UNIVERSITEIT VAN AMSTERDAM

MSc Artificial Intelligence
Master Thesis

Understanding Algorithmic Bias and Unfairness in Dynamic Recommendation

by
Emmanouil Rerres
13439022

48 EC
November 2021 - November 2022

Supervisor:
Dr. M.Mansoury

Examiner:
Dr. M.Mansoury

Second reader:
Dr. A.Yates

UNIVERSITEIT VAN AMSTERDAM

Contents

Abstract

Recommendations should serve the needs of all their partakers. Not only should they try to maximize user's utility, but also take responsibility for the notion of fairness they provide towards the objects they rank. This work aims to capture the impact of $Fairness$ of $Exposure$ in $Rankings$ framework as a bias mitigation and fairness of exposure allocation technique, under a dynamic and implicit user feedback setting. To achieve this, two recommender system pipelines are proposed. The one used as a baseline, generates its final recommendations solely based on $Bayesian\ Personalized\ Ranking$. The second one, also employs $FOEIR$ post-process fairness recommendation algorithm. We observe, that although $FOEIR$ mitigates various forms of biases in the short run, past the $120th$ recommendation round mark, it overexposes popular objects by 50% more on average, than the non-popular ones. Stricter constraints should be adopted to ensure fairness on an object level, under a dynamic temporal recommendation setting.

Chapter 1

Introduction

1.1 History of Recommender Systems

Even way before the dawn of the computer era, the concept of recommendations existed in every form of society. People during the Prehistoric ages used to consult their community peers when facing a dilemma that required a decision. Which tool is better for lighting a fire? During hunting, which route should they follow, not to allow their quarry to elude? In the time of Ancient Civilizations recommendations were reduced in the context of which seed should we plant, how often should we water the crops or when should the farming yield be picked up?

The progress of civilization and the development of more advanced and organized social structures raised a series of more sophisticated matters. The notion of the division of labor was introduced. The separation of the tasks allowed community participants to specialize and gain expertise in distinct fields. People now seek suggestions from members of society based on the role they fulfill. Kings and monarchs were seeking advice from people that they kept close to their court, regarding political and economic decisions. According to their suggestions, policies were formed and adopted [53].

A series of Industrial Revolutions led to an unprecedented technological and scientific advancement, allowing manufacturing capacity to vastly increase [9]. Mass production of standardized goods made people to give up their old habits of preparing their own food from raw materials, patching or customly making their clothes or even fixing their car or other electronic devices, by themselves. They start to consume products and services produced cost efficiently in large quantities under a normalized format that ensures a consistent quality output. The consequence movement of mass consumption brought individuals, on a daily basis, in front of a series of decisions that required their attention. Which company's goods should they acquire to satisfy their needs best [24]? At a first glance the emergence of the computer and the world wide web seemed to even further escalate the aforementioned phenomenon. In reality though, they provided a way to process information in a methodical and time efficient manner.

During the second half of 1970s a research group from the University of Duke, developed *Usenet*, a communication system that allowed its users to share e-documents and organize them on various subgroups, relevant to their topic for an easier search [30]. A bit later in the decade, *Grundy* was introduced. A computer based Librarian that after a short interview, takes into account user preferences in order to classify him into a specific predefined group and suggest him the appropriate books [23]. The first automated system for recommendations with a commercial use, is considered to be *Tapestry*. Developed in 1992 at the Xerox Palo Alto Research Center, users rate messages and comments related to a document and the most liked ones are used as recommendations [48].

The expansion of E-commerce in combination with a shift in the people's consumption habits from the homogenous standardized products to more customized and diverse ones, were the key

elements of the broad use and the establishment of recommendation systems [48]. J.Pine 1993, in his book *Mass Customization*, claims that companies should be able to match the ever increasing customer need for variety by developing multiple products that incorporate knowledge and buyer's information [46]. Such recommender systems should be used by organizations to help them adapt to this newly formed economic environment. By processing the vast amount of information expressed by their costumers, they would be able to offer them more personalized and customized products.

The exponential growth of world wide web users and the amount of time they spend on it, led to the accumulation of an unprecedented amount of information. In this ocean of hyper-information, a user may be impersonated to a wanderer who finds it extremely difficult to navigate himself to the target port. This situation may be described by the term *information overload* [37]. The urgent need to overcome this problem led to the development of search systems that browse information, filter it and return it to the user in a form of a list that contains the most relevant pages to the search query he submitted. Quickly though it became apparent that this wasn't a panacea. A user may not be aware of the existence of the information he would be interested in and therefore look for it with the help of a search engine. This also led to a more widespread use and adoption of recommendation systems by enterprises and individual consumers.

1.2 Recommender System Approaches

Recommender Systems are software tools that offer its users personalized lists of products, services or in general content that may interest them [6, 48, 8]. In essence, they filter the available information and create those suggestions based on various characteristics that objects being recommended and the subjects that receive those suggestions, carry with them. Those suggestions help users with decision making activities. Some of the most common applications concern music or video selection, what products to consume or even what news to read. In most cases such systems generate the personalized list of recommendations, by processing browsing and consuming history or value judgments a user has expressed for similar products or services [43].

Alas, most users rarely do express their feelings for the goods they consume, in the online world. On top of that, new products that appear on the market naturally do not have previous evaluations. To tackle such problems recommender systems take advantage of information related to the consumer profile of each user or produce cooperative recommendations [6]. The concept of collaboration lies in the fact that the goods that such systems recommend to a user have been evaluated by other users who are in their social environment or show similar consumption behavior. This way of producing recommendations, uses data from other users to predict the preferences of the interested party. The selection of the sample that is used for the prediction is based on social criteria, such as social networking, common demographic characteristics, perceptions or expressed preferences in the past. Depending on the philosophy that governs the method of producing the recommendations, information filtering systems are distinguished into several subcategories [7].

1.2.1 Content-based filtering

These systems choose to present the user, items that showcase similar characteristics with other objects that has consumed in the past or has implicitly or explicitly declared that enjoys [38]. To make similarity calculation between objects possible, often requires items to be represented by a set of words that correspond to specific features. Such filtering of course, assumes the existence of information on such characteristics, as the price of a product, its description or its

type. In such way, object's Item profile is formed. A common way to quantify similarity between object characteristics is to represent them in the vector space. Each item is represented by a weighted vector, with its values corresponding to the importance of the each keyword. Their values may be formed with various numerical statistics, as the tf - idf index [59].

The final comparison may be carried out by averaging the values of the features or picking the highest value of the most desired one. A major drawback of this method is that many item characteristics are difficult to quantify before there are any evaluation judgments from the users [26]. A challenge also remains in transferring user's preference from one type of object to another; e.g a system might spot that a user enjoys reading sport news, but how this is information is translated when it needs to produce him music suggestion [45].

1.2.2 Collaborative filtering

Motivation behind collaborative filtering mainly lies in people's tendency to show mutual preferences with their peers [52]. Such systems usually initiate their operation by asking users to rate several items. They generate new user's suggestions based on the following notion. If both a new user and an old one rate highly a specific object, they probably share a common taste. Thus it is a good idea to recommend him other items the old user has rated highly in the past. This way collaborative filtering systems build matching user profiles [51].

In a similar vein, when a user A and a user B buy the same product an item to item relationship matrix is formed. Such collaborative filtering techniques make more accurate recommendation predictions between different object types. Two users that enjoy movies from the same genre, are more likely to also share a common music taste [31].

To generate predictions though, ratings from past users are required. Users are prone to rate only a few items and especially the ones that really enjoyed [17]. Hence, there might be a significant number of items that do not have any ratings. Such systems, seem to neglect those during the formation of the suggestions, causing a richer gets richer domino effect, as items that are already exposed previously to users are the ones getting proposed to new ones as well [1].

1.2.3 Knowledge-based filtering

Knowledge - based systems incorporate from their construction, the cognitive relationships that connect the all objects in the itembase. For example, a music distribution portal may have a priori employed a classification algorithm to categorize the tracks according to the musical genre they represent. In addition they may include the degree of similarity between kindred tracks [16].

Knowledge-based recommendation systems do not face sparsity or cold-start problems but are difficult and costly to construct. Especially in cases they include numerous items, it is very difficult even for experts to identify and quantify in a relatively objective way all the relationships between them. In the real world, they are mostly used in markets that users do not or rarely rate products, like the real estate or the car selling sector [55].

1.2.4 Demographic filtering

Demographic filtering techniques perform a relatively simple and straightforward categorization of users based on a series of their demographic characteristics [33]. It is considered a stereotypical approach that is not particularly accurate. It is used only in cases where there are no object evaluations by users, nor information about their social network.

1.2.5 Utility based recommendations

Utility based systems produce recommendations by computing the utility an object provides to a user [17]. Such systems mainly search for matching objects with similar characteristics, without generating user profiles.

1.2.6 Hybrid approach

Hybrid systems combine characteristics of two or more of the aforementioned categories. In reality, most recommender systems are hybrid. The main problem they face is their relatively long required computational time to produce their output [15, 14].

1.3 Evaluation of Recommender Systems

Evaluating a recommender system has proven to be an extremely difficult task. What really sets this information filtering system's assessment apart from others, is that it is not sufficient to examine how accurate it predicts user's choices. A more holistic approach is required to investigate up to what degree its employment serves the needs of all its partakers [2].

Users, may not simply expect from such systems to propose them items that match their preferences. They may seek to discover novel and diverse content or even other object characteristics that system designers are not yet aware of [11]. Mapping the latter in the movie recommendations scenario, the following questions may arise; Does the system recommend to users movies they would really like to watch? Are those placed on top of the recommendation list? Does it help users to discover new movies? Does it's performance improve over the course of time? To answer these questions, a multiple level analysis from the point of view of all system's stakeholders should be performed.

Optimizing a recommender system's performance, may be decomposed to a classification and a ranking task. Error based and ranked based methods provide metrics that try to quantify the above, respectively.

1.3.1 Error Based Methods

Error based metrics are used to asses recommendation algorithm's predictions regarding item ratings from users. By forecasting them correctly, models may infer items relevance towards a user. To compute them, knowledge of the predicted and the observed values of the ratings is required. Let $\hat{y}$ be the vector of the predicted value ratings the recommender system outputs and y the one that contains the real ratings by n users, with $\hat{y}_i$ and y_i their corresponding values.

- **Mean Absolute Error (MAE):** expresses the sum of the average difference between the value predicted by the recommender and the actual value given by the user. A MAE value of zero means a perfect prediction, there was no difference between predicted and actual rating. Therefore, the lower the MAE the better. In the world of recommender systems it is used to provide a holistic view of the system's accuracy. It does not penalize or give any specific weight values to outliers.

$$MAE = \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{n} \tag{1.1}$$

- **Mean Squared Error (MSE):** It is similar statistical metric to MAE. Though there main difference lies in that in the use of square of the difference between the predicted and observed values, causing a significant penalization of large error values. Hence its value would be larger than in MAE case.

$$MSE = \sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n} \tag{1.2}$$

- **Root Mean Squared Error (RMSE):** In the case of RMSE, we take the square root of MSE to normalize the scale issue that MSE showcases. Additionally as MSE, it allows to compare errors with different rating scales.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{1.3}$$

Aforementioned metrics provide an intuition behind the accuracy, a recommender system classifies correctly items as relevant or not towards certain users. Though, they completely ignore to account for the position those items are placed in each user's personalized recommendation list. Two of the most commonly used such metrics are precision and recall, try to make up for that up to a certain point. In the context of a recomendation system, a user is mostly interested in the top - N items that receives as suggestions. Thus, it makes sense to report $precision@k$ and $recall@k$, with k showing the maximum position in the list the user is willing to seek a recommendation. Again both system predictions and ground truth data from the users are required.

- **Precision:** Precision is the number of true positives divided by the number of true positives plus the number of false positives. In our case it expresses the percentage of the items that the model predicted as relevant and truly are, based on the expressed user preferences. If the recommender system selects 3 items to recommend out of which 2 are relevant then precision will be 66%. In its gist it shows the accuracy the system shows in identifying the truly relevant items to a user.

- **Recall:** It is computed as the number of true positives divided by the number of true positives plus the number of false negatives. It shows the percentage of the total number of relevant items the system has recommended to the user. As a metric, recall checks if useful items are not missing. This measure is proven to be exceptionally useful especially in dealing with imbalanced classification problems, where accuracy measures fail to correctly evaluate a model.

- **ROC Curve:** Receiver Characteristic Curve is a graph that plots true positive rate over the false positive rate at various threshold values. It helps to determine the best threshold regarding the maximum number of relevant items the system correctly classifies and the minimum number of non - relevant items that the model wrongly predicts as relevant to the user. ROC curve diagrammatically relates the correctly predicted items and the incorrectly predicted items. We can infer from it the points that contribute positively to the performance of the recommender system and which ones cause failures.

1.3.2 Ranking Based Methods

Previously mentioned methods do not take into account sufficient information related to the position items appear in the recommendation list. They mostly concern the classification component of the task. For example, the results of a recommender system may show a good RMSE score, despite the fact that the first three of the recommended items could be irrelevant to the user's preferences. In such a case the usefulness of the RS would be significantly reduced.

To address the ranking optimization in the suggestions, ranking based methods propose metrics that measure the quality of the recommendations in terms of their relevance to the users. Two main assumptions are made. It is better to have more relevant items posses top list positions. The more relevant an item is to a user, higher is the relevance grade it comes with.

- **Average Precision:** Is given by the summation of the relevant items precision scores divided by the number of relevant items m, found up to position n. It helps to understand the overall performance of the model but does not tell if the items were ranked properly. In essence, it measures the quality of the recommender model by emphasising on the rank relevant items appear, but does not penalize the existence of non-relevant items in the list.

$$AP_p = \frac{1}{m} \sum_{i=1}^{n} P(i).rel(i) \tag{1.4}$$

- **Cumulative Gain (CG):** It is a popular metric used to evaluate ranked lists. It shows the sum of the graded relevance values of the recommendations up to a specific position in the list. Summing all scores rel_i up to the position p, we get the cumulative gain of the recommendations.

$$CG_p = \sum_{i=1}^{p} rel_i \tag{1.5}$$

- **Discounted Cumulative Gain (DCG):** It incorporates the notion that if a highly relevant item appears lower in the recommendation list, it should be penalized. To achieve this, it divides relevance scores rel_i of each item by the logarithm of its position in the list. Various versions of the DCG appear in the literature. Some of them emphasise more in recommending relevant items.

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)} \tag{1.6}$$

- **Normalized Discounted Cumulative Gain (nDCG):** It is given by the division of the DCG at position p with the ideal one, $IDCG_p$. To compute the latter, we rearrange the elements of the list by placing the most relevant suggestions on top in order to form the perfect ranked list $IDCG_p$. This normalization, is unaffected by the choice of the logarithm base. It helps us compare the quality of rankings produced from different models, even when the length of the list and the number of items is not the same. An $nDCG$ score of 1, implies a perfect ranking score. Issues regarding this metric in real world scenarios occur when the ideal ranking of the objects is not available.

$$nDCG_p = \frac{DCG_p}{IDCG_p} \tag{1.7}$$

$$IDCG_p = \sum_{i=1}^{REL_p} \frac{rel_i}{log_2(i+1)} \tag{1.8}$$

, with REL_p being the list with the decreasing order of relevance scores rel_i.

1.3.3 Probabilistic Approach

To produce practical and meaningful recommendations, an information retrieval system requires information, regarding which objects a user finds useful. In real world applications, a user rates a number of items and based on this knowledge the model predicts the relevance of its items to that user. The notion behind the probabilistic approach of relevance lies in the fact that no system is able to make such a definitive prediction. *Probability Ranking Principle* suggests that each object has a probability to be relevant to a user. Thus, the optimal ranking that maximizes user utility derives from placing items in a decreasing order in terms of their probability of relevance [49]. To mathematically derive this, we formulate a function that provides a utility value for an item i being recommended at rank r, for each user u [54].

$$U(r \mid i) = \sum_{u \in U} \lambda(rel(i \mid u)) \sum_{i \in I} v(rank(i \mid r)), \tag{1.9}$$

where U and I are the user and item vectors, respectively. Both v and λ are application depended functions. Ranking based methods defined in previous subsections may be used in this case. $v(rank(i \mid r))$, shows the attention users pay at rank r and $\lambda(rel(i \mid u)$ models the relevance score of item i to user u. In essence they show the utility rank r carries with it and the utility item i provides to user u. Assuming relevance to be a binary variable, term $rel(i \mid u)$ shows whether a recommended item i is relevant to a user u or not. With the use of the DCG metric as stated in equation 1.6, we may replace terms from equation 1.9 accordingly; $v(rank(i \mid r)) = \sum_{i=1}^{p} \frac{1}{log_2(i+1)}$ and $\lambda(rel(i \mid u)) = rel(i \mid u)$. Combining both utilities under the same expectation, results in the following statement:

$$DCG(r \mid i) = \sum_{i \in I} \frac{rel(i \mid u)}{log(1 + rank(i \mid r))} \tag{1.10}$$

Performing the same merge of equation 1.9 in its general form we get:

$$U(r \mid i) = \sum_{i \in I} v(rank(i \mid r)) t(i \mid u), \tag{1.11}$$

with $t(i \mid u) = \sum_{u \in U} \lambda(rel(i \mid u))$, that shows the expected utility of an item i to user u. With the previously made assumptions that relevance variable takes binary values and function λ is the identical one, $t(i \mid u)$ represents the probability of relevance and thus sorting recommended items in terms of that measure leads to the optimal ranking that maximizes user utility [54].

1.3.4 Other Non-Accuracy Methods

Various evaluation methods are mentioned, regarding the relevance of the recommended items and their appropriate ranking, but none of them helps us to evaluate hypotheses such as how many items the model is suggesting out of its total item base. Or whether the model suggests to users something innovative or it only recommends items similar to its past expressed preferences. On top of that, a recommendation system's goal does not confine in maximizing individual users utility. Through there use, business that employ them also pursue their own goals. Also

minority or groups that share certain characteristics may also require special treatment from the model. For those cases, some qualitative and quantitative methods are presented below.

- **Coverage:** It measures the total number of final proposed items a recommender system includes in all its generated lists, out of its total item base. Suppose a system base contains 1000 products and the model recommends 800 of them across different users. Then the recommender's coverage is 80%. Coverage may be further categorized in terms of item types. Per say, it may report the percentage of popular items and non popular ones that the model is able to suggest. Coverage metric may also be linked with higher diversity and novelty in terms of item recommendations [35].

- **Popularity:** Users tend to express their positive feelings for the very same items others also do so. Recommendation models, capture this information, reserving them a spot in their generated item lists. We refer to those, as popular items. A well-known phenomenon in recommendations is the popularity bias, where a few popular items are over-represented in recommendations, while the majority of other items do not get significant exposure. Despite the fact that there might be a good reason suggesting those items, not every user has the same degree of interest towards popular items. Users might be interested in less popular ones. The algorithm should be able to address the needs of those users as well. To measure the extend each algorithm amplifies the popularity bias, *Popularity Lift* is proposed [4]. Firstly, the average item popularity of the user's profile, $GAP_p(G)$ and its recommendations $GAP_q(G)$, are computed.*Popularity Lift* for group G is computed accordingly:

$$PL(G) = \frac{GAP_q(G) - GAP_p(G)}{GAP_p(G)} \tag{1.12}$$

  PL values greater than zero suggest a popularity bias amplification, whereas a negative value appears in cases that the user's profile is more focused on popular items compared to the model's recommendations.

- **Novelty:** In various domains, as in the case of music search, users may be satisfied with receiving recommendations that match their already expressed preferences. But even if at first this satisfies the user, shortly after he will most likely start looking for something different. Novelty is a very important feature of a recommendation system. It is important for a user to receive interesting and unexpected recommendations. To achieve it, recommender systems may randomly suggest a few items to users every once in a while [20].

- **Diversity:** Similar to novelty, depending on the domain, measuring the model's diversity is also useful. With high diversity we consider that users will always have something different and diverse to view or consume. The most well - known diversity quantitative metric in the field of Information Retrieval is the *Maximal Marginal Relevance* or MMR. Most search engines rank documents based on a decreasing relevance principle, while *relevant novelty* is what they should strive for. *Maximal Relevance* metric provides a linear combination of novelty and relevance. A document possesses high maximal relevance, when it is relevant to the query and simultaneously has a minimum similarity compared to previously retrieved documents [19].

- **Temporal Evaluation:** People alter tastes over the course of time. A user immediately after watching a movie might rate it with a 10 out of 10. After two or three years though, this rating might drop to a 8 or even a 6. There are many factors that may cause a shift

their preferences. This may occur as trends constantly evolve. Or even because growing up makes a person more mature and changes the way that perceives things. Netflix uses temporal elements in its model. Considering every rating a user has ever provided, compared to only taking into account the recent ones, may have a substantial impact on model's predictions under a specific time frame [36].

- **Business Metrics:** Apart from measuring the predictive power of the recommendation system, it is important to measure how the model is performing in terms of business objectives [50]. Does the integration of the recommender system increases organization's revenues or its web traffic?

1.4 Fairness Framework

1.4.1 Bias and Unfairness in the Web

Biases in society are evident. They make their presence notable in various aspects of life [10]. In the modern era and especially after the recent technological achievements and the dawn of the world wide web, forms of bias have transformed. Web may have mitigated some of the former ones, mainly as far as access to knowledge and communication are concerned, though it has brought to the surface new more complex and difficult to identify matters [44].

The most severe web bias is that there are over 3.5 billion people still not connected to the internet, either due to lack of infrastructure or technological expertise [27]. Language biases also appear more frequently after the emergence of the internet. Around 50% of most well known web pages have their content solely in English, though only 5% of the total population does have it as a mother language [10]. Moreover, gender bias rollovered in the web. Only 12% of Wikipedia's knowledge is actually written or edited by female users [10]. On top of that, the web itself throughout the years has seen enormous changes. Knowledge on the web at its early stages used to be accessed almost solely by a query given in a search engine machine.

Nowadays it has become a much more complex structure with many social, economical and political extensions. Only during the last decade average time spent online has quintupled [10]. Without a doubt the internet fulfills a vital role in our modern society. Most time spent on the web nowadays is on social media platforms or e-commerce ones. In 2021, the average American social media user spends approximately two hours and 25 minutes per day across all platforms. Moreover, 52% of them claim that they read the news most of the time through social media [27]. Finally, around 26% of e-commerce platform's revenues, comes from items that were personally recommended to consumers [27].

1.4.2 Bias and Unfairness in Recommendations

Ranked search results, news feed and recommendations have become the main medium through which we discover content, products, places and people online. These rankings are typically constructed to provide a maximum utility to searchers [49]. However, there are reasons to depart from a simple ranking by relevance or utility. When the items to be ranked represent people or when they represent businesses and places, ranking algorithms have consequences that go beyond simple utility measures. With hiring, selecting, purchasing, and dating, rankings may determine career and business opportunities, educational placement or even social success.

Typical recommendation systems produce their output with the use of *Probability Ranking Principle* introduced by Robertson S.E in 1977, as discussed previously [49]. This way the system produces suggestions that maximize user's utility and are considered fair from an individual's subject point of view. But this is only the one side of a coin, in a pouch with several

ones. Recommendation Systems should not solely focus on addressing the needs of a single partaker. In a profound way they also affect the objects themselves that are being ranked. In the job-seeker example, objects are workers. Low ranked ones by the system will not be able to land a job. Non - suggested items in an e-commerce platform, will not be purchased. Their suppliers may not withstand the competition and be forced to shut their production down, limiting the range of goods that are available to consumers. In the social media scenario, objects represent opinions and notions. Promoting only a subset of them may confine polyphony in public speech.

Thus, recommendations may unfairly treat certain individuals or groups of entities. Term $unfairness$ by its nature, is highly vague. To formally define it, in each case, we need to specify the involved partakers and the point of view fairness is perceived. In most recommender system applications, a user receives a list of suggested items. User and Item terms though, depending on the scenario might refer to the same entities. For instance, an organization, seeks to hire the most suitable candidate from a number of applicants. There is also a chance, for an employee to search for the most relevant organization to apply for a job. In the first case, the user is the employer and the item corresponds to the job-seeker person, though in the second example the roles are vice versa. Thus, it makes more sense to talk about objects being recommended to subjects. A system may provide a notion of fairness towards individual objects and subjects or amongst specific groups of them.

1.4.3 Fairness in Recommendations

As it is priorly mentioned, in essence recommendation system optimization may be seen as a combination of a classification and a ranking problem. Fairness notions and concepts from the two aforementioned research fields may be generalized and applied in this case.

- **Fairness in Classification:** Abstractly, unfairness is caused by treating individuals that belong to certain protected groups[1], not in the same way. Fairness, in Machine Learning applications that perform Classification tasks, is primarily examined from the scope of the subject. As far as Group Fairness is concerned, *Fairness through unawareness* tries to ensure equality in treatment [2] It suggests, that the machine learning algorithm should not take into account any information about the objects that are classified that may lead to discriminatory predictions[28]. *Equal Opportunity*, provides a notion of parity in impact[3]. It represents the equal opportunity principle for supervised learning tasks. To satisfy this principle, false positive and true positive rate should be equal across all different protected groups [34]. *Demographic parity*, mandates both true and false positive rate of the predictor alongside all protected groups to be the same. Both concepts perceive fairness on a group level. Individual fairness may be reached, when individual objects that share common characteristics, receive homogenous results [57].

[1]**Protected Groups:** Groups of Objects/Subjects that share one or more sensitive attributes, such as ethnicity, gender, marital status, etc.

[2]**Parity in Treatment:** Refers to the notion that a fairness constraint should target the input of the algorithm (e.g a recommender system should be trained with the same number of movies from each genre).

[3]**Parity in Impact:** The idea is to provide a notion of fairness by enforcing a constraint directly in the final output of the system (e.g a recommender system suggests to the user the same number of films per genre).

- **Fairness in Rankings:** Typically, Fairness in Rankings revolves around object assignment in an ordered list, at their appropriate position. Concepts proposed to provide a sense of fairness in classification tasks, may be used in this case as well. *Statistical Parity* in rankings ascertains that a minimum number of protected objects makes the final recommendation list [56]. Similarly, recent works suggest that in the top - k positions, a predefined ratio of protected and non - protected items should exist [21]. *Demographic Parity* constraints enforces not only the final recommendation list to include objects from the protected group, but to also place them in positions that cause the average exposure between groups to be equal [18]. To simultaneously address group object fairness while maximizing individual user utility, a Fair Top - k Ranking Algorithm generates a list of items, in decreasing order of utility, up to the point the latter fairness constraint is not violated [58]. *Equity of Attention* dictates that an object of a specific protected group should get exposure relative to its average utility for a specific subject [12]. *Fairness of Exposure in Rankings* proposes a general fairness framework that efficiently computes utility optimization in rankings subject to certain constraints [54]. Particularly, Disparate Impact fairness constraint enforces the expected clickthrough rate of popular and non popular items to be proportional to the their average utility. In the movie recommendations scenario, the subjects are users and objects are movies. Let's consider the non-popular items (e.g movies from independent producers) to form the protected G_u and the popular ones (e.g Hollywood films) the non-protected group G_p. The latter fairness constraint for this scenario is formulated accordingly:

$$\frac{CTR(G_u)}{U(G_u \mid i)} = \frac{CTR(G_p)}{U(G_p \mid i)}, \tag{1.13}$$

  where $CTR(G_u)$ is computed as the product of exposure and relevance score in expectation, while $U(G_p \mid i)$ refers to the average utility of the popular and non popular items.

1.4.4 Forms of Bias in Recommender Systems

Unfairness in recommendations is primarily caused by the effects of a series of biases. Most recommender systems used in real world applications do suffer from such intrinsic forms of bias [22], presented below:

- **Position Bias:** Items recommended in the form of a ranked list, suffer from a positional bias. Top suggested items are by far more likely to be spotted and therefore clicked. For example the first item in a catalog is 10 times more likely to be clicked in comparison to the tenth one, no matter their corresponding relevance to the user [25].

- **Popularity Bias:** Popular items are rated more frequently by users. Such information is used as input data in recommender systems to produce suggestions that as result are also heavily dominated by popular items [5].

- **Exposure Bias:** As a consequence of popularity bias, non popular items receive less attention than the popular ones, leading to an exposure bias. Under a multi - stakeholder environment, such phenomenon may affect the interests of consumers that do not enjoy popular items and suppliers whose products do not receive significant exposure [3].

- **Model Bias Propagation:** Alas, popular items are more likely to be observed and therefore clicked by the user. This information returns as input in the system for future recommendations, signifying a model bias propagation [41].

1.5 Contribution

Scope of this research thesis is to investigate the effectiveness of bias mitigation techniques, initially designed for static recommendation approaches, under a dynamic setting. To achieve this I simulate the operation of two recommender system pipelines that present personalized ranked lists of objects to users. The first pipeline is used as a baseline. It produces its recommendations by employing *Bayesian Personal Ranking*, a well known learning to rank algorithm [47]. The second one, follows the same procedure, except this time object recommendations are filtered through a post-process bias mitigation algorithm [54].

A very important factor of this work, is the element of time. Under a non - dynamic temporal setting, the proposed reranking fairness algorithm succeeds in allocating exposure between groups of protected and non - protected objects more fairly, while maximizing user utility in expectation. Should also this be the case in the long run, past a few hundred rounds of recommendations? Does it allocate exposure between demographic groups more equally? Does it alleviate the effects of popularity, exposure and model bias? Does it enhance diversity and novelty in recommendations? Is it possible for the Disparate Impact fairness constraint that imposes, to ensure a sense of demographic parity? If so, up to which point and to what extent? Or stricter limitations should be adopted to alleviate the overexposure of non - protected objects to subjects.

To answer such questions, for both experiments a post process analysis on a clicked and recommended object level is performed, alongside the use of descriptive statistics measures regarding the generated data. Lastly, I compare their results based on a series of metrics presented in the *Experimental Results* section, alongside some empirical ones discussed in the *Recommender System Evaluation* section.

Chapter 2

Methodology

In this chapter an overview of the two proposed recommendation pipelines is presented, together with the intuition behind their design. Both models, are evaluated with the use of synthetically generated data. *Experimental Setup* section, contains information and general assumptions that govern the employment of both recommendation simulations. Their specifics and their mathematical background is presented, together with an analysis of their components.

2.1 Introduction & Motivation

Real world applications operate on an online ever evolving setting, where new objects are constantly added in the item base and multiple rounds of recommendations take place. Most studies, asses a recommender system under a static temporal setting, meaning that they perform their analysis by only considering the results of a single recommendation round. While this simplified approach of the task may be adopted for a number of valid reasons, as of real data sparsity or time efficiency, it may severely affect the quality of recommendation's evaluation. Under such a stable environment, the impact of various forms of biases as discussed in $Fairness\ Framework$ Section 1.4, may be underestimated or totally ignored, as it intrinsically, unfolds to its full extent over the course of time. Particularly, the amplification of the model, exposure and popularity bias, may be propagated over time by the way users interact with the system, leading to a bias piling [60]. This may be also affect the notion of fairness a model exhibits towards its partakers. Hence, dynamic recommendation process is simulated in an offline setting, with 300 rounds of recommendations taking place. Term $offline$, mainly refers to the fact that access to a real-world platform with live stream of data, is not possible, so this process is simulated in an offline mode to approximately mimic factual scenario's patterns. Next, the specifics of the two proposed pipelines will be discussed. From that point and onwards, *Base Experiment*, will refer to the base recommender pipeline that employs *Bayesian Personalized Ranking* and $FOEIR\ Experiment$ to the one that enforces the post-processing bias mitigation techniques [54].

2.2 Experimental Setup

Generated data may represent any type of object being recommended to any subject. For convenience and explainability reasons, I refer to objects as items and to subjects as users. A movie recommendation scenario may be assumed. An overview of both recommendation pipelines operation and information flow is presented below.

2.2.1 Base Experiment

Base recommender pipeline, produces its final recommendations without using any algorithm that claims to offer a notion of fairness to the items it ranks. Thus, it aims to maximize user's utility. These results, are used as a baseline against $FOEIR$ experiment suggestions. Figure 2.1 depicts the stream of information between pipeline's constituent parts.

- **Bootstrap Phase & Data Generation:** No prior knowledge of items relevance to users is taken for granted. To kickstart the system and overcome this lack of user's preferences, I randomly recommend items to users and let them interact with those. Let $S = \{1, .., s\}$ be the user vector and $O = \{1, .., n\}$ the item one, with $s = 6040$ and $n = 3800$. Every single one of the 6040 users receives a $k = 10$ items recommendation list. Item to user relevance is presumed through an implicit form of feedback, when a user clicks an item. Any proposed click model from the literature of IR may be used. For convenience, across the whole tenure of both experiments, I assume that users click the first ranked item from their recommendation list. This kind of information is used as an input to the base recommendation algorithm. The aforementioned procedure occurs only during the very first round of the simulation.

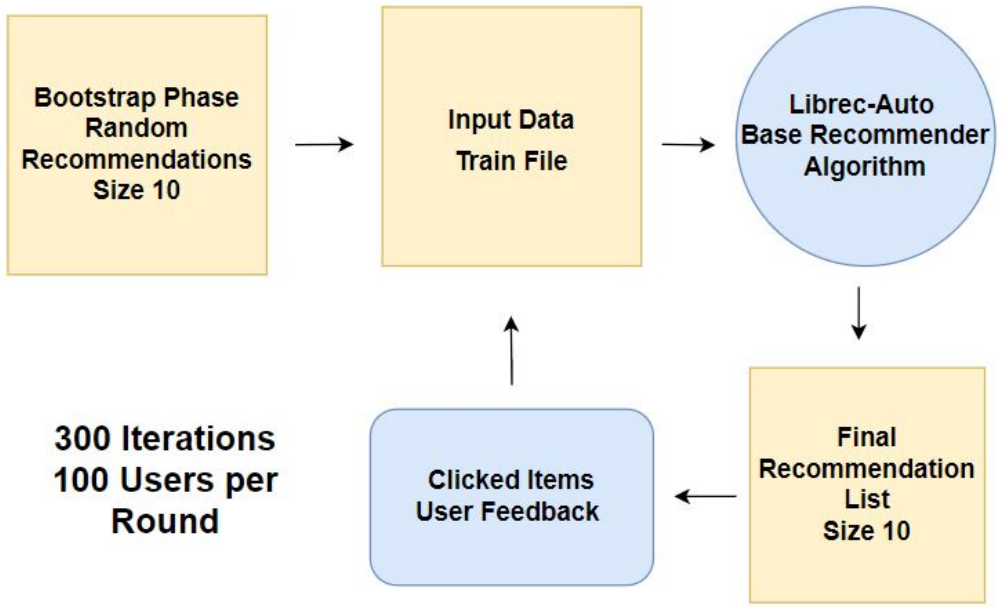


Figure 2.1: Base Experiment's Flowchart

- **Training Phase:** Data fed to the recommender alogrithm, appear in the form of $\{user, item, rel\}$. Relevance is assumed to be a binary variable, $rel = \{0, 1\}$. In this case, all clicked items are relevant to a user, so the final form of the train set during bootstrap phase consists of 6040 rows of $\{u, i, 1\}$, that indicate, item i being relevant to user u.

- **Generating Final Recommendations:** *Librec Auto* [40, 29] opensource recommendation library is used to generate the personalized 10 items list with suggestions for each user that participates in every recommendation round. Out of the 6040 total, 100 random unique users are picked to receive recommendations per iteration. The recommendation algorithm produces the personalized item proposals to 100 random unique users per iteration. To achieve that it uses a probabilistic approach to compute a relevance score per user - item tuple. The final output the algorithm appears in the form of

$\{user, item, relevance$ score$\}$. Thus, a user receives as recommendations the top ten items in terms of relevance score, sorted in a descending order.

- **User Feedback:** Once again users are asked to provide their feedback. As priorly discussed, the first ranked items that appear in the 100 recommendation lists that users receive are clicked. This new information about user - item relevance is incorporated to the existing train file. The aforementioned procedure from the training phase step, is repeated for 300 times.

2.2.2 FOEIR Experiment

FOEIR Experiment pipeline, up to a certain degree follows the structure presented in the previous subsection with some key differences and some important additions. Primarily, Base Experiment's recommendations, alongside some further information discussed below, is used as input in the FOEIR post-process reranking algorithm. That way, final item to user recommendations are produced. FOEIR framework, tries to maximize user's utility, subject to a certain constraint that aims to alleviate the effects of various types of bias and provide a notion of fairness towards non - protected objects. Regarding the implementation, authors of the Fairness of Exposure in Rankings, work [54], did not provide a repository with their implementation codebase. Versions of FOEIR algorithm may be found in online repositories[1], but still further refinement was required to address this work needs. Below, an analysis of the pipeline is presented, as shown in figure 2.2:

- **Bootstrap Phase & Data Generation:** System's initialization and the data generation that allows it, happen in the same manner as discussed in the case of Base Experiment. Users are randomly recommended items and through clicks, they express their preferences towards them.

- **Training Phase:** Train set again contains information provided in the form of $\{user, item, rel\}$. This information is passed as input to Base recommendation algorithm to produce the initial recommendations.

- **Generating Initial Recommendations:** Term *Initial* to characterize the recommendations is adopted as these are not the final rankings a user receives. This time, base recommendation algorithm through *Librec Auto* produces a personalized list of size $k = 40$ items for the 100 randomly selected users. Further motivation behind this choice is provided in the FOEIR Mathematical Background subsection, but $FOEIR$ algorithm requires to receive as input a larger list of recommendations in order to perform its reranking and produce the final shorter one of size 10. On top of that, items should be classified into protected and non protected groups. We assume the existence of only two groups. Let G_p contain the popular items and G_n the non popular ones. Items are characterized as popular or non popular based on their *popularity score*. Popularity score of an item i is computed as:

$$ps_i = \frac{n_i}{N}, \tag{2.1}$$

with n_i being the number of times item i appears across all recommendation lists for a specific round and N the total number of recommended items. In essence, it shows the probability item i is being recommended. Now let $PS = \{ps_1, .., ps_N\}$ be the sorted in a decreasing order vector, with all ps_i values. The sum of the first $j \subseteq N$ items with

[1] `https://github.com/MilkaLichtblau/BA_Laura` , `https://github.com/jfinkels/birkhoff`

popularity scores greater than h, are labeled as popular and the rest as non popular. Mathematically, $\sum_{i=1}^{j} ps_i >= h$. A value of $h = 0.4$ is considered, representing the 40% of the total popularity score. Moreover, to compute the attention a user pays in each position k in the recommendation list, the following formula is used:

$$u(k) = \frac{11 - k}{10 + 1} \tag{2.2}$$

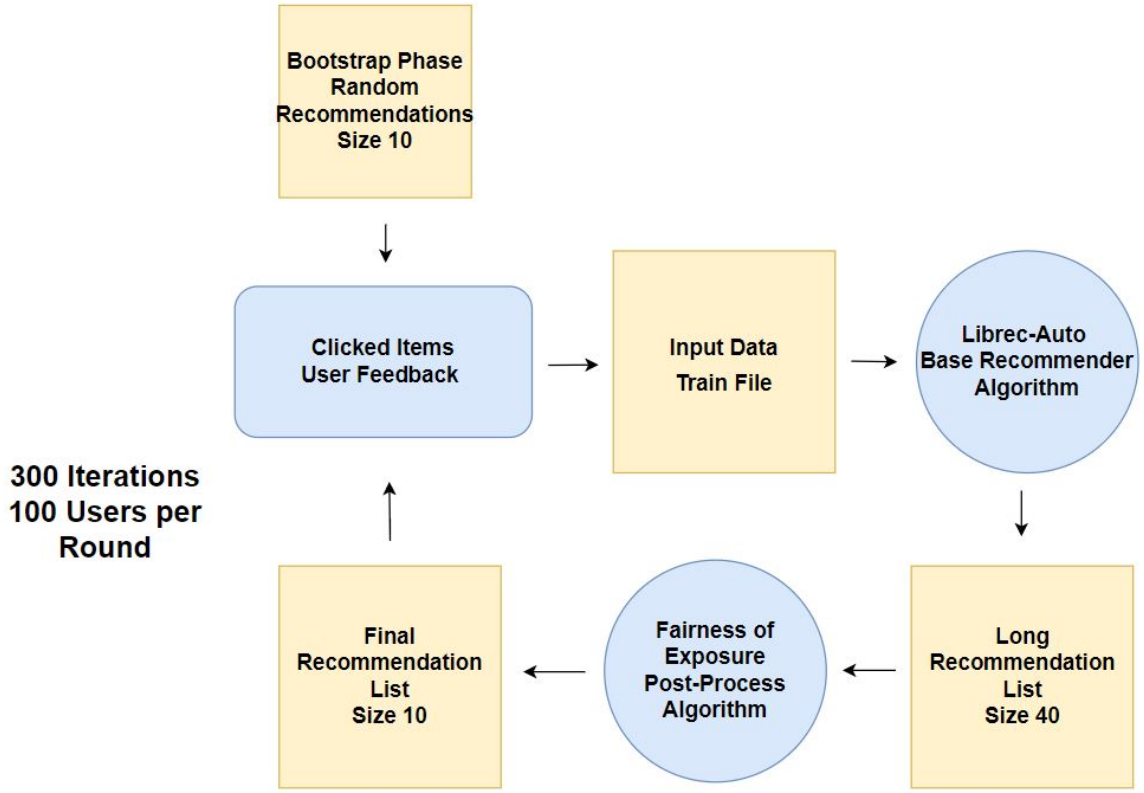


Figure 2.2: FOEIR Experiment's Flowchart

Popularity score as also the aforementioned way items are assigned labels, is also adopted in the Base experiment, but they are post - process appended, as they are not necessary to be computed while the pipeline is in motion.

- **Post-Process Reranking:** FOEIR algorithm receives as input the long recommendation lists of 40 items for the 100 randomly selected users, alongside their label $l = \{0, 1\}$, with value of 0 corresponding a popular and 1 to a non popular item. Gathering all the elements, it receives information in the form of $\{user, item, label\}$ for every recommendation produced by base recommender.

- **User Feedback:** As depicted in figure 2.2, users interact with the final FOEIR recommendation list of size 10 the same way as before. Clicked items are appended in the train set, as a user item tuple. This procedure is executed for 300 times.

2.3 Mathematical Background

2.3.1 Base Experiment

Mathematical intuition behind Base Experiment's recommendation algorithm comes from *Bayesian Personalized Ranking* or BPR [47]. Although there are alternative methods for item recommendation from implicit user feedback, like *Matrix Factorization* [39], BPR introduces $BPR - OPT$, an optimization criterion that is directly optimized for rankings. An interesting observation is that only positive user feedback through user clicks or items purchases is available, but no information at all regarding items the user might want in the future as recommendations. Most approaches deal with this issue by replacing those missing relevance values with negative ones and then produce a utility score for every item individually. BPR uses item pairs and directly optimizes the ranking of those. Base pipeline, produces 1000 final item recommendations to 100 unique users, in each round. Additionally, 100 recommendations are clicked. Different users might click the same item more than once. Those user - item tuples are appended in the train set that is used as an input to the recommendation algorithm, to produce suggestions for the next recommendation round. Newly added $\{u, i, 1\}$ entries might be unique, meaning that there is no prior information regarding an item i being relevant to user u, but it may also be duplicate. In the first case, system learns something new regarding user's preferences. In the second one, the degree of relevance increases, causing user's relevance score for the specific item to increase.

2.3.2 FOEIR Experiment

FOEIR algorithm aims to maximize users utility, subject to certain fairness constraint. To formally define it, let $U(r \mid i)$ be the utility item i has in rank r. Then the latter is formulated as:

$$r = argmax_r U(r \mid i), \text{s.t.r is fair} \tag{2.3}$$

FOEIR proposed framework takes a probabilistic approach on relevance and rankings. Regarding relevance, motivation behind that is discussed in subsection 1.3.4. As far as rankings are concerned, due to their combinatorial nature, trying to compute the optimal ranking of items that maximizes user's utility subject to a certain fairness constraint would require exponential time. Hence, a probabilistic distribution R over the real rankings r is presumed. Incorporating this notion in the way utility of an item i for a user u is computed, makes equation 1.11 take the following form:

$$U(R \mid i) = \sum_r R(r) \sum_{i \in I} v(rank(i \mid r)) t(i \mid u) \tag{2.4}$$

Given the above, utility of an item to a user over a rank $R(r)$, may be computed from the marginal rank of the distribution of ranks R. Probability distribution R places item i at rank j with a probability of P_i, j. These marginal probabilities form a doubly stochastic matrix $\mathbf{P}$ with size NxN, with N being the number of positions items may take. The sum of each row and column of this matrix should be equal to 1. The expected utility of item i given by distribution R in rank r, $U(R \mid i)$ may be written in terms of the doubly stochastic matrix P and its marginal probabilities P_i, j. Reformulating equation 3.4 we get:

$$U(P \mid item) = \sum_{item_i \in I} \sum_{j=1}^{N} P_{i,j} t(item_i \mid u) v(j), \tag{2.5}$$

where v_j is the attention items receive at rank j and $t(item_i \mid u)$ the relevance score provided by Librec Auto. Rewriting the above in matrix notation, we get:

$$U(P \mid i) = \mathbf{t}^T \mathbf{P} \mathbf{v}, \tag{2.6}$$

where $\mathbf{t}$ and $\mathbf{v}$ are column vectors of size N. Given the above, the initially formulated optimization FOEIR framework presented in equation 2.3, becomes:

$$\mathbf{P} = argmax_{\mathbf{p}} \mathbf{t}^T \mathbf{P} \mathbf{v}, \text{s.t.P is fair}, \tag{2.7}$$

including all the constraints that the doubly stochastic matrix $\mathbf{P}$ comes with. The aforementioned equation expresses the maximization of the expected utility under a probabilistic rank. According to the authors of the FOEIR framework, fairness constraint formulation happens also in a similar way [54].

$$\mathbf{f}^T \mathbf{g} \mathbf{v} = c, \tag{2.8}$$

with $\mathbf{f}$ being a vector that incorporates the binary label l and $\mathbf{g}$ a gradually descending function that accounts for the position bias, like the one presented in equation 2.2. Variable c is a constant one, that receives its value based on the nature of the fairness constraint. The solution of this linear problem is given by computing a matrix R that contains the optimal probabilities an item has to appear at a specific rank. Deriving R from the doubly stochastic matrix $\mathbf{P}$ is done with the use of Birkhoff-von-Neumann decomposition (BvN) [13]. Based on the latter, matrix $\mathbf{P}$ may be decomposed accordingly:

$$\mathbf{P} = a_1 \mathbf{P_1} + ... + a_n \mathbf{P_n}, \tag{2.9}$$

where $\mathbf{P}_i$ are the permutation matrices that contain all the possible combinations of items being ranked in every possible position. By extracting the probabilities from the doubly stochastic matrix $\mathbf{P}$, for every possible item - rank scenario, a_i coefficients are computed as the product of the marginal probabilities $P_{i,j}$ and correspond to the probability of picking each of the n possible rankings. Thus, the solution of the linear program is given by picking the ranking the permutation matrix P_i suggests with the higher corresponding coefficient a_i. This ranking R, maximizes the expected utility and respects the fairness constraint also in expectation. Finally, the choice of the fairness constraint itself needs to made. All three constraints, authors of the original paper propose are related to the more fair allocation of exposure to objects that belong in the protected group. For the implementation, Disparate Impact Constraint was selected. It forces the expected clickthrough rate of popular and non popular items to be proportional to the their average utility. Item's i probability to be clicked while being relevant is P(i) = P(examining) x P(i is relevant) or $P(i) = Exposure(item_i \mid \mathbf{P})$ x u_i. From equation 2.6 occurs that the average clickthrough rate for non popular items is formulated accordingly:

$$CTR(G_u \mid \mathbf{P}) = \frac{1}{|G_u|} \sum_{i \in G_u} \sum_{j=1}^{N} P_{i,j} u_i v_j, \tag{2.10}$$

where $|G_u|$ is the number of unpopular items, u_i the relevance score vector and v_j the vector that contains values from the function of attention for rank j. So the final form of the disparate impact constraint becomes:

$$\frac{CTR(G_u \mid \mathbf{P})}{U(G_u \mid i)} = \frac{CTR(G_p \mid \mathbf{P})}{U(G_p \mid i)} \tag{2.11}$$

By substituting $CTR(G_u \mid \mathbf{P})$, with equation 2.10, alongside the use of equation 2.8 and setting $c = 0$, we may infer a value for vector $\mathbf{f}$ that will finally allow us to solve the linear program as stated in equation 2.7. As a final remark, there are cases where respecting fairness constraint presented in equation 2.11, while maximizing user utility in expectation(2.7) is not numerically

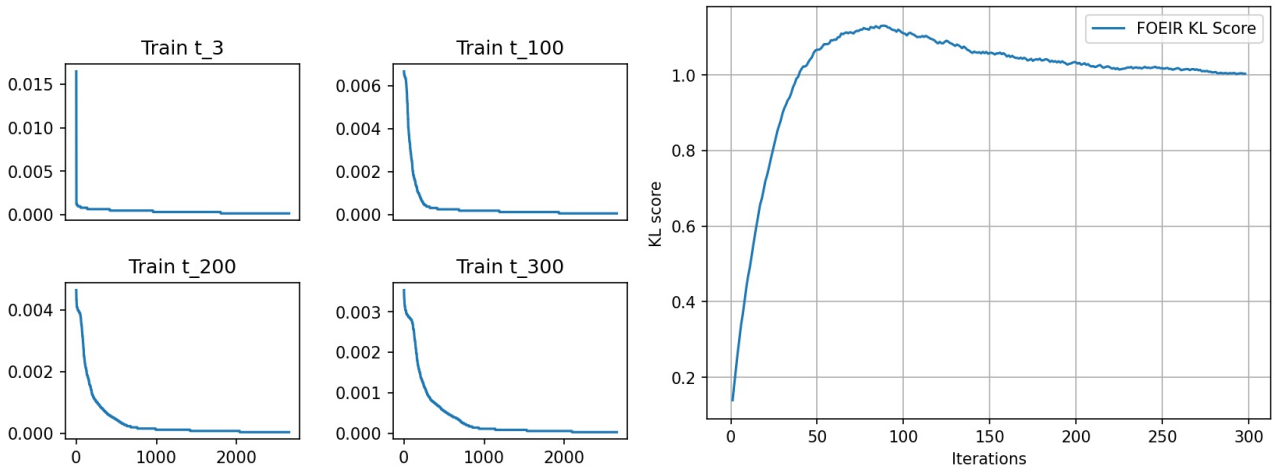
feasible. Under such circumstances, an increase in the number of popular items is performed, as suggested by the authors that originally proposed the framework [54]. During this pipeline's implementation, this issue was tackled with the aforementioned method.

Chapter 3

Experimental Results

To increase the validity of the simulated experiments, Base and FOEIR recommendation pipelines were employed five times each, for 300 iteration rounds. Minor discrepancies were observed among those. Thus, for convenience and comparability reasons, results presented in this chapter come from a single run that addresses best research questions raised in the *Contribution* section. First part of the experimental analysis, up until figure 3.10, is dedicated to the degree of popularity bias both pipelines exhibit and the way this is propagated through model bias in the long run. Secondly, the effects such phenomena have on the exposure allocation between items that belong to popular and non popular groups are discussed. Finally, by perceiving experimental results under a more holistic view, I examine the development of the initially randomly recommended items with the highest exposure, over the course of time.

3.1 Descriptive Statistics



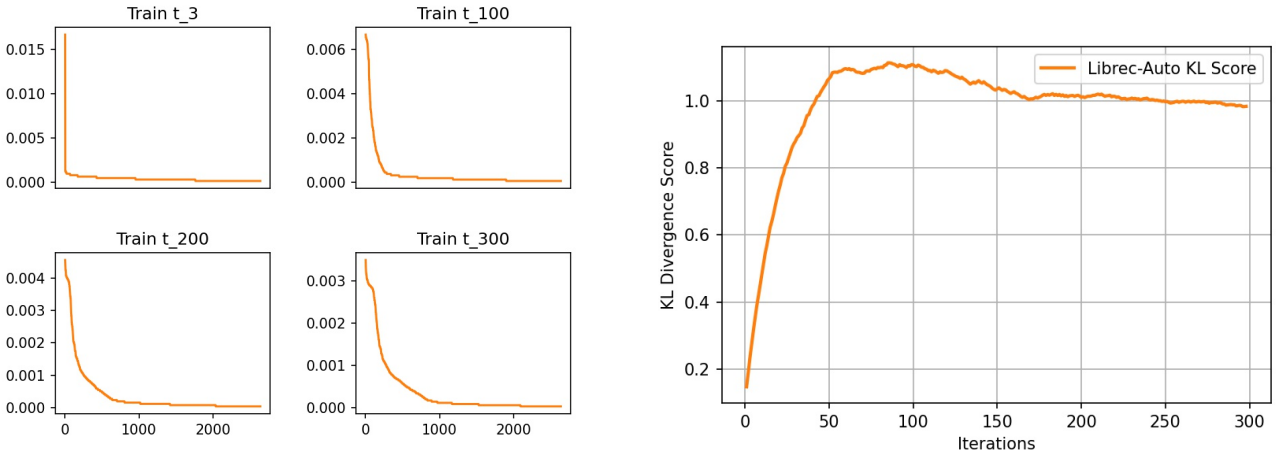
(a) Popularity Score Distribution

(b) KL Divergence score per Iteration

Figure 3.1: Divergence between Popularity and Uniform Distribution FOEIR Experiment

Figure 3.1(a) depicts the distribution of the popularity score 2.1 over the 300 iterations. Vertical axis contains the probability each item has to appear in the train file of the FOEIR Experiment pipeline. The horizontal one contains all items, popularity score sorted in a descending order, that appear in the train data. During the initial recommendation rounds, the distribution appears to be much sharper. This may be attributed to the intrinsic nature of our experiments. During each iteration, the clicked items are appended to the train file of the previous round. Thus the number of item appearances throughout the final iterations is much higher.

The first subplot shows that in the third iteration, the most frequent item has a 1.5% probability of appearance. There are only a few such popular items that showcase high probability. In the last iteration, this probability is five times lower. The curve of the distribution is much smoother. Train file contains many more items as for the expressed user's implicit feedback, that make the highest value of the popularity score to significantly drop.



(a) Popularity Score Distribution

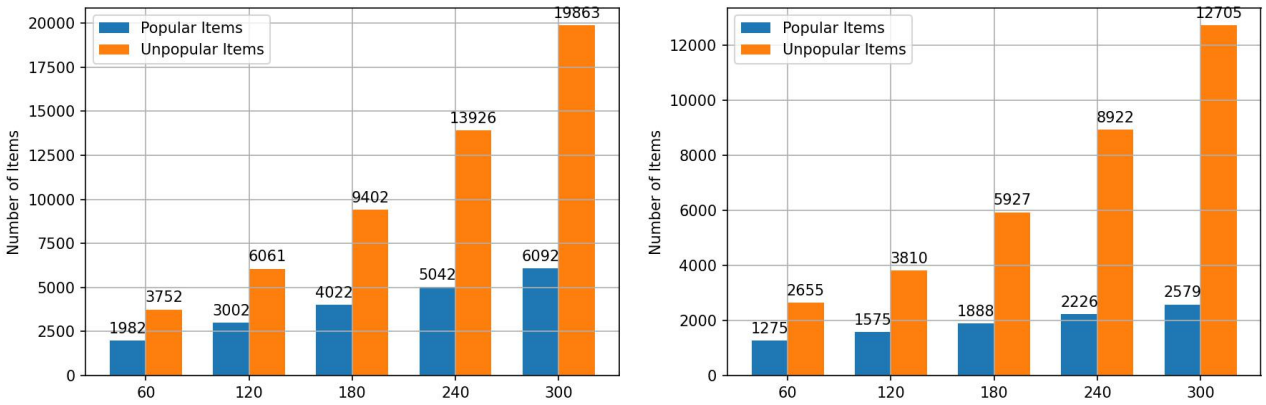
(b) KL Divergence score per Iteration

Figure 3.2: Divergence between Popularity and Uniform Distribution Base Experiment

Comparing the train set's popularity distribution for every iteration, with the Uniform one, where every item would have the same probability to appear, figure 3.1(b) is produced. The greater the score is, the bigger the deviation between those distributions actually is. *KL Divergence* score value reaches its peak just before the $100th$ iteration. Even though the distribution seems to become smoother as the rounds of recommendation progress, the divergence from the uniform distribution increases sharply, especially during the first 50 iterations.

As more items are appended in the train data set, the probabilities of appearance of other items increases, causing the ascendance of the KL Divergence score and the maximum popularity score of the few most popular items that so far are clicked, to decrease accordingly. Between the $100th$ and the $300th$ iteration KL score ranges from 1,1 to 1, showcasing minimal fluctuations.

A possible explanation might be that up to the first third of total iterations a sufficient number of items has been added to the train set and the probabilities have stabilized. Maximum probability also is approximately three times lower than its initial value, while only dropping to two times lower throughout the rest of the experiment. Specifically, figure 3.2 shows that after the $200th$ iteration, popularity distributions seem very much alike, with only minor discrepancies from each other as it is depicted in the subplots Train 200 and Train 300.

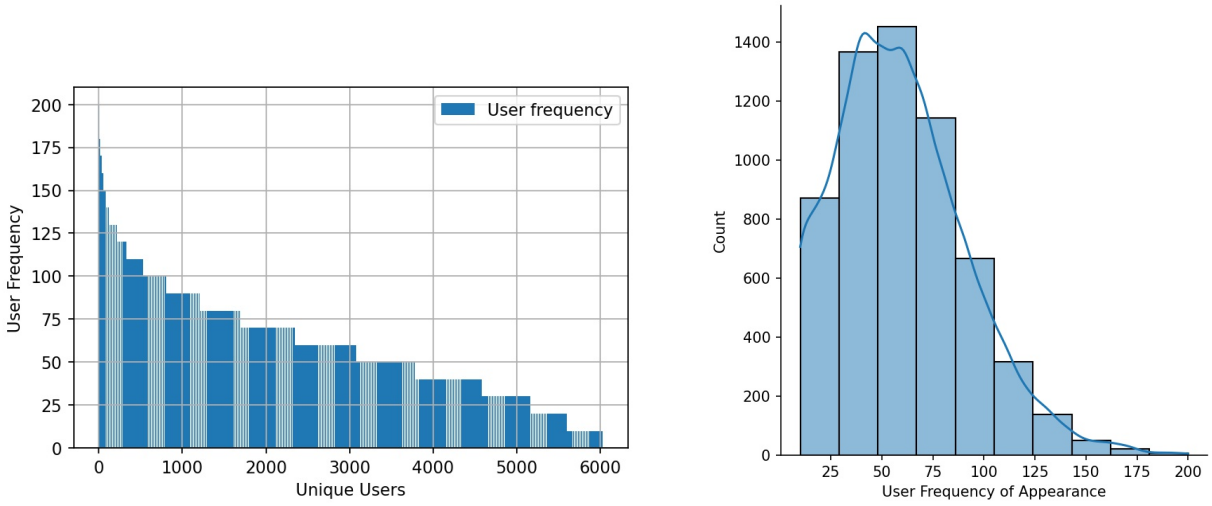


(a) FOEIR Experiment (b) Base Experiment

Figure 3.3: Total Cumulative Number of Popular & Unpopular Recommendations

Figure 3.3 shows the total cumulative number of popular and non popular items over five specific time milestones. In both cases, the number of non popular items greatly surpasses the popular ones. As the number of iterations increases, the ratio between those two groups sharply decreases. Subfigure (a), exhibits that up to the 120th iteration this fraction is around 50%. Across the next three landmarks, this value ranges around 42%, 36% and 30%, respectively. Regarding the Base experiment, on the $120th$ iteration mark the ratio is approximately 40%. For the $180th$ iteration the number of non popular items is 3 times greater than the popular ones, at 240 four times greater and in the final recommendation round, five times greater. This difference in the ratio between the two experiments, mainly lies in the way an item is defined as popular as discussed in the Methodology chapter.

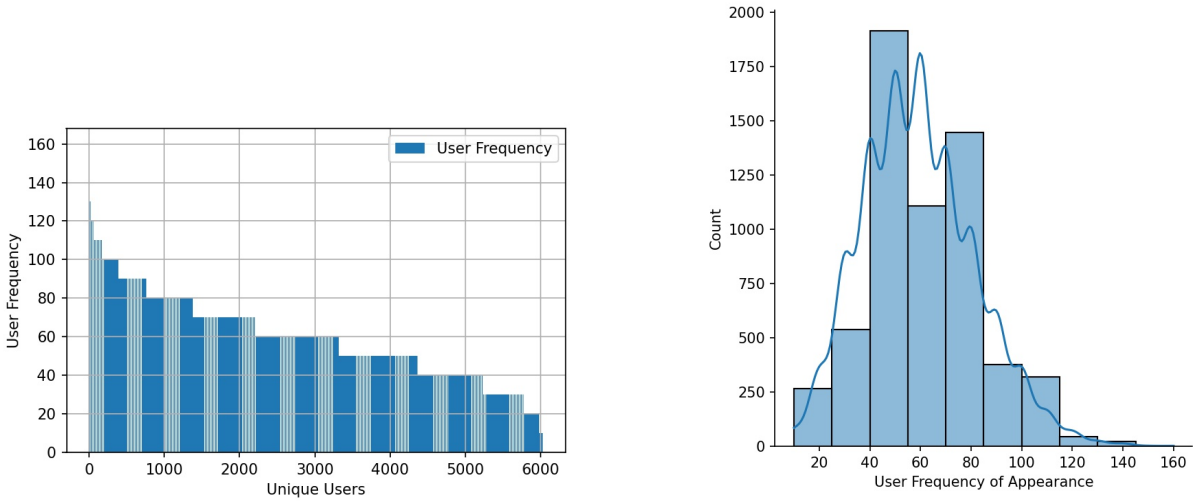
Throughout the whole process the number of the first most popular k items that account for the 40% of the total items popularity score, remains unchanged or even drops. In essence, the lower the ratio between the popular and non popular items is, fewer unique items are recommended and therefore clicked, more times. Driven by the latter statement, FOIER post-process algorithm recommends more unique items to its users, showcasing a higher notion of diversity and novelty [32].



(a) User Frequency of Occurrence (b) User ID Frequency of Occurrence Distribution

Figure 3.4: FOEIR Experiment User ID Frequency over 300 Iterations

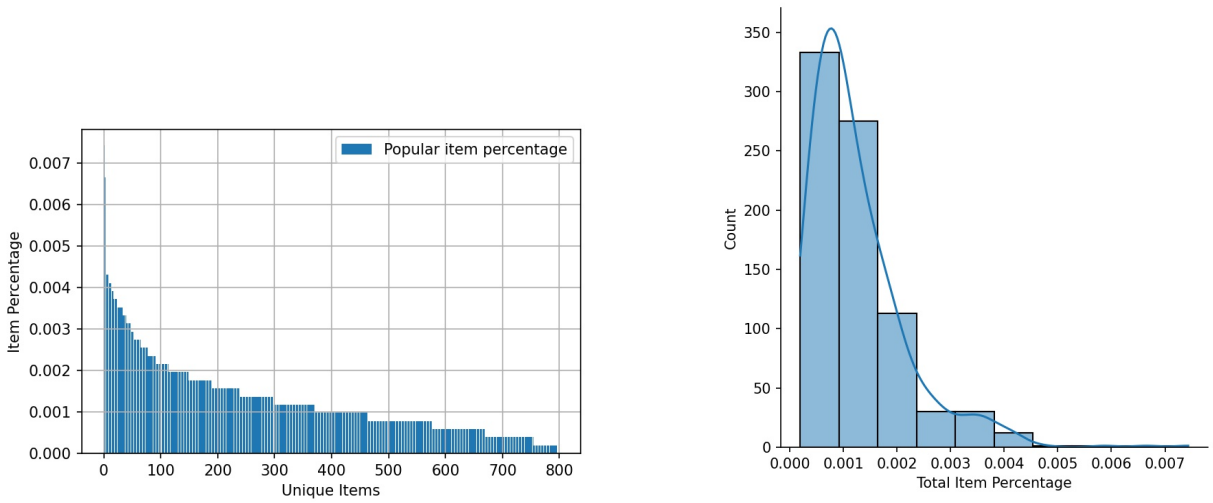
During each iteration 100 unique users are randomly picked out of a pool of 6040 users, to participate in the specific recommendation procedure. Figure 3.4 (a) presents the unique *user ids* sorted on their frequency of appearance throughout the tenure of FOEIR experiment. The most prevalent user appears in 173 iterations out of the 300, accounting for 0.5% of the total user appearance. The least, only 13 times. More than half of the total unique users show up between 40 and 85 times. The distribution may be characterized as bell-shaped, almost center based one, without a left tail.



(a) User Frequency of Occurrence (b) User ID Frequency of Occurrence Distribution

Figure 3.5: Base Experiment User ID Frequency over 300 Iterations

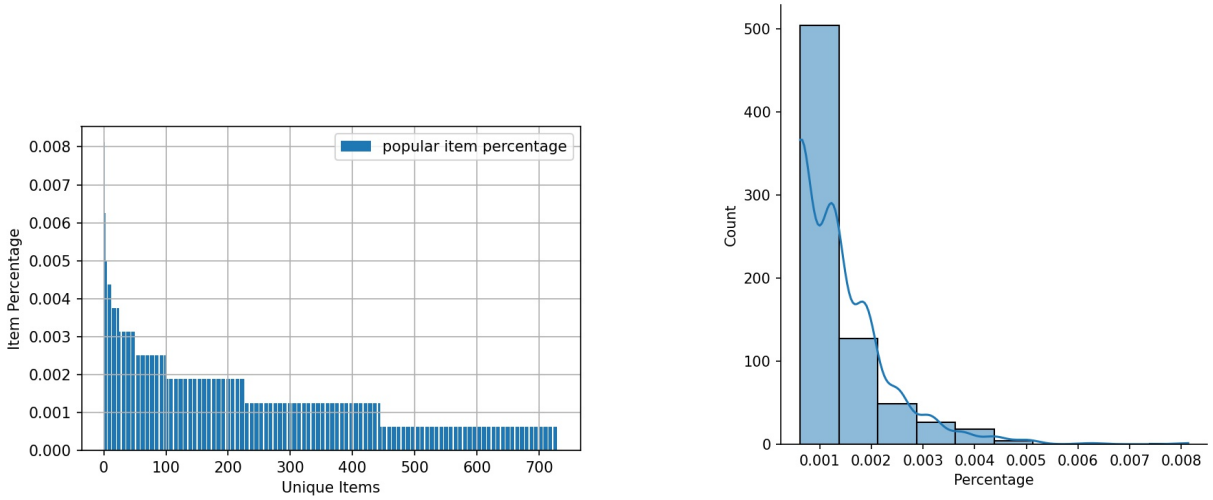
In the case of Base experiment, user frequency is more uniformly distributed across *user ids*. The top user shows up 130 times, amounting to 0.35% of total user appearance, while the last one only 10 times. More than 30% of the total *user ids* appear between 40 and 60 times and about 25% from 70 to 85 times. Around 75% of the users appear 40 to 85 times. Distribution is again bell-like shaped, but this time there is also a left and a right tail. In the center of multiple ridges do appear.



(a) Item Frequency of Occurrence (b) Item ID Frequency of Occurrence Distribution

Figure 3.6: FOEIR Experiment Item ID Frequency over 300 Iterations

Figure 3.6 (a) shows the total number of unique *item ids* that are labeled as popular. As discussed in the methodology chapter, items receive a new label during each recommendation round. On the vertical axis we may find the percentile probability of a specific *item id* to appear, when a popular item occurs in a final recommendation list. In the FOEIR experiment 5113 recommendations concern popular items across all 300 iterations, bootstrap phase excluded. The unique ones are 797 or 25% of the total unique items. The most frequent *item id* appears 38 times, or approximately 0.74% of the total times. In essence, figure 3.6 presents the probability distribution of popular *item ids* popularity. Over 85% of the unique *item ids* have a probability of up to 0.2% to appear as a popular recommendation.



(a) Item Frequency of Occurrence (b) Item ID Frequency of Occurrence Distribution

Figure 3.7: Base Experiment Item id Frequency over 300 Iterations

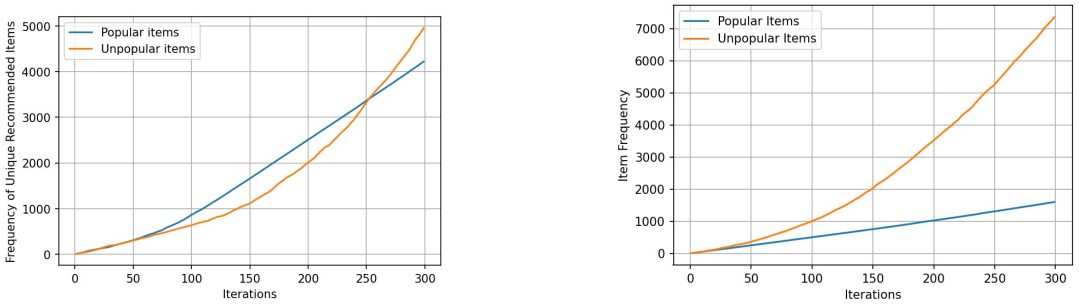
In the Base experiment, recommendations of popular items appear 1599 times. The number of unique popular suggestions is 730, meaning that 45% of the times a popular item occurs in the final recommendation list, it is characterized as popular for the very first time. Roughly, 70% of the *item ids* have a little bit more that 0.1% chance to be a popular recommendation. An interesting observation, is that although the proposals of popular items in the FOEIR experiment are three times more, the unique popular recommendations outnumber them only by 10%.

The fact that Base experiment exhibits that less popular items, may be attributed to that only a few k of them are enough to cover the 40% of total item popularity per iteration that is required to characterize an item as popular ones. Though, over the course of iterations, there are the same *item ids* labeled again and again iterations, as popular ones. In a sense FOEIR post process algorithm, distributes the item popularity among items in a more uniform way, though the items that are considered popular over time, carry the same *item ids*. In the Base experiment, we have very few different items as popular ones in each iteration. In FOEIR we have many items considered as popular, but they actually are the same ones in every iteration.

This, causes the list with popular items, temporarily be more diversified. The important question that arises now is, what happens when those specific *item ids* are recommended and therefore clicked? Are those the ones getting overexposed? Does this phenomenon leads to a propagation of popularity bias in favor of those specific items?

3.2 Recommended Items

3.2.1 Popularity Analysis



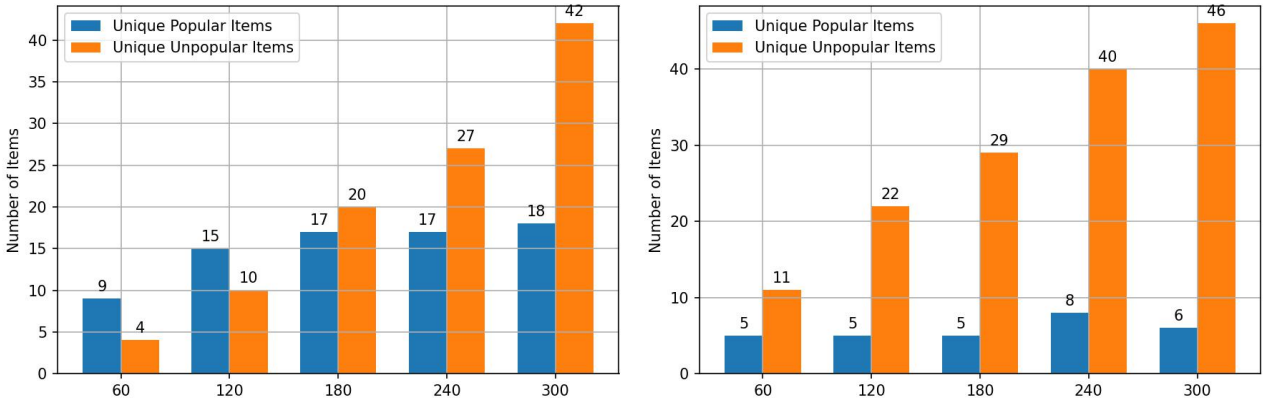
(a) FOEIR Experiment (b) Base Experiment

Figure 3.8: Number of Final Unique Recommended Popular & Non Popular Items

At this point we focus our gaze into the final recommendation list that each user receives and try to provide an intuition behind the characteristics of those final recommended items. Figure 3.8 depicts the cumulative number of final unique recommended items. In subfigure (a), up to the $60th$ iteration, the number of suggested popular and non - popular items is equal. Between the $60th$ and the $250th$ iteration, popular ones are clearly more frequently proposed to users.

In the last 50 rounds, non popular unique recommended items surpass them. During these iterations the raw data suggest that unique non popular recommended items are approximately three times more than the popular ones. Both subfigures, after the initial iterations, exhibit an increase at a constant rate regarding the number of popular recommended items. In the first 50 to 60 iterations the unique number of both groups is the same. At a first glance the latter seems to contradict with the fact that the ratio between general unique popular and non popular items reaches its highest value during that period of time. A possible explanation for this phenomenon stems from the fact that Base recommendation system, during its early rounds, has not yet received sufficient user feedback information.

The most popular recommended and clicked items, still do not dominate the short post-processed recommendation list produced by FOEIR algorithm. Neither it does, in the case of the experiment, that we do not employ any fairness constraints in the final proposed results. In other words model bias propagation effects are yet to be observed [42].



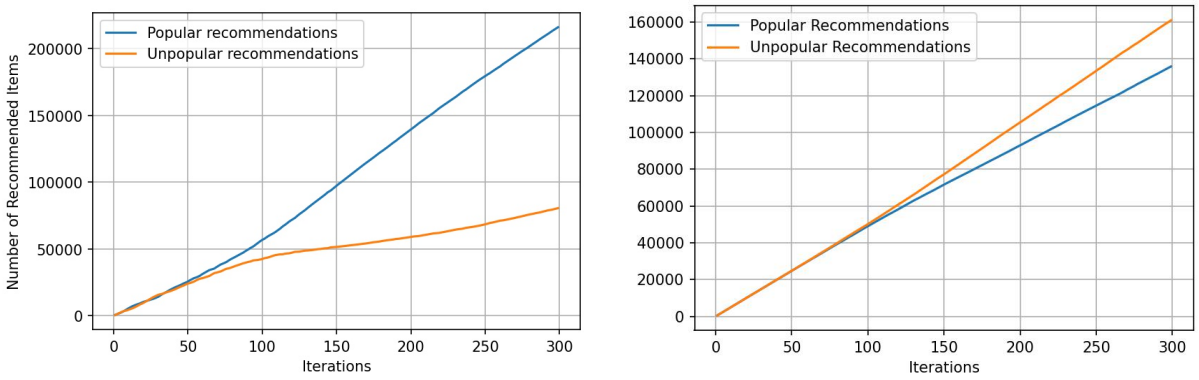
(a) FOEIR Experiment (b) Base Experiment

Figure 3.9: Unique Number of Recommended Popular & Non Popular Items over Iterations

Figure 3.9 shows, that after the initial iterations, as more items are appended in the train set, the number of non popular recommended items starts to rise. Even in the FOEIR experiment case, the number of popular recommended items per iteration after a certain point remains completely unchanged. Despite that from the $120th$ to $300th$ iteration the number of proposed popular items is stable as number of non popular recommended items constantly increases. It is only after the $250th$ iteration that the cumulative number of non popular unique recommended items surpasses the popular ones. As discussed before, although the number of popular recommendations in FOEIR case is almost triple than in the Base experiment, based on the way labels to items are assigned, it signifies that FOEIR allocates popularity more evenly amongst its items. Despite that, it still recommends those popular items, even if their popularity score is much less than in the Base Experiment case.

An uncommon phenomenon in the world of recommendation systems that contributes to this direction, is the saturation of popular items. I dedicate a whole separate subsection for this matter, but in a few words, Base recommendation algorithm, tends to propose to the same user items that has not been suggested before. Thus, in the later stages of the experiment, a number of users that participate in the specific round of suggestions, might not be able to receive as a recommendation a significant percentage of the popular items. In the Base experiment case, as far as subfigure 3.8 (b) is concerned, the line that shows the cumulative increase in the number of non popular recommended items seems to follow the same pattern as the one in subfigure 3.8 (a).

The number of popular items is so low that despite being recommended in top ranks, the rest positions in the list are filled in with the non popular ones. Saturation in this case, seems to have an even more significant role. The percentage of popular items that are being saturated for some users might very much affect the final recommendation list.



(a) FOEIR Experiment

(b) Base Experiment

Figure 3.10: Final Recommendations of Popular & Non Popular Items

Figure 3.10 shows the cumulative number of final recommended items. At a first glance in subfigure (a), the popular suggested items line goes along the non popular one up to the same timestamp as in figure A.2. From that point and onwards, unlike the unique recommended items case, the total cumulative popular recommendations surpass the non popular ones. In the FOEIR experiment the number of popular items allows the recommendation algorithm after a sufficient number of iterations to include all of those items through the user feedback in the train set. After this point the gap between those two item categories seems to constantly increase up to the $300th$ iteration, where there are cumulatively 216.031 popular suggeestions made to users versus only 80.529 non popular ones.

Comparing subfigure 3.10 (a) and 3.8 (a), validates observations made in 3.7(a), about the difference in the total popular items and the total unique ones. Even though the number of unique popular items remains the same and the number of non popular ones increases, users at the end of the day were served with 2.6 popular recommendations for every non popular one. In the Base experiment, after the $115th$ iteration the cumulative non popular recommendations exceed the popular ones. Their difference slightly increases over the course of time. Figure 3.8 (b) showed that non popular unique recommended items are 4.6 times more than the popular ones. In this case though they are only 1.2 times more. We may conclude that during each iteration, in the Base experiment only a handful of items are labeled as popular,though their number is not sufficient to overcome the non popular ones. FOEIR experiment, after a certain point shows at least twice the amount of unique popular items, thrice unique popular recommended items and approximately 1.6 times more total popular recommendations.

3.2.2 Exposure Analysis

Despite the fact that FOEIR postprocess algorithm does not entirely succeed in mitigating the over recommendation of the popular items in the long run, the proposed fairness framework is meant to balance the exposure between the protected and the non-protected item groups. Apart from identifying whether a popular item is recommended or not, it is really essential to also observe the rank position these items appear in a recommendation list.

As it is discussed in the *Methodology* chapter, objects placed on top of the list are more probable to be observed and therefore clicked by the user. To address position bias I assign a descending exposure - attention score values $\frac{11-k}{10+1}$, with $k = 10$, representing the position an item appears in the final recommendation list. Visualizations provided below, will try to enlighten us on whether popular items are indeed except from recommended, also exposed more to users. How is the exposure score accumulated for the top and less exposed items throughout time? Do the items that are randomly more exposed during the bootstrap phase also stay the most exposed ones over the course of time?

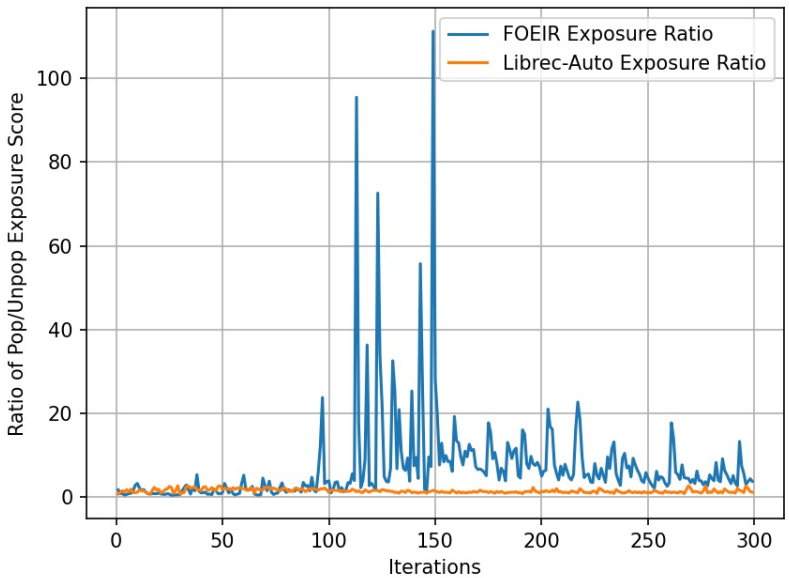


Figure 3.11: Popular & Non Popular Exposure Ratio over Iterations

Figure 3.11 depicts the cumulative item exposure ratio over time for the popular and non popular items. For each iteration I aggregate the total exposure score for items of both groups and divide them to produce the aforementioned ratio.

$$\frac{Exposure((G_) \mid i_p))}{Exposure((G_u \mid i_u))} \tag{3.1}$$

Up until the $93th$ iteration, aformentioned value for both experiments ranges from 0.8 to 1.4, with the blue line to appear higher fluctuations. From that point and on and particularly for the next 50 iterations in the case of FOEIR experiment the ratio skyrockets in favor of the popular items. There are numerous spikes, with the higher one to appear in the $149th$ iteration, where popular items exposure score is 111.2 times more than the non popular one. Its average ratio across every iteration is 6.9, though the occurrence of many outliers causes this value to highly increase. Over the 50% of the recommendation rounds, this fraction receives the value of 4.5. In the Base experiment, the mean exposure score ratio is 1.3, approximately the same as the median as well.

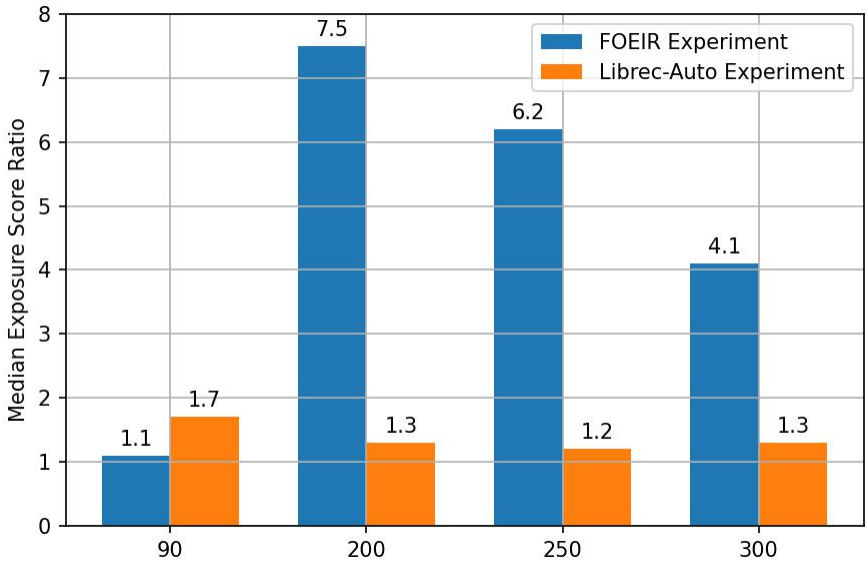


Figure 3.12: Popular & Non Popular Median Item Exposure Ratio over Iterations

Clearly, throughout the whole tenure of FOEIR experiment popular items are overexposed almost seven times more. Nonetheless, performing the same type of analysis for specific time-frames produces quite different results. Up till the $90th$ iteration the median value of the ratio for the blue line is at 1.1. From $90th$ up to the $200th$ point, exposure ratio succeeds its larger median value 7.5. During the last 50 iterations, this number noticeably reduces up to 4.1, though still remains far from the desirable outcome that FOEIR's notion of fairness in terms of equity of exposure suggests [54]. Base experiment achieves its higher median ratio 1.7, between the bootstrap phase and the $90th$ iteration. Throughout the rest of the rounds, it remains stable around 1.3.

As it is discussed in subfigure 3.8(a) and 3.9(a), after a certain amount of iterations,the post process fairness algorithm, seems to fail to maintain the balance, between popular and non popular unique & total recommended items, as also in terms of exposure score allocation. On the contrary, figure 3.12 shows that the collapse does not come at the $60th$ iteration, but around the $90th$ point.

In figure 3.8 (a), over exposure of popular items seems to decline after the $250th$ iteration mark, fact that may be as well attributed to the vast increase in the number of unique non

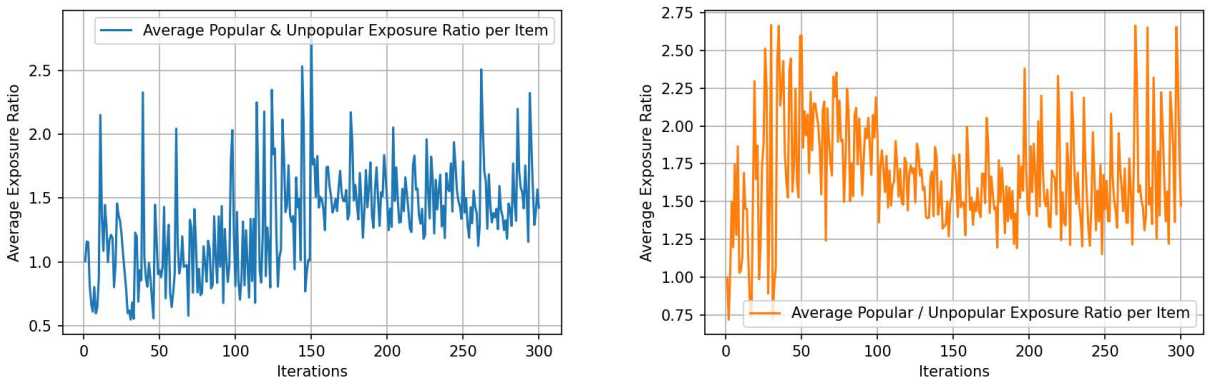
popular items and the effect of saturation, that it is around that point that starts to have a real impact. The average exposure score per iteration for the popular and non popular items in the FOEIR experiment is 373.5 and 122.3 respectively. Their corresponding ratio value is 3.05. This number may be interpreted in the following way: popular recommended items gather 305% more exposure score than the non-popular ones on average across all iterations. Using solely the base recommendation pipeline the value of the latter ratio is 137%.

This method of computing the average exposure for the two different item groups though, is greatly affected by the consequences that popularity bias causes, as the number of popular recommendations is much higher, especially in the FOEIR case in which the notion of popularity that is adopted causes the number of recommended popular items be three times more than in the base recommender experiment case after a certain time point. To overcome the aforementioned issues and to report a metric that accurately reflects the exposure bias that both experiments exhibit, I compute the average exposure score gained per item for the two different item groups across all iterations.

$$\frac{Exposure((G_p) \mid i_p))}{|G_p|}, \tag{3.2}$$

$$\frac{Exposure((G_u \mid i_u))}{|G_u|} \tag{3.3}$$

With minor discrepancies, figure 3.13 (a) shows that up to the $110th$ iteration, the mean exposure score each item contributes is around 0.5 for both protected and unprotected groups. Hence the average ratio is 1, providing a strong sense of fairness of exposure allocation in terms of demographic parity, as it is proposed in the work of [54].



(a) FOEIR Experiment

(b) Base Experiment

Figure 3.13: Popular & Non popular Exposure Ratio per Item over Iterations

The same value though taken from the $120th$ iteration and onwards surpasses the point of 1.5, denoting that popular items receive 50% more exposure on average for approximately the two thirds of the experiment in terms of duration. In subfigure 3.13(b) the average ratio reaches the 1.5 value only after 30 rounds of recommendations. Henceforth, the latter ends up being 1.7, taken across the whole tenure of the experiment.

Last figure analysis, lead us to the conclusion that up to a certain time point disparate in impact fairness constraint ensures an on average equality of exposure between the protected and unprotected item groups both in a deterministic and in an expectation level. Thus, although the notion of fairness from the object perspective may be fulfilled for the first 120 iterations, it is not enough to guarantee in the long run. Though, FOEIR mitigates the effects of model

bias, as it requires four times more number of iterations to overexpose popular objects to users. To monitor exposure's distribution throughout time, I present the cumulative score gain graph for the three most and least exhibited items over time.

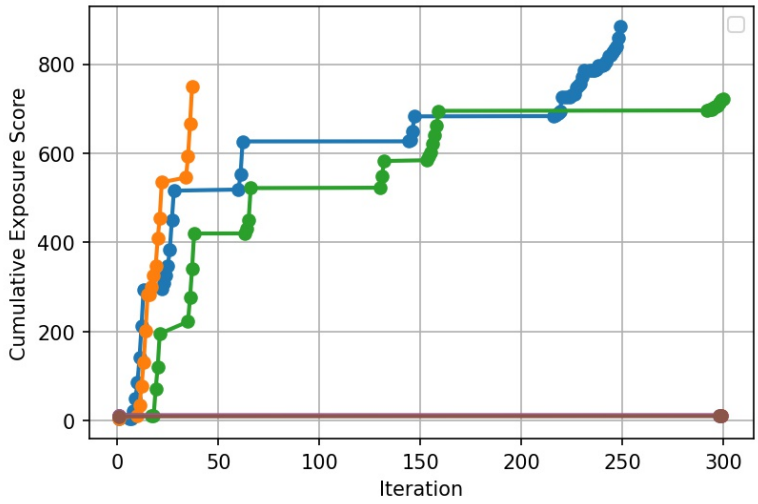


Figure 3.14: FOEIR Cumulative Item Exposure for Top & Bottom three Items

In figure 3.14, blue line points represent the 52 iterations, in which the most exposed item during FOEIR experiment, makes the final recommendation list in a position from 1 to 10. Its total gained score is 885.27, averaging around 17.02 exposure score and 38 occurrences per iteration. Unlike the latter, the second most exposed item pops up only up to the $37th$ iteration of the experiment. It manages to cumulatively score 748.18 exposure over 18 unique iterations. Approximately 87 occurrences account for 41.5 exposure score per iteration.We observe the pattern, that a single item is getting overexposed for two or three consecutive iterations to approximately all the users, but after that it stops to appear for a significant amount of time. Despite the fact that the users are randomly picked per iterations, we may assume that this occurs as we expect those users to have already consume this product so there's no point in recommending it again. The third most exposed item, depicted with the green line, gathers its exposure across the whole spectrum of the experiment. Its total exposure score is 712.81 is collected across 33 iterations.

The accumulated exposure score of the top 10 items accounts for the 3,5% of the total exposure score gained by all unique items throughout the FOEIR experiment. This percentage goes to 4,1% when we remove the items that do not appear after the bootstrap phase. The top 28% of the most suggested items, though, accounts for 88% of the total exposure score.

Figure 3.15 shown below, depicts the average position in rank top three exposed items achieve in each iteration they appear. The brighter the color of the box is, the higher the rank of the item in the given iteration. A white square represents position 1 and dark red colored ones the $10th$ rank. At a first glance for all three cases the lighter colored boxes dominate the heatmap, whereas the dark red color seems to pop up only three to four times across all the iterations they appear.

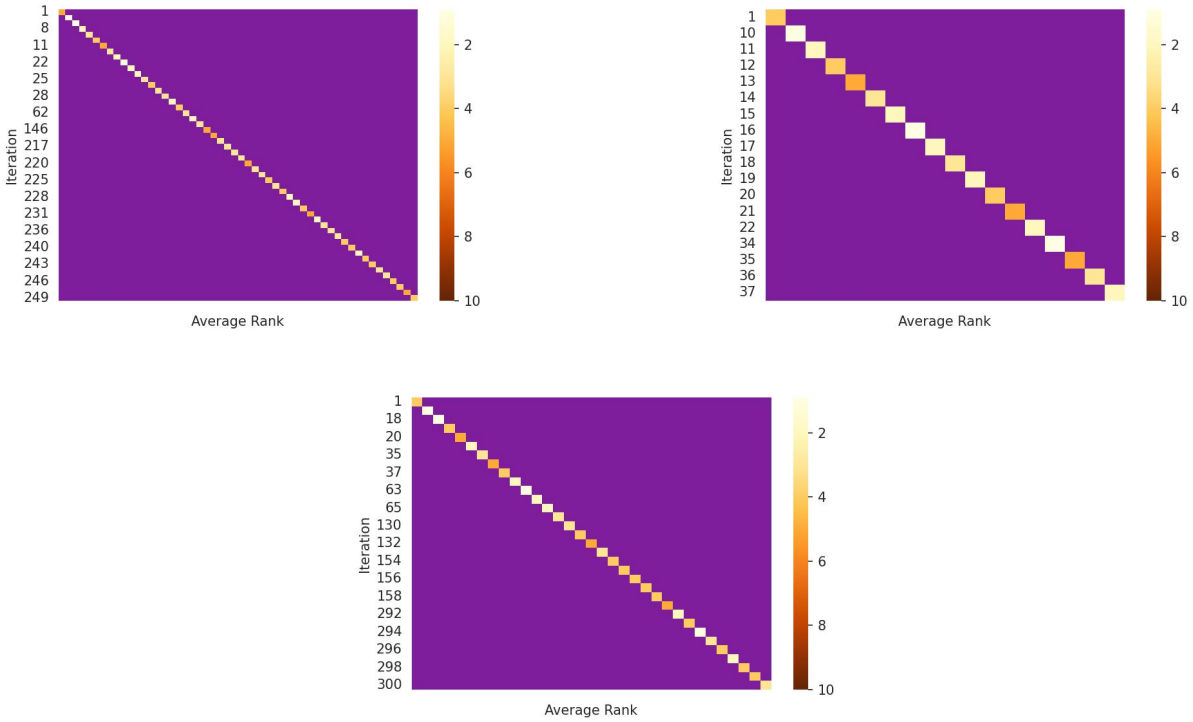


Figure 3.15: FOEIR Average Rank per Iteration for Top 3 Exposed Items

The three items that have actually gathered the least amount of exposure and appear again in a final recommendation list after the bootstrap phase, may only be found during the first and the very last iterations. Their corresponding rank appears in figure 3.17.

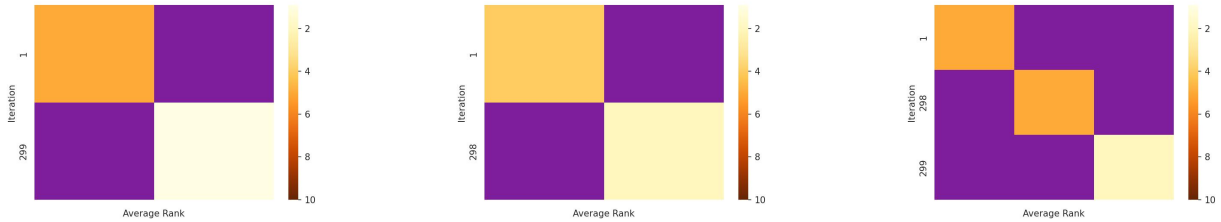


Figure 3.16: FOEIR Average Rank per Iteration for Bottom 3 Exposed Items

In the Base recommender case at first sight the top suggested items collect their exposure in the same way as depicted in figure 3.14. The most exposed one, accumulates 853.36 score, over 42 separate iterations, the second one 633.27 and the third 542.9, in 15 and 25 rounds respectively. When it comes to the least exhibited items, the same pattern applies as in the FOEIR experiment. All three items appear during the bootstrap phase and after that only in the very last 4 to 5 iterations.

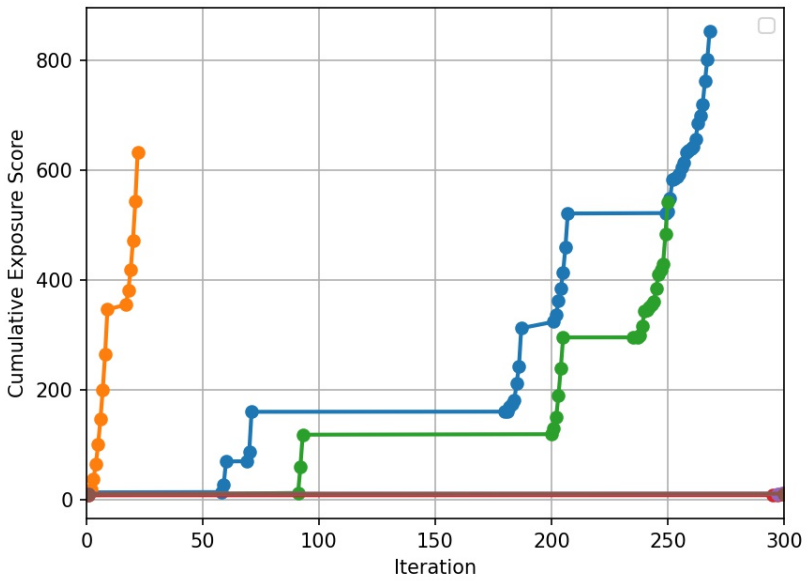


Figure 3.17: Base Experiment Cumulative Item Exposure for Top & Bottom three Items

First ten displayed items make up for the 3.1% of the total exposure score gained from all unique item ids. By removing the items that do not appear again after the initialization step, this percentage reaches the 4%. In a similar manner, this time 29,5% of the most exposed items receive the 88% of the total exposure score.

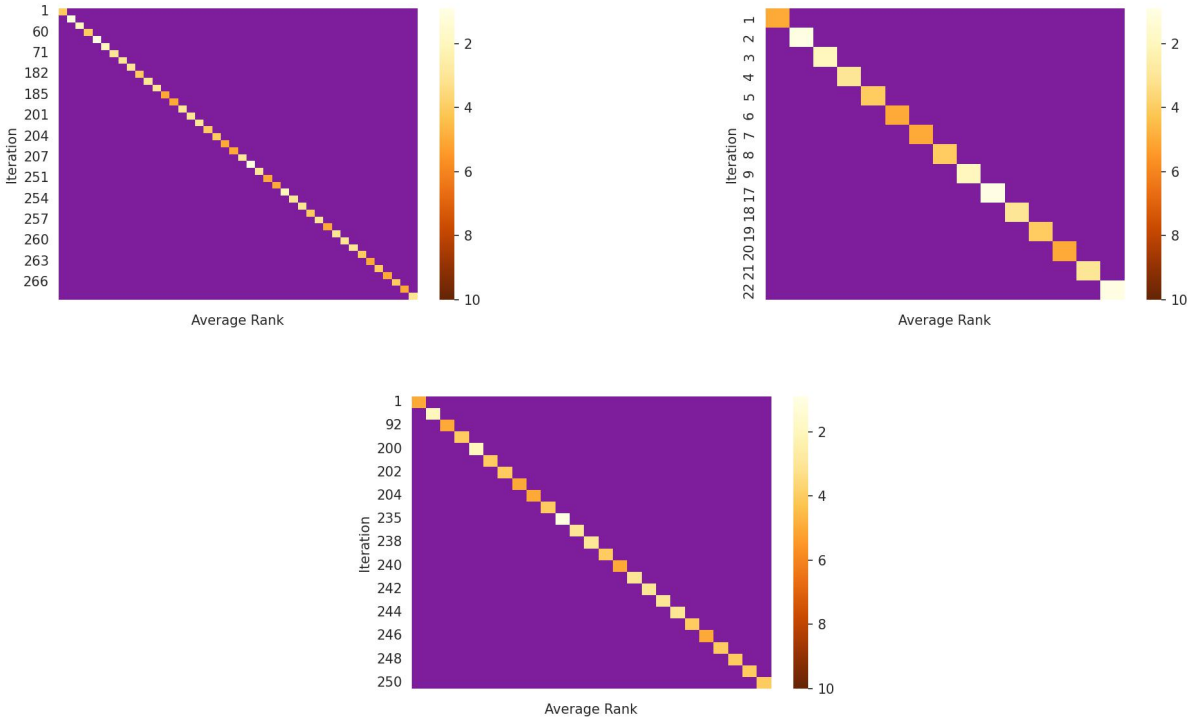


Figure 3.18: Base Experiment Average Rank per Iteration for Top 3 Exposed Items

Comparing figure 3.15 with the above one, we observe less light colored boxes. The number of times that on average the rank is from 8 to 10 is approximately the same. Though the extremely bright painted squares are profoundly more in the first case. In the case of the least exposed items, as expected, they only appear in a handful of iterations on an average rank of 3.

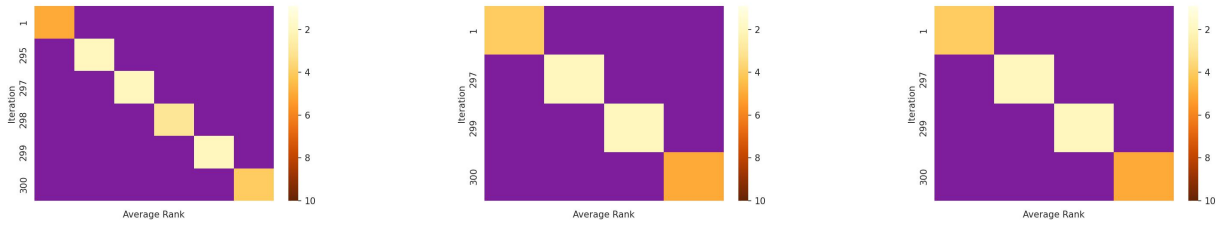


Figure 3.19: Base Experiment Average Rank per Iteration for Bottom 3 Exposed Items

To sum up the outtakes from the last four figures, in both experiments the top exposed items seem to gain their score in a similar way. In the FOEIR case, the most exhibited items account for a larger percentage of the total accumulated score, as according to the provided heat map graphs, they achieve higher rank on average in the user lists that they appear. Visualizations shown below, will try to provide an intuition on whether randomly generated recommendation lists during the bootstrap iteration do truly affect the final cumulative exposure score rank of unique *item ids*.

Bootstrap Rank	**Final Rank**	**Bootstrap Exposure**	**Final Exposure**
1	667	19.6	108.9
2	868	18.7	18.7
3	869	18.4	18.4
4	410	18.3	163
5	870	18.1	18.1
3075	3077	3.5	3.5
3076	3079	3.3	3.3
3077	3078	3.3	3.3
3078	20	3.3	441.2
3079	444	3	151.2
Pearson's Correlation		**0.72**	

Table 3.1: FOEIR Exposure Score Rank Correlation

To produce the table above, I take the 10 most and least exposed items during the bootstrap phase and try to monitor their rank after the 300 iterations mark. Their corresponding values are presented in the Bootstrap Rank and Final Rank columns. Bootstrap Exposure and Final Exposure labels contain the absolute exposure score the aforementioned items gained throughout the whole experiment. Although only the attributes of ten items are displayed the following Pearson's Correlation score of 0.72 is produced by taking into account the 20 *item ids*.

Bootstrap Rank	Final Rank	Bootstrap Exposure	Final Exposure
1	895	20.9	20.9
2	897	19.8	19.8
3	899	18.4	18.4
4	900	18.1	18.1
5	901	18	18
3075	3077	3.5	3.5
3076	3079	3.3	3.3
3077	3078	3.3	3.3
3078	20	3.3	441.2
3079	444	3	151.2
Pearson's Correlation		**0.76**	

Table 3.2: Base Experiment Exposure Score Rank Correlation

Not applying the post process fairness constraints, causes correlation to be higher, reaching the point of 0.76. In both cases there is no correlation at all between the items ranks during the initial phase and the final one, when items that appear only during the first iteration, are excluded, as shown in tables A1 and A2 in the Appendix A chapter. As we may see the Bootstrap Exposure values in both experiments are approximately the same as the final exposure ones, causing the degree of relation to reach the aforementioned values.

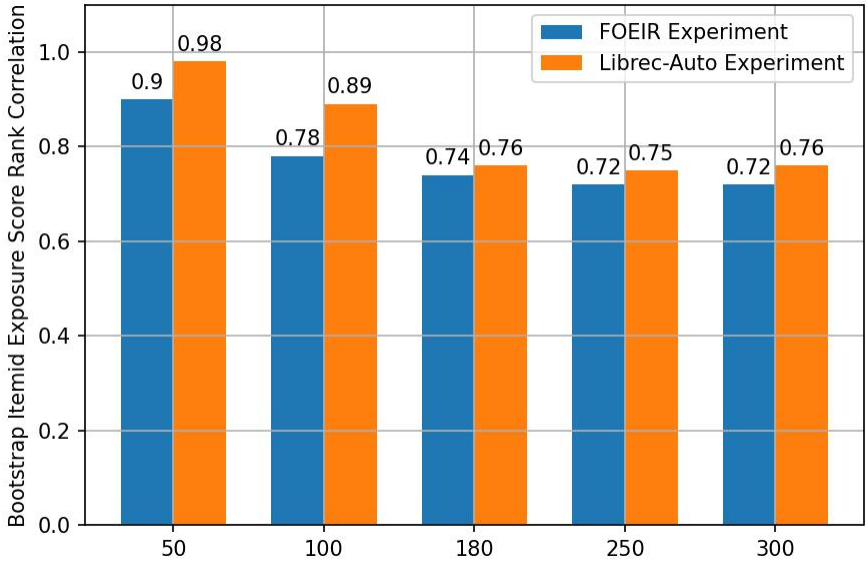


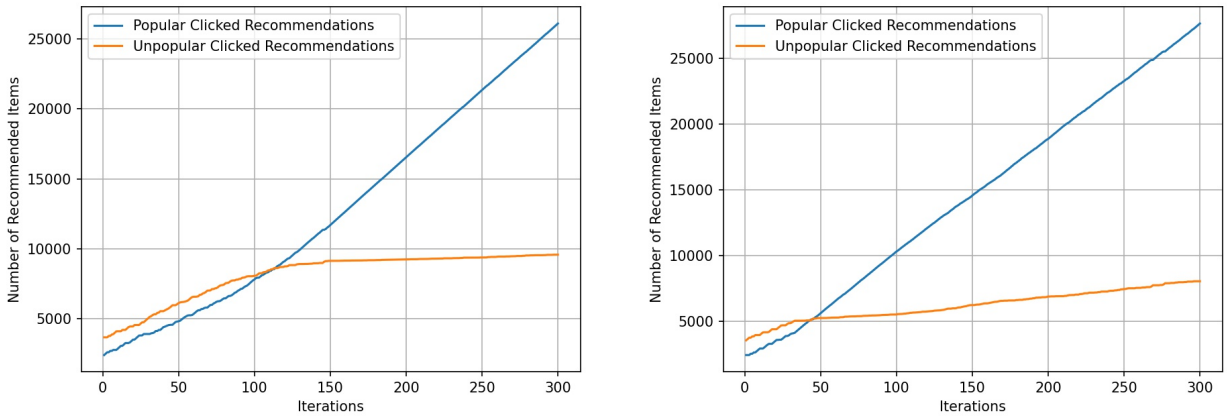
Figure 3.20: Bootstrap and Final Rank Correlation based on Cumulative Exposure Score over specific Time frames

Tables 2.1 and 2.2 exhibit the degree of correlation between the very start and the finish line items rank in terms of exposure. Through figure 3.20 I try to provide a sense of the aforementioned metric in the course of time. As it is expected in the early stages, correlation value is really high as the effect of the bootstrap phase in the rankings is still very impactful. At the $50th$ iteration mark, correlation for the FOEIR experiment lies at 0.90 and for the Base one around 0.98. After the next 50 iterations the blue bar drops for 0.12 and the purple one for 0.10. At the 180th recommendation round, correlation values roughly reach the scores initially

reported after the final iteration. Across every instant, the base recommender experiment displays higher correlation values, suggesting that originally randomly assigned most exposed items tend to also be more exposed than others even after 300 iterations, when the post process re ranking algorithm is not employed.

3.3 Clicked Items

In both experiments and for every iteration, we consider the user clicking the top ranked item in its recommendation list. Those clicked items are appended to the train file of the next iteration and are used as input for the base recommender system to produce the long recommendation list of size 40. During bootstrap phase, 6040 items are clicked. In the next 300 iterations 100 of them are clicked, producing in total 36040 clicked items, per experiment. Figures presented below will try to provide an intuition behind the label of those items, the exposure they get and its distribution throughout time. In total 2652 unique item ids have been clicked throughout the FOIER experiment and 2640 in the Base one, accounting for approximately 86 % of the total items in both cases. Around one sixth of the clicks happens during the bootstrap phase.



(a) FOEIR Experiment

(b) Base Experiment

Figure 3.21: Cumulative Number of Popular & Unpopular Clicked Recommendations over Iterations

The total cumulative number of clicked popular items is 2.73 and 3.42 times more than the non popular ones, as shown in subfigures 3.21 (a) and (b) respectively. In the FOEIR experiment the blue line surpasses the orange one after the 108 iteration. In the second graph the aforementioned phenomenon happens way earlier again at the 48th iteration. We observe a similar pattern as it is more extensively presented in the cases of recommended items in subsection 3.2.

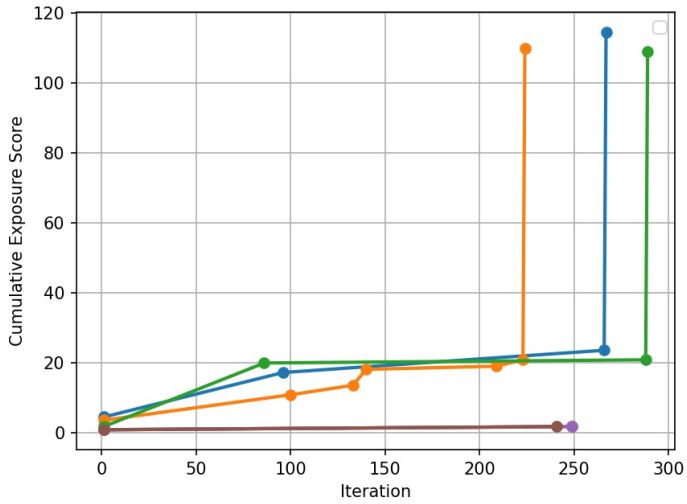


Figure 3.22: FOEIR Cumulative Clicked Item Exposure for Top & Bottom three Items

As in the previous subsection the above figure shows the cumulative gain of exposure score gathered by the top and bottom three clicked items. On the contrary with the analysis of subfigure 3.14, this time the score is gained only in a handful of separate time points. More specifically, all three most exposed clicked items acquire more than 80% of their tally during their final iteration of appearance. Before the last 70 iterations they had around 20 exposure score. After that timestamp, they are recommended to almost all users that participate in that round, earning more than 90 score.

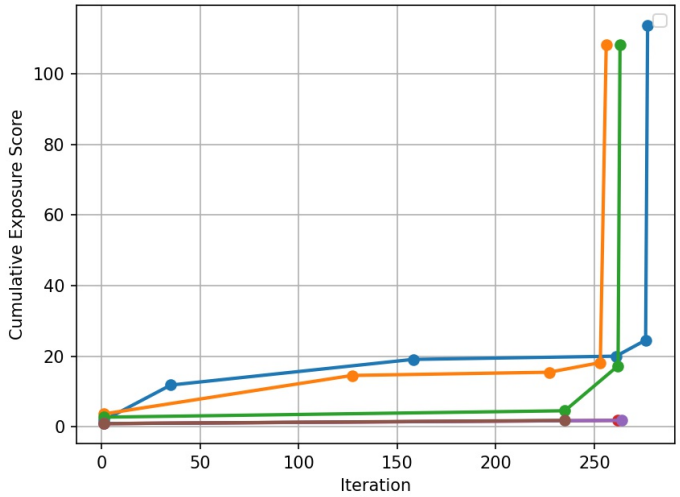
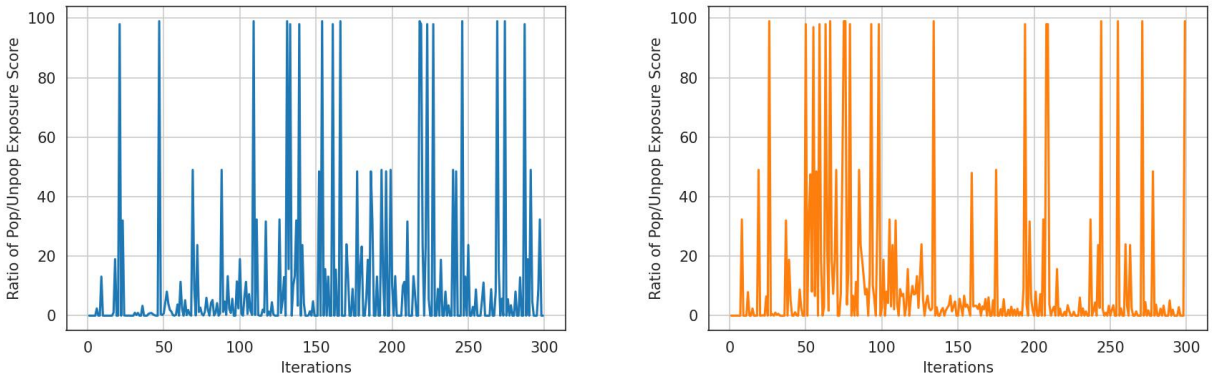


Figure 3.23: Base Experiment Cumulative Clicked Item Exposure for Top & Bottom three Items

As far as base recommender experiment is concerned, the distribution of exposure for the top three items happens in the same manner as before. Through a single of the final 45 iterations, all items secure more than 75% of their total exposure. Least exposed items in both cases act like figure 3.13 in the recommended items section. They appear at an early iteration and then

only at the very end of the experiments. Again the first ten most exposed items cumulatively account for the 3.3% and 3.2% of the total exposure of the clicked items, respectively for figures 3.22 and 3.23. As it is expected, for the top exposed items that are clicked the average rank per iteration they appear is 1. Their corresponding heat map graphs look very much alike figure A.3 in the Appendix A chapter.



(a) FOEIR Experiment

(b) Base Experiment

Figure 3.24: Popular & Non popular Clicked Item Exposure Ratio over Iterations

As in figure 3.11, I report the exposure ratio gained between popular and unpopular items over the course of the 300 iterations, but only for objects that a user has interacted with. In subfigure 3.21 (a), 166 non-zero ratio values are depicted. Only in 10% of those cases the popular clicked recommendations score is zero. Thus in 40% of the iterations, there isn't a single non-popular clicked item by a user. On average, when both groups receive a tally, popular items are 20 times more exposed. The value that separates the higher half from the lower one, in terms of exposure ratio is 7.7. In the line chart presented next, across 60% of total iterations that produce a valid ratio value, the mean is 11.5. In 122 recommendation rounds only popular items are clicked. For the unpopular case the same number is only 10. Lastly, the median value ratio is 2. Both charts present extreme cases in approximately 40% of the iterations, where an item group is not represented, though this is mainly due to the nature of the click model that is used in the experiments.

The difference presented in the average values may be attributed to the fact that the Base experiment recommends a minimal number of popular items. Despite that, again the high value blue lines become more dense after the 130 iteration, whereas in the orange case the spike is observed after the $50th$ time mark, in accordance with our analysis in figure 3.13. In the table presented below, I examine whether the items that gathered the most exposure through user clicking during the bootstrap phase are also the ones that are recommended in the higher ranks during the first iteration, but also throughout the tenure of the whole experiment.

Clicked Boot Rank	Rec Boot Rank	Clicked Final Rank	Rec Final Rank
1	84	776	935
2	213	777	1021
3	174	775	990
4	147	6.3	276
5	798	6.3	1048
2648	2279	338	123
2649	341	2547	2655
2650	1562	2548	1459
2651	2792	2549	1738
2652	2414	2652	182
Pearson's Correlation	**0.82**	**0.62**	**0.42**

Table 3.3: FOEIR Experiment Clicked Items Exposure Score Rank Correlation

Clicked Boot Rank label refers to the rank of the initially, top and least ten exposed *item ids*. *Rec Boot Rank* corresponds to the rank the latter objects have after the kickstart of our experiments across all recommended items in terms of exposure. Thirdly, term *Clicked Final Rank* stands for the rank the items in the first column have after the 300 iterations alongside the total cumulative gain of score. *Rec Final Rank* also refers to the latter, but in the case of recommended items.

Clicked Boot Rank	Rec Boot Rank	Clicked Final Rank	Rec Final Rank
1	149	783	983
2	39	802	917
3	97	804	950
4	44	805	918
5	21	803	907
2636	340	687	473
2637	749	2465	1368
2638	929	2375	1515
2639	875	2374	1473
2640	2292	2321	2496
Pearson's Correlation	**0.80**	**0.76**	**0.60**

Table 3.4: Base Experiment Clicked Items Exposure Score Rank Correlation

As expected, for the first column both tables showcase a high degree of relation. The 6040 items that appear first in the recommendation lists are also the ones that gather most of the score across all positions in the catalog. In the Base experiment case, the correlation is still high up till the end, between the first and third column item ranks. Finally, the two last reported Pearson's Correlation values are not high enough to indicate a direct link among them. Higher values, especially in the last two cases in table 3.4, may be attributed to the reranking that FOEIR post process algorithm performs through the imposition of the fairness constraints presented in the Methodology Chapter. Similarly to figure 3.20 the aforementioned values gradually decline through time, ending up taking their final form as discussed right above. To wrap up this section regarding the analysis on the popularity bias both experiments unveil, I provide the two following summary graphs.

Every line presented in the line charts below is produced by plotting the value produced by the division of the cumulative number of popular items over the non popular ones, for the subsequent cases. The blue line represents the ratio between the total number items, characterized so after the generation of the long recommendation list from our base recommender algorithm. Orange one, depicts the cumulative number of final recommendations that makes it in the ten size list. Finally, the green and the red line refer to the cumulative number, of unique final recommended items and the objects that appear first in the recommendation catalog each user receives, respectively.

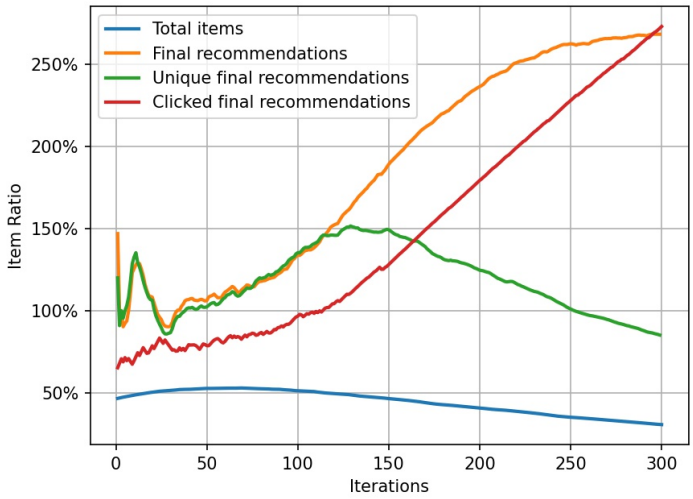


Figure 3.25: FOEIR Experiment Item Ratio per Iteration

The most interesting outtake taken from the above figure 3.25 is the fact that up to the 115th iteration the orange and the green line almost completely coincide. Though just before reaching the 1.5 ratio value, they swing to opposite directions. Next, the growth rate of the number of popular final recommended and clicked items greatly surpasses the non-popular ones, causing for a steep ascent of the orange and red line, gradually up to the 250% point at the $300th$ iteration.

Throughout the same time frame though, the ratio of unique final recommendations drops from 150% to 80%, confirming the interpretation provided under figures 4.7 and 4.8, that after a certain point, in every iteration the same popular items are recommended over and over again in the top spots of the lists for every user that participates in those recommendation rounds. Even though, the ratio between the recommended protected and unprotected item groups surpasses the 100% value only after 38 iterations ; all the above take place just after the time point when the equity of exposure post process fairness algorithm fails to balance the average exposure gain between the popular and non-popular items, as discussed in figure 3.13.

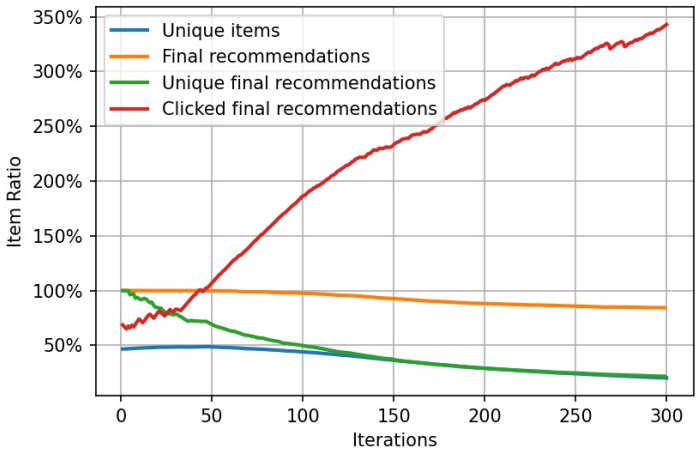


Figure 3.26: Base Experiment Item Ratio per Iteration

Both figures 3.25 and 3.26, show that during the initial iterations the number of non popular items is double, over the popular ones. Gradually this number declines, reaching towards the final iterations the 25% point. Another common remark between those graphs, is the great growth rate the number of popular clicked recommended items exhibit. Especially, in the base recommender case the ascend of the red line is really steep after the $50th$ iteration, ending up reaching 350%.

Popular suggestions eventually dominate the top rank position of the recommendation lists. On top of that the number of unique recommended popular items constantly decreases, causing the same popular items being clicked and thus appended in the train file of the recommendation algorithm. The aforementioned phenomenon causes the base recommendation algorithm to assign greater relevance scores to those popular items, as they exhibit a higher chance of appearance in the train file. Thus the nominator of the fairness constraint as appears in equation 2.11, shows that CTR term, contains the relevance score value, causing its increase in expectation. Lastly, in figure 3.25 we observe a halt in the growth rate of the number of popular recommendations, possibly caused by the impact of the saturation. Throughout the final subsection of the experiments chapter, I will try to provide a more in depth analysis regarding this uncommon phenomenon.

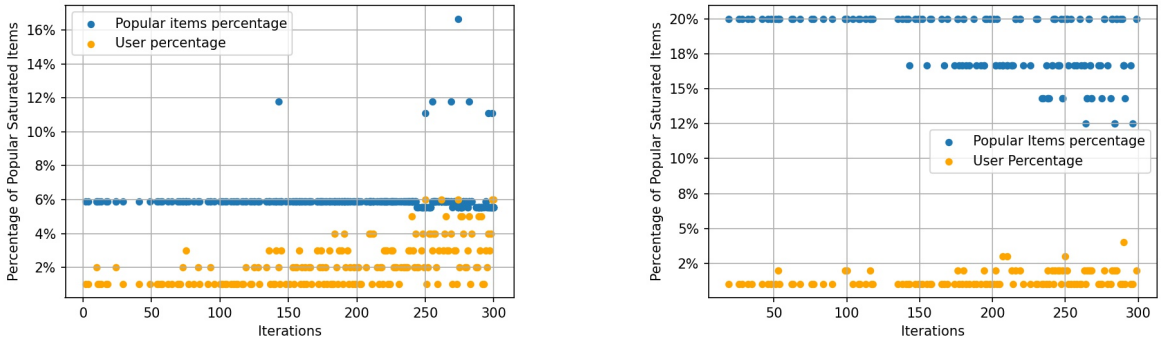
3.4 Item & User Saturation

Once an item is clicked from a specific user, this information is stored in the train set of both recommenders, as an implicit expression of item to user relevance. According to Librec Auto's documentation[1] for the BPR algorithm, the item is considered 'consumed' from the specific user by the recommender, making it not possible to be recommended again in the near future. During the initial recommendation rounds, this is not a severe problem, but after a couple of hundred iterations have passed, the number of user and item entries greatly increases. To monitor the impact of the aforementioned phenomenon , I examine the *user ids* that appear simultaneously in the same iteration's train file and recommendation round, as the train file contains all the *item ids* clicked by user ids throughout all previous iterations. I report numeric values for the two experiments, for the following measures:

[1] `https://librec-auto.readthedocs.io/en/latest/`

- **Popular Item Saturation:** the percentage of popular items that are not able to be recommended to a specific.
- **User Saturation:** the percentage of the users that are unable to receive a specific popular item as a suggestion.

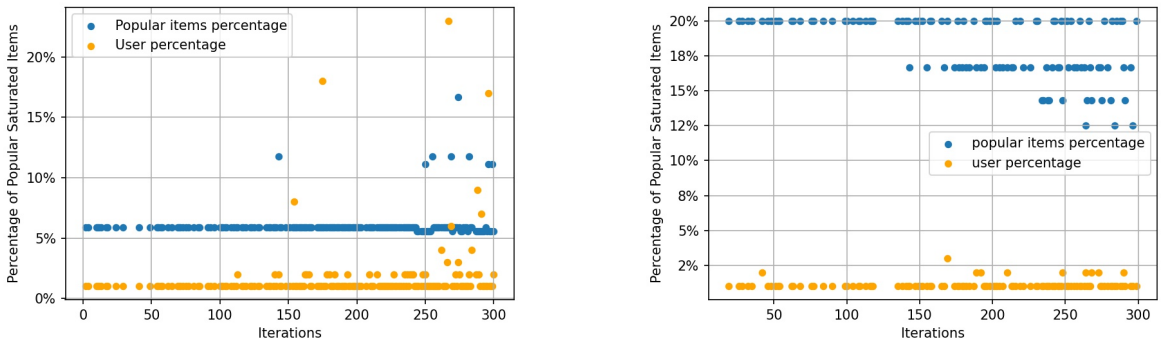
In real world item recommendation scenario, where new popular and non popular items ,as well as new users, are added over the course of time, saturation phenomenon wouldn't exist. In our two proposed experiments though, these two cases are not included, as users and items are not added dynamically to the system.



(a) FOEIR Experiment (b) Base Experiment

Figure 3.27: Total User & Item Saturation Percentages per Iteration

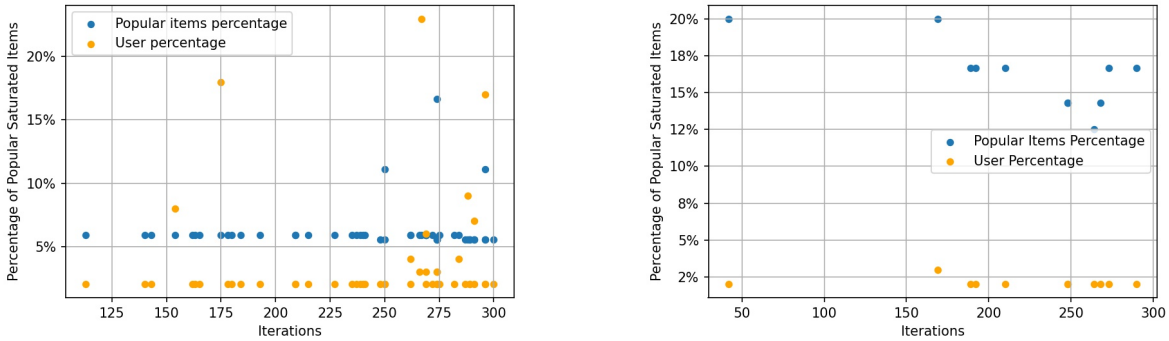
The first presented saturation graph contains the total amount of items and users that are saturated over time with their matching percentages. In each iteration, multiple dots correspond more than user or item. Subfigure 3.27 (a), shows that up until the first 200 iterations the number of saturated users is minimal, ranging from 2 to 4%. During the last 50 iterations this percentage increases to 6%. The blue line appears to be steady around 6%. Only in the very last iterations we observe values that exceed the 12% and the 16% rate. For example in iteration 284 we come across 3 users that are unable to get the 5%, 6% and 17% of the total popular items recommended to them. In the Base experiment case, the extent of user saturation is even less, taking values from 1% to maximum 4%. In the 294th iteration 20%, 16%,14% and 12% of the popular items cannot be suggested to four specific users. In general, even from the initial iterations, item saturation is around 20%, a fact that may be attributed to the very small number of popular items in that experiment.



(a) FOEIR Experiment (b) Base Experiment

Figure 3.28: Maximum User & Item Saturation Percentages per Iteration

Both (a) and (b) in figure 3.28 present the maximum user and item saturation percentages for a single user and item, meaning that a single orange and blue dot corresponds in each unique iteration. Comparing subfigure (a) with the previous one, we may conclude that there are multiple users per iteration that cannot be recommended. Apart from a handful of discrepancies, this rate lies around 6 of the popular items. In subfigure (b) that is not the case. Both graphs look very much alike, thus there are not a lot of users that have saturated items. Below I present the cases, where more than a single popular items cannot be suggested to at least one user per iteration.

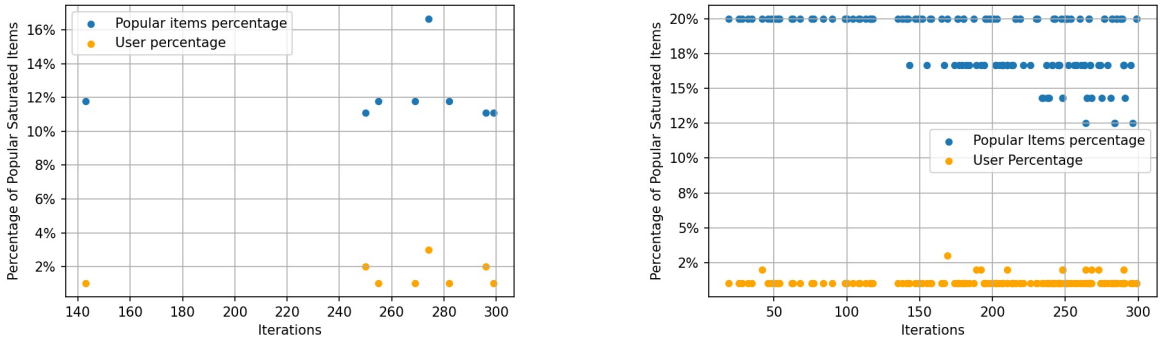


(a) FOEIR Experiment

(b) Base Experiment

Figure 3.29: At least a single Popular Item Saturated for more than a single User per Itearation

In subfigure 3.29 (a) we observe at iteration the $298th$ that 16% of the popular items are saturated for 12 out of the 100 users. On average 6% of the items are saturated, when two or more users cannot appear to have a constraint. In the Base experiment, user percentage is very low in such cases around 2%, though the popular items percentage lies around 16% or more.



(a) FOEIR Experiment

(b) Base Experiment

Figure 3.30: At least two Popular Items Saturated for a single or more Users per Iteration

Finally, in the FOEIR algorithm inquiry only in 8 recommendation rounds the at minimum two popular items are saturated. On the contrary, in subfigure (b) we detect the same phenomenon in about 75% of the iterations, averaging 19% of popular items rate. To sum up, popular items saturation is much more evident in the case of the base recommender system assessment, but a much lower ratio of them cannot be recommended to a greater percentage of users in the FOEIR case.

Chapter 4

Conclusions

A recommendation pipeline that employs the *Fairness* of *Exposure* in *Rankings* post-process algorithm, with the use of *Disparate Impact* fairness constraint, allocates exposure to protected groups more fairly than a base recommender, under a dynamic implicit user feedback setting. Particularly, while it maximizes user's utility and enforces clickthrough rate of each group to be proportional to its average utility in expectation, it provides a strong sense of demographic parity in terms of exposure for approximately 120 rounds of recommendations. After that point, the effects of model and popularity bias eventually prevail, causing an on average 50% more exposure of popular objects. In the case of the Base experiment, this occurs after the $30th$ recommendation round, leading to a mean 70% overexposure of popular items. FOEIR algorithm succeeds in mitigating, up to a certain degree, the effects of the various types of bias that recommender systems suffer from. Another interesting remark, is that the most exposed randomly recommended items during bootstrap phase have a relatively high correlation, in terms of their rank, with the cumulatively most exposed items throughout the whole tenure of both experiments.

To conclude, although FOEIR provides a strong notion of fairness in the short run, equity of attention related constraints, fail to ensure a parity in terms of exposure in the long term. This is mainly due to the nature of the enforced fairness constraint. During each iteration, item's relevance score are computed independently in a static manner. As, recommender systems operate in a dynamic environment, through implicit user feedback, relevance of some specific items is increased, eventually leading to their over-recommendation.

To achieve demographic parity, stricter fairness constraints need to be adopted. Imposing such tough constraints though, may lead to a significant drop in user's utility, as multiple research works have previously shown [54].

For future work, possibly a mixture of stricter and looser constraints may be used interchangeably, over time. When a significant drop in fairness towards objects of a protected group is observed, then demographic parity should be imposed. Once utility starts to decline, disparate treatment constraints may be considered. In any case, even if fairness in recommendations is considered a highly domain dependant subject, information filtering systems should emphasize on object fairness to avoid future plausible winner takes all scenarios.

Acknowledgements

I would really like to thank Masoud for all of his invaluable help and for allowing me to work happily in this project. I would like to also thank M.

Bibliography

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. "Managing popularity bias in recommender systems with personalized re-ranking". In: *The thirty-second international flairs conference*. 2019.

[2] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. "Recommender systems as multistakeholder environments". In: *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 2017, pp. 347–348.

[3] Himan Abdollahpouri and Masoud Mansoury. "Multi-sided exposure bias in recommendation". In: *arXiv preprint arXiv:2006.15772* (2020).

[4] Himan Abdollahpouri et al. "The connection between popularity bias, calibration, and fairness in recommendation". In: *Fourteenth ACM conference on recommender systems*. 2020, pp. 726–731.

[5] Himan Abdollahpouri et al. "The unfairness of popularity bias in recommendation". In: *arXiv preprint arXiv:1907.13286* (2019).

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

[7] Haifa Alharthi, Diana Inkpen, and Stan Szpakowicz. "A survey of book recommender systems". In: *Journal of Intelligent Information Systems* 51.1 (2018), pp. 139–160.

[8] Asim Ansari, Skander Essegaier, and Rajeev Kohli. *Internet recommendation systems*. 2000.

[9] Thomas Southcliffe Ashton et al. "The industrial revolution 1760-1830". In: *OUP Catalogue* (1997).

[10] Ricardo Baeza-Yates. "Bias on the web". In: *Communications of the ACM* 61.6 (2018), pp. 54–61.

[11] Joeran Beel et al. "Research paper recommender system evaluation: a quantitative literature survey". In: *Proceedings of the International Workshop on Reproducibility and Replication in Recommender Systems Evaluation*. 2013, pp. 15–22.

[12] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. "Equity of attention: Amortizing individual fairness in rankings". In: *The 41st international acm sigir conference on research & development in information retrieval*. 2018, pp. 405–414.

[13] Garrett Birkhoff. *Lattice theory*. Vol. 25. American Mathematical Soc., 1940.

[14] Robin Burke. "Hybrid recommender systems: Survey and experiments". In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[15] Robin Burke. "Hybrid web recommender systems". In: *The adaptive web* (2007), pp. 377–408.

[16] Robin Burke. "Knowledge-based recommender systems". In: *Encyclopedia of library and information systems* 69.Supplement 32 (2000), pp. 175–186.

[17] Robin Burke, Alexander Felfernig, and Mehmet H Göker. "Recommender systems: An overview". In: *Ai Magazine* 32.3 (2011), pp. 13–18.

[18] Robin Burke, Nasim Sonboli, and Aldo Ordonez-Gauger. "Balanced neighborhoods for multi-sided fairness in recommendation". In: *Conference on fairness, accountability and transparency*. PMLR. 2018, pp. 202–214.

[19] Jaime Carbonell and Jade Goldstein. "The use of MMR, diversity-based reranking for reordering documents and producing summaries". In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 1998, pp. 335–336.

[20] Pablo Castells, Neil Hurley, and Saul Vargas. "Novelty and diversity in recommender systems". In: *Recommender systems handbook*. Springer, 2022, pp. 603–646.

[21] Carlos Castillo. "Fairness and transparency in ranking". In: *ACM SIGIR Forum*. Vol. 52. 2. ACM New York, NY, USA. 2019, pp. 64–71.

[22] Jiawei Chen et al. "Bias and debias in recommender system: A survey and future directions". In: *arXiv preprint arXiv:2010.03240* (2020).

[23] Yibo Chen et al. "Solving the sparsity problem in recommender systems using association retrieval." In: *J. Comput.* 6.9 (2011), pp. 1896–1902.

[24] Lizabeth Cohen. "A consumers' republic: The politics of mass consumption in postwar America". In: *Journal of Consumer Research* 31.1 (2004), pp. 236–239.

[25] Nick Craswell et al. "An experimental comparison of click position-bias models". In: *Proceedings of the 2008 international conference on web search and data mining*. 2008, pp. 87–94.

[26] Alexander Felfernig and Robin Burke. "Constraint-based recommender systems: technologies and research issues". In: *Proceedings of the 10th international conference on Electronic commerce*. 2008, pp. 1–10.

[27] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. "Are we really making much progress? A worrying analysis of recent neural recommendation approaches". In: *Proceedings of the 13th ACM conference on recommender systems*. 2019, pp. 101–109.

[28] Pratik Gajane and Mykola Pechenizkiy. "On formalizing fairness in prediction with machine learning". In: *arXiv preprint arXiv:1710.03184* (2017).

[29] Guibing Guo et al. "LibRec: A Java Library for Recommender Systems." In: *Umap Workshops*. Vol. 4. Citeseer. 2015.

[30] Vishal Gupta and Shri Ram Pandey. "Recommender systems for digital libraries: a review of concepts and concerns". In: *Library Philosophy and Practice* 2417 (2019).

[31] Jonathan L Herlocker et al. "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53.

[32] Neil Hurley and Mi Zhang. "Novelty and diversity in top-n recommendation–analysis and evaluation". In: *ACM Transactions on Internet Technology (TOIT)* 10.4 (2011), pp. 1–30.

[33] Sarika Jain et al. "Trends, problems and solutions of recommender system". In: *International Conference on Computing, Communication & Automation*. IEEE. 2015, pp. 955–958.

[34] Michele EA Jayne and Robert L Dipboye. "Leveraging diversity to improve business performance: Research findings and recommendations for organizations". In: *Human Resource Management: Published in Cooperation with the School of Business Administration, The University of Michigan and in alliance with the Society of Human Resources Management* 43.4 (2004), pp. 409–424.

[35] Marius Kaminskas and Derek Bridge. "Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7.1 (2016), pp. 1–42.

[36] Neal Lathia et al. "Temporal diversity in recommender systems". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 2010, pp. 210–217.

[37] Byung-Kwan Lee and Wei-Na Lee. "The effect of information overload on consumer choice quality in an on-line environment". In: *Psychology & Marketing* 21.3 (2004), pp. 159–183.

[38] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. "Content-based recommender systems: State of the art and trends". In: *Recommender systems handbook* (2011), pp. 73–105.

[39] Hao Ma et al. "Sorec: social recommendation using probabilistic matrix factorization". In: *Proceedings of the 17th ACM conference on Information and knowledge management*. 2008, pp. 931–940.

[40] Masoud Mansoury et al. "Automating recommender systems experimentation with librecauto". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. 2018, pp. 500–501.

[41] Masoud Mansoury et al. "Bias disparity in collaborative recommendation: Algorithmic evaluation and comparison". In: *arXiv preprint arXiv:1908.00831* (2019).

[42] Masoud Mansoury et al. "Feedback loop and bias amplification in recommender systems". In: *Proceedings of the 29th ACM international conference on information & knowledge management*. 2020, pp. 2145–2148.

[43] Bradley N Miller, Joseph A Konstan, and John Riedl. "Pocketlens: Toward a personal recommender system". In: *ACM Transactions on Information Systems (TOIS)* 22.3 (2004), pp. 437–476.

[44] Abbe Mowshowitz and Akira Kawaguchi. "Bias on the Web". In: *Communications of the ACM* 45.9 (2002), pp. 56–60.

[45] Weike Pan et al. "Transfer learning in collaborative filtering for sparsity reduction". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 24. 1. 2010, pp. 230–235.

[46] B. Pine II, Bart Victor, and Andrew Boynton. "Making Mass Customization Work". In: *Harvard Business Review* 71 (Jan. 1993).

[47] Steffen Rendle et al. "BPR: Bayesian personalized ranking from implicit feedback". In: *arXiv preprint arXiv:1205.2618* (2012).

[48] Paul Resnick and Hal R Varian. "Recommender systems". In: *Communications of the ACM* 40.3 (1997), pp. 56–58.

[49] Stephen E Robertson. "The probability ranking principle in IR". In: *Journal of documentation* (1977).

[50] Alan Said et al. "Recommender Systems Evaluation: A 3D Benchmark." In: *RUE@ RecSys*. 2012, pp. 21–23.

[51] Badrul Sarwar et al. “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 285–295.

[52] J Ben Schafer et al. “Collaborative filtering recommender systems”. In: *The adaptive web*. Springer, 2007, pp. 291–324.

[53] Richa Sharma and Rahul Singh. “Evolution of Recommender Systems from Ancient Times to Modern Era: A Survey”. In: *Indian Journal of Science and Technology* 9 (May 2016). DOI: 10.17485/ijst/2016/v9i20/88005.

[54] Ashudeep Singh and Thorsten Joachims. “Fairness of Exposure in Rankings”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*. KDD ’18. London, United Kingdom: Association for Computing Machinery, 2018, pp. 2219–2228. ISBN: 9781450355520. DOI: 10.1145/3219819.3220088. URL: https://doi.org/10.1145/3219819.3220088.

[55] John K Tarus, Zhendong Niu, and Ghulam Mustafa. “Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning”. In: *Artificial intelligence review* 50.1 (2018), pp. 21–48.

[56] Ke Yang and Julia Stoyanovich. “Measuring fairness in ranked outputs”. In: *Proceedings of the 29th international conference on scientific and statistical database management*. 2017, pp. 1–6.

[57] Sirui Yao and Bert Huang. “Beyond parity: Fairness objectives for collaborative filtering”. In: *Advances in neural information processing systems* 30 (2017).

[58] Meike Zehlike et al. “Fa* ir: A fair top-k ranking algorithm”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 1569–1578.

[59] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. “A comparative study of TF* IDF, LSI and multi-words for text classification”. In: *Expert systems with applications* 38.3 (2011), pp. 2758–2765.

[60] Ziwei Zhu et al. “Popularity Bias in Dynamic Recommendation”. In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 2439–2449.

Appendix A

Appendix

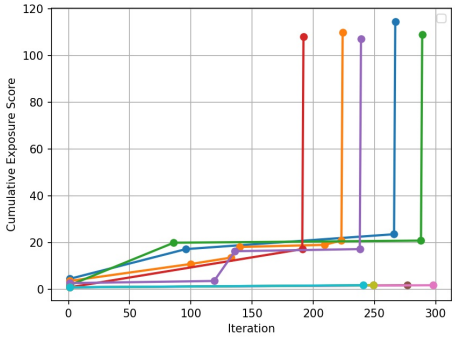


Figure A.1: FOEIR Experiment Cumulative Clicked Item Exposure for Top & Bottom five Items

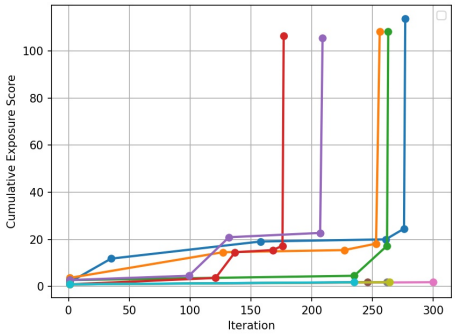


Figure A.2: Base Experiment Cumulative Clicked Item Exposure for Top & Bottom five Items

Bootstrap Rank	Final Rank	Bootstrap Exposure	Final Exposure
1	667	19.6	108.9
2	401	18.3	163
3	515	16.7	136.8
4	86	16.7	322.7
5	94	16.6	317.1
873	1	4.1	885.2
874	759	4	82.4
875	874	3.8	12.7
876	20	3.3	441.2
877	444	3	151.2
Pearson's Correlation		**-0.02**	

Table A.1: New FOEIR Experiment Exposure Score Rank Correlation

Bootstrap Rank	Final Rank	Bootstrap Exposure	Final Exposure
1	296	18.8	198.5
2	500	17.2	143.3
3	193	16.7	239
4	811	16.6	67.6
5	871	16.4	38.5
897	696	5	100
898	120	4.5	286.3
899	310	4.1	3.3
900	228	3.8	224.8
901	721	3.5	92
Pearson's Correlation		**0.07**	

Table A.2: New Base Experiment Score Rank Correlation

Bootstrap Rank	Final Rank	Bootstrap Exposure	Final Exposure
1	488	19.6	108.9
2	107	18.3	163
3	20	16.7	136.8
4	671	16.7	322.7
5	181	16.6	317.1
873	283	4.1	885.2
874	710	4	82.4
875	866	3.8	12.7
876	424	3.3	441.2
877	202	3	151.2
Pearson's Correlation		**0.29**	

Table A.3: NEW FOEIR Experiment Clicked Items Exposure Score Rank Correlation

Bootstrap Rank	Final Rank	Bootstrap Exposure	Final Exposure
1	54	18.8	198.5
2	462	17.2	143.3
3	207	16.7	239
4	763	16.6	67.6
5	815	16.4	38.5
897	450	5	100
898	91	4.5	286.3
899	623	4.1	194
900	346	3.8	224.8
901	453	3.5	92
Pearson's Correlation		**0.18**	

Table A.4: New Base Experient Clicked Items Exposure Score Rank Correlation

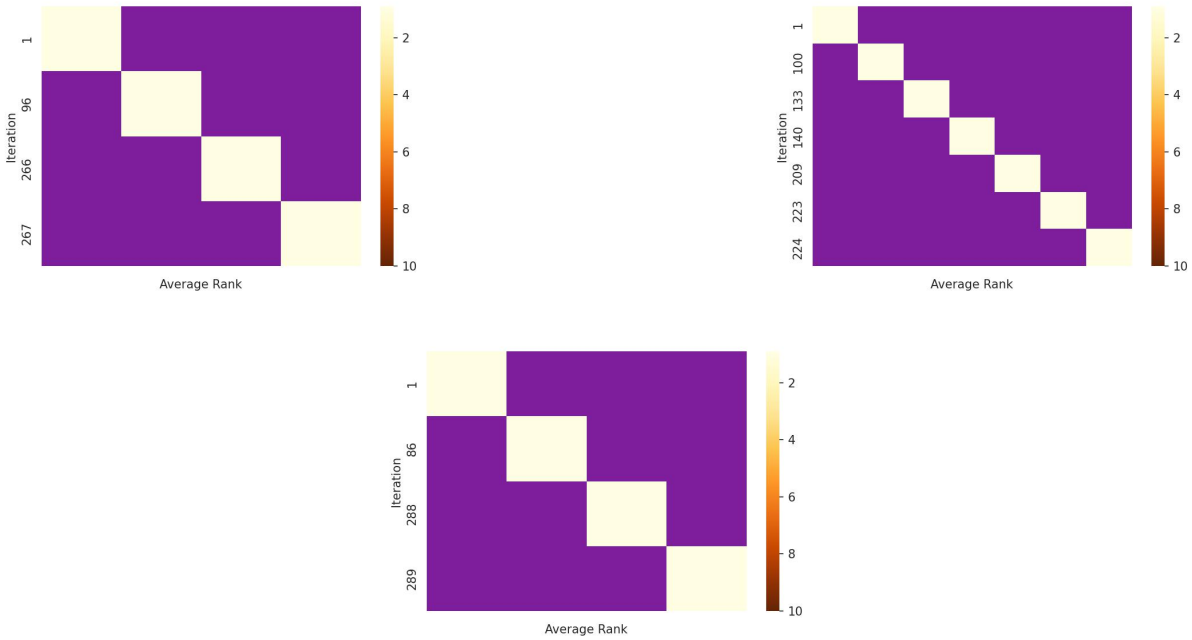


Figure A.3: Average Rank per Iteration for Top 3 Clicked Exposed Items