.0370 \tag{3.12}$$

A deeper research on the causes of this result should be carried out. However, our initial theory concludes that the strong links between similar nodes within the graph structure are to blame for the worsening in terms of bias effect. We assume that popular items will be identified as very similar by the embedding because of the huge amount of paths between them provoked by the amount of users that have seen both of them.

3.6 Popularity Drift and Bias Effect

As explained in 2, popularity is dynamic during time and therefore it is natural to expect that the bias effect will change with time too. In order to analyze this, we will compute the bias effect in different points in time. In order to do that, we will apply different *Leave N Out* (LNO) strategies during the data splitting stage. The behavior of these strategies is explained in the beginning of this chapter.

From now on and for simplicity, all bias effects will be computed through the cosine similarity.

Cosine Similarity Bias Effects for different LNO strategies			
LNO Strategy	Itempop	FM	FM-GCE
LOO	0.082904	**0.036112**	**0.036980**
L2O	**0.082867**	0.037565	0.037625
L3O	0.082898	0.038824	0.037359
L4O	0.082905	0.037450	0.038232
L5O	0.082899	0.037808	0.038429
L10O	0.082897	0.039401	0.038440

As concluded previously, Factorization Machine is the best model out of the three in terms of handling popularity bias. This conclusion is consistent along all Leave N Out strategies. It is curious how both Factorization Machine implementations reach the least bias effect in the last available point in time. In general this metric does not vary much in the latest points in time. This is the expected behavior, as exposed in (Zhang et al., 2021).

Popularity bias is expected to grow noticeably in the earlier stages and then increase more slowly in the later points in time. That is why it is interesting to move

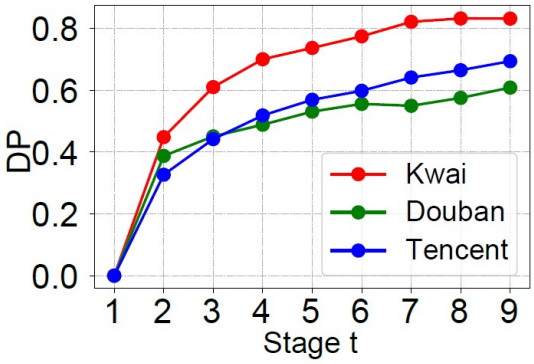

FIGURE 3.15: Popularity Drift along successive stages in time (Zhang et al., 2021). Datasets used do not match with our used datasets.

also to earlier stages in our recommendations.

We attempted this by applying the Leave 10 Out strategy. However, most of the users have a larger amount of interactions and moving 10 interactions towards the past does not make a great difference. Sadly, we can not leave more than 10 interactions out because some of the users have only 20 interactions and their recommendations could be severely damaged.

Apart from considering the popularity bias in the recommendations, we should also take into perspective the quality of said results. We expect to identify a trade-off between bias-awareness and accuracy. In order to assess the quality of the recommendations, the following metrics are introduced:

- **Hit Ratio** (HR@k): Computes the ratio of test samples among the top k recommendations for each user.
- **Normalized Discounted Cumulative Gain** (NDCG@k): Evaluates the positions in which the test samples are included in the top k recommendations for each user, giving higher scores to higher positions.
- **Coverage**: Analyses the diversity among recommended items. The higher, the more diverse.
- **Inverse Propensity Score (IPS)**: Describes the popularity of the items that are recommended. The lower, the more popular.

Fixing $k = 10$ for all metrics and evaluating them for each model in each LNO strategy we have applied, we may display the results in the following tables:

Metrics for all models in LOO strategy				
Model	HR	NDCG	Coverage	IPS
Random	0.0033	0.0019	**1.00**	**0.8667**
Itempop	0.0432	0.0216	0.06	0.0070
FM	0.0788	0.0404	**0.61**	**0.1293**
FM-GCE	**0.0841**	**0.0411**	0.57	0.1289

Metrics for all models in L2O strategy				
Model	HR	NDCG	Coverage	IPS
Random	0.0031	0.0017	**1.00**	**0.9055**
Itempop	0.0430	0.0215	0.06	0.0070
FM	0.0985	0.0472	**0.65**	**0.1410**
FM-GCE	**0.1224**	**0.0606**	0.58	0.1297

Metrics for all models in L3O strategy				
Model	HR	NDCG	Coverage	IPS
Random	0.0034	0.0015	**1.00**	**0.9228**
Itempop	0.0425	0.0209	0.06	0.0070
FM	0.1137	0.0561	**0.64**	**0.1401**
FM-GCE	**0.1414**	**0.0719**	0.59	0.1287

Metrics for all models in L4O strategy				
Model	HR	NDCG	Coverage	IPS
Random	0.0024	0.0011	**1.00**	**0.9334**
Itempop	0.0415	0.0206	0.06	0.0070
FM	0.1217	0.0600	**0.63**	**0.1371**
FM-GCE	**0.1466**	**0.0746**	0.60	0.1294

Metrics for all models in L5O strategy				
Model	HR	NDCG	Coverage	IPS
Random	0.0031	0.0015	**1.00**	**0.9400**
Itempop	0.0426	0.0211	0.06	0.0070
FM	0.1317	0.0650	**0.64**	**0.1376**
FM-GCE	**0.1560**	**0.0790**	0.59	0.1233

Metrics for all models in L10O strategy				
Model	HR	NDCG	Coverage	IPS
Random	0.0029	0.0013	**1.00**	0.**9581**
Itempop	0.0447	0.0225	0.06	0.0070
FM	0.1591	0.0801	**0.6358**	**0.1315**
FM-GCE	**0.1830**	**0.0951**	0.63	0.1287

Before jumping to conclusions, we will point the obvious. The Random Recommender is the best one in terms of diversity in its recommendations and evidently, it also the one that recommends the highest amount of unpopular items. In exchange, the quality of the recommendations is really low.

Leaving the Random recommender a little apart in our conclusions and considering it just a baseline reference for some metrics, we can conclude the following ideas. The Factorization Machine with Graph Convolutional Embedding consistently is the best model in terms of the metrics that evaluate accuracy (HR) and ranking quality (NDCG). Meanwhile, the original Factorization Machine is the best performing model in terms of diversity (Coverage) and recommending more unpopular items (IPS). This last conclusion is consistent with the analysis carried out through the Bias

Effect metric defined in this work.

However, the differences in Coverage and IPS between FM and FM-GCE are not that prominent. So, one could consider giving up a little of popularity bias awareness in exchange for better recommendations in terms of accuracy and ranking. This is the trade off between popularity bias handling and accuracy that we were expecting to find.

Chapter 4

Conclusions

4.1 Final Observations

In this work, we have introduced the concept of popularity bias, its negative effects and motivations to prevent it. Also, we presented several techniques and methodologies oriented to popularity bias mitigation and leveraging. Among them, one can find the *Inverse Propensity Score* (IPS), *Causal Embeddings*, Ranking Adjustment techniques such as *Popularity Compensation* (PC) and *Explicit Query Aspect Diversification* (xQuAD) and lastly, Causal Intervention through do-calculus application, also named *Popularity-bias Deconfounding and Adjusting* (PDA).

Afterwards, we performed an analysis on the MovieLens-1M dataset targetted towards popularity bias detection within the interaction samples provided. Next, we moved onto the generation of recommendations performed by several models: *Random*, *Itempop*, *Factorization Machine* (FM) and *Factorization Machine with Graph Convolutional Embeddings* (FM-GCE).

We defined some metrics to characterize our dataset and to detect and quantify the bias amplification effect within the recommendations given by the used models. Those metrics receive the names of *popularity* of an item, *consciousness* of the users, *aggregated popularity bias* of a genre and most importantly, the *bias effect* of a model.

Later, we applied the concept of *popularity drift* through several Leave N Out Strategies (LNO). This allowed us to compare the bias effects of all the models in several points in time of our dataset. We also evaluated several metrics concerning the accuracy (*Hit Ratio*, HR), the ranking quality (*Normalized Discount Cumulative Gain*, NDCG), the diversity (*Coverage*) and the popularity (IPS) of the recommendations.

Finally, we concluded that from the used models, the original implementation of the Factorization Machine is the model that least amplifies the popularity bias in its recommendations. This conclusion could be drawn from both the bias effect analysis and from the evaluated metrics. The equivalence in both analysis reinforces our definition of the bias effect.

Also, it is worth noting that Factorization Machines with Graph Convolutional Embeddings provide more accurate recommendations with better ranking without a great increment in bias amplification. We expected this kind of phenomenon as we were prepared to find a trade-off between bias awareness and recommendation accuracy.

4.2 Future Work

Principally, our goal in future work would be applying the bias mitigation and leveraging techniques and check their impact by comparing the bias effect in recommendations before and after their application.

Also, we would aspire to extend our analysis to further distance and similarity functions, to more models and to a higher amount of diverse datasets.

Finally, the popularity drift section of our analysis could be improved by filtering out users more agressively so that it is possible to move more backwards in time.

Bibliography

Abdollahpouri, Himan, Robin Burke, and Bamshad Mobasher (2017). "Controlling popularity bias in learning-to-rank recommendation". In: *Proceedings of the eleventh ACM conference on recommender systems*, pp. 42–46.

— (2019). "Managing popularity bias in recommender systems with personalized re-ranking". In: *The thirty-second international flairs conference*.

Adamic, Lada A and Bernardo A Huberman (2000). "Power-law distribution of the world wide web". In: *science* 287.5461, pp. 2115–2115.

Aggarwal, Charu C et al. (2016). *Recommender systems*. Vol. 1. Springer.

Bokde, Dheeraj, Sheetal Girase, and Debajyoti Mukhopadhyay (2015). "Matrix factorization model in collaborative filtering algorithms: A survey". In: *Procedia Computer Science* 49, pp. 136–146.

Bonner, Stephen and Flavian Vasile (2018). "Causal embeddings for recommendation". In: *Proceedings of the 12th ACM conference on recommender systems*, pp. 104–112.

Clevert, Djork-Arné, Thomas Unterthiner, and Sepp Hochreiter (2015). "Fast and accurate deep network learning by exponential linear units (elus)". In: *arXiv preprint arXiv:1511.07289*.

Endres, Dominik Maria and Johannes E Schindelin (2003). "A new metric for probability distributions". In: *IEEE Transactions on Information theory* 49.7, pp. 1858–1860.

Gangwar, Ajay and Shweta Jain (2021). "An Adaptive Boosting Technique to Mitigate Popularity Bias in Recommender System". In: *arXiv preprint arXiv:2109.05677*.

Ge, Yingqiang et al. (2020). "Understanding echo chambers in e-commerce recommender systems". In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 2261–2270.

Gómez Duran, Paula et al. (2021). "Graph Convolutional Embeddings for Recommender Systems". In: *arXiv e-prints*, arXiv–2103.

Gruson, Alois et al. (2019). "Offline evaluation to make decisions about playlistrecommendation algorithms". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 420–428.

Jager, KJ et al. (2008). "Confounding: what it is and how to deal with it". In: *Kidney international* 73.3, pp. 256–260.

Liu, Yudan et al. (2019). "Real-time Attention Based Look-alike Model for Recommender System". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2765–2773.

Pearl, Judea (2009). *Causality*. Cambridge university press.

— (2012). "The do-calculus revisited". In: *arXiv preprint arXiv:1210.4852*.

Rendle, Steffen (2010). "Factorization machines". In: *2010 IEEE International conference on data mining*. IEEE, pp. 995–1000.

Santos, Rodrygo LT, Craig Macdonald, and Iadh Ounis (2010). "Exploiting query reformulations for web search result diversification". In: *Proceedings of the 19th international conference on World wide web*, pp. 881–890.

VanderWeele, Tyler J (2019). "Principles of confounder selection". In: *European journal of epidemiology* 34.3, pp. 211–219.

Zhang, Yang et al. (2021). "Causal intervention for leveraging popularity bias in recommendation". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 11–20.

Zhu, Ziwei et al. (2021). "Popularity-opportunity bias in collaborative filtering". In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pp. 85–93.