

Integrating Counterfactual Evaluations into Traditional Interactive Recommendation Frameworks

Yan Andrade[1], Nícollas Silva[2], Adriano Pereira[2], Elisa Tuler[1], Cleverson Vieira[1], Marcelo Guimarães[3], Diego Dias[1(✉)], and Leonardo Rocha[1]

[1] DCOMP/UFSJ, São João del-Rei, MG, Brazil
yrandrade123@aluno.ufsj.edu.br, {etuler,diegodias,lcrocha}@ufsj.edu.br
[2] DCC/UFMG, Belo Horizonte, MG, Brazil
{ncsilvaa,adrianoc}@dcc.ufmg.br
[3] UNIFESP, Osasco, SP, Brazil

Abstract. Online recommendation task has been recognized as a Multi-Armed Bandit (MAB) problem. Despite the recent advances, there still needs to be more consensus on the best practices to evaluate such bandit solutions. Recently, we observed two complementary frameworks that allow us to evaluate bandit solutions more accurately: *iRec* and OBP. The first has a complete set of datasets, metrics, and MAB models implemented, allowing only offline evaluations of these solutions. However, the second is limited to a few bandit solutions with more current metrics and methodologies, such as counterfactuals. In this work, we propose and evaluate an integration between these two frameworks, demonstrating the potential and richness of analyzes that can be carried out from this combination.

1 Introduction

Several Web applications have invested in recommendation systems (RSs) to guide the entire user experience from their first interactions as a sequential decision model [15]. In this case, the system should recommend one or more items upon each user interaction, receive the user's feedback, and update its knowledge for the next recommendation [12]. The idea is to learn with each interaction to increase the knowledge of the system and maximize user satisfaction in the long run. Current works have addressed this challenge as a Multi-Armed Bandit (MAB) problem, where items are modeled as arms to be selected, and the user experience is represented by the accumulated reward [7].

Despite the recent advances, there needs to be more consensus on best practices for evaluating an interactive RS. Traditionally, evaluations of new algorithms are performed offline using a pre-selected dataset related to item recommendations and their respective ratings by users. This set is divided into training, used to train the recommenders, and testing for evaluation. The algorithms'

This work was partially funded by CNPq, CAPES, FINEP and Fapemig.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13956, pp. 635–647, 2023.
https://doi.org/10.1007/978-3-031-36805-9_41

performance is measured through metrics such as precision and recall. For this scenario, we highlight the *iRec* [8], a framework to evaluate interactive RSs providing a fair comparison between different RSs with several evaluation methodologies widely tested and used in the literature. However, offline evaluations suffer from data bias, as all recorded user evaluations were collected based on items recommended by a previously implemented policy, which may result in an inaccurate reflection of the users' real preferences [13].

RSs can also be evaluated by online methodologies, such as A/B [3] tests to avoid this bias. However, these methodologies require more time and effort to obtain results. To overcome these limitations, researchers have turned to counterfactual evaluation approaches [5]. In this case, from pre-existing data, as in offline evaluation, for which we know the recommendation policy used in the recommendations, we estimate what the performance of a new strategy would be like if it was used to replace the source policy (estimators). Thus, counterfactual evaluation simulates a hypothetical situation where the user received a different recommendation and compares the probability of taking the desired action in these two situations, similar to an offline A/B test [5]. In this scenario, we have the Open Bandit Pipeline (OBP), a framework that provides a complete and standardized experimental evaluation procedure using a counterfactual methodology. Since most public data collections do not present the used recommendation policy, the OBP also includes modules for creating synthetic data.

It is clear that, in order to perform comprehensive evaluations of interactive RSs, it is necessary to take into account both the traditional methodologies provided by the *iRec* and the counterfactual ones offered by the OBP, supporting different types of analysis allowing the precise and objective comparison of results. Thus, in this paper, we propose an integration between the *iRec* [8] and the OBP [5], resulting in a tool that performs both offline assessments (with the main metrics) and online assessments (with different estimators)[1]. We evaluated our proposal through an experiment considering four collections (three traditional and one synthetic), five different RSs, three traditional evaluation metrics, and three counterfactual estimators. Through this experiment, we were able to evaluate not only the algorithms' performance but also the impact of external factors such as user context and the recommender's interaction with the user over time.

The remainder of the paper is organized as follows. Section 2 presents the theoretical framework of the work. The tools proposed and developed in this work are presented in Sect. 3. In Sect. 4, we present the experiments and discuss their results. Finally, the conclusions and future work are presented in Sect. 5.

2 Theoretical Framework

Collaborative filtering (CF), MAB, and counterfactual evaluation approaches are machine learning techniques used in various contexts. Succinctly defined,

[1] Available at https://github.com/YanAndrade61/iRec-OBP.

CF is an approach that uses the similarity of preferences, tastes, and choices among users over time; MAB is an optimization technique that seeks to find the best trade-off between exploration and exploitation in a set of options; and the counterfactuals address what the model would predict if changed the action input. They enable to recognize and evaluate a machine learning model in terms of how it reacts to input (feature) changes. This section briefly introduces each of these techniques.

2.1 Collaborative Filtering

CF model-based approaches have been achieving the best results in the recommendation. In this context, users and items are represented by multiple aspects. Each user $u \in U$ has a probability of being interested in each aspect $z \in Z$. Therefore, users with similar preferences participate in the same interest groups. Similarly, each item $i \in I$ will likely attract users in each aspect $z \in Z$. Thus, model-based approaches define the probability of interest of u over i by the combination of these interests:

$$P(i|u) = \sum_{z \in Z} P(i|z) \cdot P(z|u) \tag{1}$$

The term $P(i|z)$ does not depend on the target user and represents how relevant an item i is to group z. The term $P(z|u)$ is the user's personalization, also known as the user model $\theta_u = \theta_{uz} : \theta_{uz} = P(z|u) \wedge z \in Z$. In recent years, both have been addressed by latent factor models [1]. These methods factorize the classification matrix $M^{m \times n}$ into the product of two low-rank matrices $P \in \mathbb{R}^{m \times z}$ and $Q \in \mathbb{R}^{z \times n}$. While the matrix $P^{m \times z}$ contains the user model θ_u, representing the multiple interests of each user u in groups z, the matrix $Q^{z \times n}$ represents the relevance of item i to groups z. In this approach, the recommendation is redefined as the association of user and item factors: $s(i, u) = p_u^\top \cdot q_i$

2.2 Multi-armed Bandits

Although effective, CF approaches are inefficient and must be retrained at each system update. They cannot handle online scenarios, where the model has to learn from the data provided in sequential order and update itself at each iteration [9,14]. Recent advances in RSs have been related to the theory of reinforcement learning, modeling the online learning problem as a MAB problem to handle the recommendation task in this context. The MAB is a sequential decision model that continuously chooses an action a from a set of actions $\mathcal{A}$ – *a.k.a.* arms. Selecting an action $a \in \mathcal{A}$ at a point t in time results in a particular reward $R(a_t) \in \mathbb{R}$, which can be summarized as an actual number. Although the reward is not known until the arm is selected, the main goal in the bandit model is to maximize the expected rewards returned $\sum R(a_t)$.

In case the system has sufficient knowledge about the domain, the best option is to select the action that provides the maximum possible reward at all

times [6,14]. However, this knowledge is uncertain and often unknown. Therefore, the MAB model always has to decide between two options. The first, more conservative option is to select the arms with the highest past rewards – an exploitation approach. In contrast, another option is investing in different arms to gain more information about the domain and make even better future decisions – an exploration approach. Such options characterize the dilemma of exploitation-exploration (i.e., exp-exp) and require that the model be able to exploit the maximum available knowledge while also exploring the solution space to acquire even more knowledge about the domain [14]. Thus, different MAB methods have been proposed in the literature to consider the dilemma exp-exp [9]. They generally include the traditional MAB methods, such as ϵ-*Greedy* [14], Upper Confidence Bounds (UCB) [14] and Thompson Sampling (TS) [2].

2.3 CF-Based Multi-armed Bandits

In MAB representations for RSs, the items to be recommended are modeled as the arms. Selecting an arm is equivalent to recommending an item, and the reward is the feedback from the user (e.g., satisfaction) [6]. The feedback is saved continuously in a $\mathcal{H}(t)$ set. An item is recommended according to a prediction rule π, defined as a function that exploits current information about the user: $i(t) \equiv \pi(\mathcal{H}(t))$. The optimal strategy should maximize rewards on T interactions:

$$i^*(\cdot) = \underset{i(\cdot)}{} \sum_{t=1}^{T} \mathbb{E}\left[r_{u,i(t)}|t\right] \tag{2}$$

The prediction rule π is based on combining the CF classes and the MAB models from the literature [10,14]. As in traditional CF scenarios, model-based approaches achieve better recommendations and propose a probabilistic matrix formulation via PMF (Probabilistic Matrix Factorization), modeling the distribution of rewards by the latent factors of users and items similar to model-based [10,11,14] methods. The expected reward is modeled as the product of the user's latent factors $\boldsymbol{p_u}$ with the item factors $\boldsymbol{q_i}$. Such modeling reformulates the objective function as follows:

$$i^*(\cdot) = \underset{i(\cdot)}{} \sum_{t=1}^{T} \mathbb{E}\left[r_{u,i(t)}|t\right] = \underset{i(\cdot)}{} \sum_{t=1}^{T} \mathbb{E}\left[\boldsymbol{p_u}^\top \boldsymbol{q_{i(t)}}|t\right] \tag{3}$$

Current efforts focus on optimizing this objective function, balancing the dilemma of *exp-exp*. Traditional MAB approaches, such as ϵ-*Greedy*, UCB and TS, are being adapted to consider this objective function as a prediction rule [2,10,14]. Table 1 summarizes the main features of each MAB approach. The difference between the algorithms is in how they control the exp-exp dilemma. The *silon-Greedy* exploits the prediction rule with probability $(1 - \epsilon)$ and the UCB and TS first measure an uncertainty Σ around the available information about users and items.

Table 1. Models used as a basis by CF-based MAB algorithms

	ϵ-Greedy	UCB	Thompson Sampling
Prediction Rule	- Estimate $p_{u,t}$ based on $\mathcal{H}(t)$ - With probability (1 - ϵ): – $i^*(t) = \underset{i \in I}{} (p_{u,t}^\top \cdot q_i)$ - Otherwise: – randomly $i^*(t)$ select - Get the reward $r_{u,i}$ - Update $\mathcal{H}(t)$ based on $r_{u,i}$	- Estimate $p_{u,t}$ based on $\mathcal{H}(t)$ - Estimate $\Sigma_{u,i}$ com $\mathcal{H}(t)$ e $\{q_i : i \in I\}$ - Select the item: – $i^*(t) = \underset{i \in I}{} (p_{u,i}^\top q_i + \Sigma_{u,i})$ - Get the reward $r_{u,i}$ - Update $\mathcal{H}(t)$ based on $r_{u,i}$	- Estimate $\mu_{u,t}$ based on $\mathcal{H}(t)$ - Estimate $\Sigma_{u,t}$ based on $\mathcal{H}(t)$ - Sample $\tilde{p}_u$ of $\mathcal{N}(p_{u,t}\|\mu_{u,t}, \Sigma_{u,t})$ - Select the item: – $i^*(t) = \underset{i \in I}{} (\tilde{p}_{u,t}^\top \cdot \tilde{q}_i)$ - Get the reward $r_{u,i}$ - Update $\mathcal{H}(t)$ based on $r_{u,i}$

2.4 Counterfactual

Although several approaches exist to solve the problem, there still needs to be a consensus on the best way to evaluate these algorithms. Traditional metrics have served as a basis for comparing the efficiency of each one, but recent studies have shown how selection bias and exposure bias [4], existing in collections, impact the evaluation result. Both occur when the data used to train the system needs to be representative of the population as a whole, resulting in an inaccurate reflection of user preferences that leads to a biased selection of the exposed items. That happens in offline datasets because all recorded user evaluations were collected based on items recommended by a previously implemented policy (i.e., the model in production at the time of user interaction). Counterfactual estimators allow using existing log data to estimate how some new target recommendation policy (i.e., a new approach) would have performed if it had been used instead of the policy that recorded the data. It allows for an Off-Policy Evaluation (OPE) like an unbiased offline A/B test. There are several counterfactual estimators proposed in the literature [3]. Direct method (DM) uses a model to complete missing rewards and uses the rewards of the selected items. Inverse propensity score (IPS) assigns importance weights to the current policy based on the original recommender values used to generate the dataset collection. Clipped IPS (CIPS) adapts the original IPS, limiting large weights with a λ parameter. Self-Normalized IPS (SNIPS) rescales the original IPS value by summing all importance weights. Doubly robust (DR) combines DM with IPS to reduce variance and work well with small samples. OBP has all these estimators implemented internally.

3 Tools for MAB Models

The tool proposed in this work aims to integrate two classical frameworks: iRec and OBP, obtaining a complete environment for analysis of interactive RSs, allowing from data processing to experiments execution and comprehensive evaluation, encompassing both offline and online aspects.

3.1 iRec

iRec[2] is a framework proposed to enable the use of interactive models, in particular those based on MAB models, in the recommendation domain. As we can see in Fig. 1, the iRec is composed of three main components that cover the whole experimentation process: (1) the construction of an Environment; (2) the definition of a Recommendation Agent; and (3) the definition of an Experimental Evaluation. In the Environment component, we configure the entire data structure to be processed by the framework. In it, we load the desired databases and define all the data preparation modules to be applied to them, such as the pre-filtering and splitting strategies for the training and test sets. The iRec has 17 public datasets for various application scenarios, such as movies, music, points of interest, products, and clothing. In turn, the Recommendation Agent component selects the model that will be used to define the best item(s) for each user in each iteration. In other words, this component implements the RS used in the recommendation. Finally, the component Experimental Evaluation integrates the models proposed in the Recommendation Agent over the data specified in the first component - Environment. First, we define the interaction between these two components through an Evaluation Policy module. In it, we configure an environment in which an item (or a set of items) is recommended in each algorithm iteration (one trial). For a predetermined time (number of tries), the Recommendation Agent performs the recommendations within the Environment, receiving a positive or negative reward and updating its knowledge. All logs (i.e., records of the actions performed by the Recommendation Agent and rewards provided by the users, according to the data in the test suite) are stored. Finally, the Experimental Evaluation analyzes these logs, applies the recommendation evaluation metrics, and performs the necessary statistical tests to perform a proper evaluation of the performance of the RSs implemented in the Recommendation Agent.

3.2 Open Bandit Pipeline - OBP

OBP [5] is a library that includes some modules to implement preprocessing of data collections and various recommendation methods. Unlike iRec, this library attempts to deal with the bias of existing databases for the recommendation scenario. Currently, the datasets used during the recommendation model's training phase are logs from a given recommender containing information about user interactions and items. However, the recommendation process changes how users interact with the system through clicks, ratings, etc. Thus, using this user interaction data, we are ignoring the intervening nature of recommendations. As a result, we are not evaluating whether users would click or buy more due to our new recommendations but rather to what extent the new recommendations fit the recorded data. It is precisely this problem that OBP tries to minimize through a Counterfactual evaluation policy. We present a library overview in

[2] Available at https://github.com/irec-org.

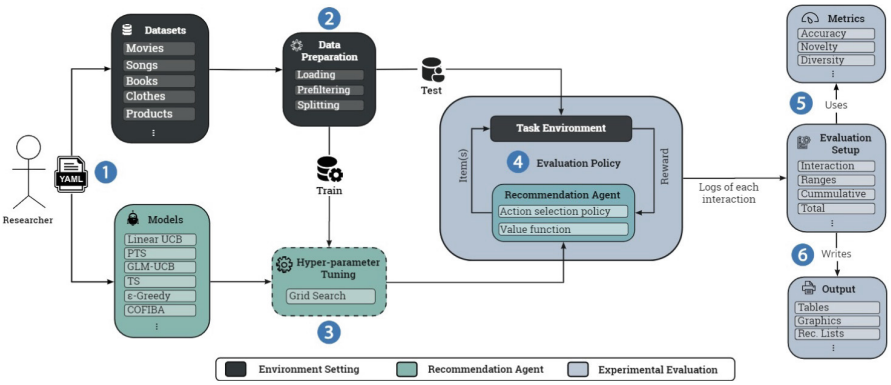

Fig. 1. An overview of the iRec framework [8]

Fig. 2. We can see the whole structure and its main modules. In short, we divided the library into four main modules: (i) the data module, which provides mechanisms to work on data collections, from the data loading step, and basic preprocessing steps to methods to generate synthetic data, among others; (ii) the policy module that provides interfaces for implementing new bandits methods and new evaluation policies, besides already having several relevant evaluation models and policies from the literature, both for the online and offline scenario; (iii) the simulation module that provides functions to perform offline simulation of bandits models. Through it, it is possible to compare and evaluate the performance of MAB algorithms; and (iv) the evaluation policy module that has generic abstract interfaces, ideal for custom implementations, in which users can add new estimators, besides having classical and advanced estimators already implemented (Sect. 2.2).

3.3 Integration Tool

As previously mentioned, there is no consensus on evaluating interactive recommendation models. Applying frameworks such as iRec and OBP is essential for this scenario. Even though they are already beneficial individually, there is more potential in combining them. iRec offers implementations of the main recommenders known in the literature, with modules for offline simulation and evaluation using traditional metrics. In turn, OBP provides an environment for evaluation using the main counterfactual estimators by generating synthetic data with a base recommender and calculating the probabilities of each item in the interactions. Thus, it is possible to simulate the synthetic data generated by OBP on iRec recommenders and evaluate performance online (using OBP) and offline (using iRec))[3].

In this sense, we propose a tool that integrates the two frameworks iRec and OBP (Fig. 3), to perform a complete analysis of recommenders implemented as a

[3] Available at https://github.com/YanAndrade61/iRec-OBP.

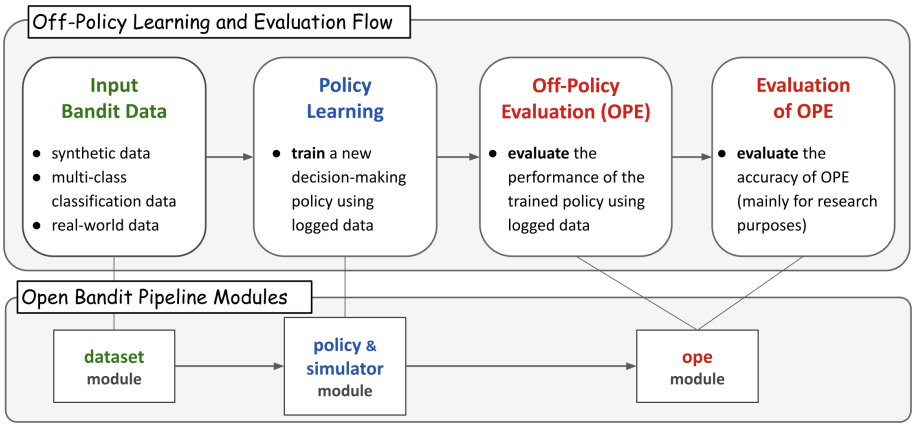

Fig. 2. OBP library overview [5]

MAB model. This tool provides a standardized methodology for the implementation of data preprocessing, the selection of learning methods, and a complete evaluation employing traditional metrics and counterfactual estimators (Off-Policy Evaluation - OPE). The unification process between the frameworks is separated into three main modules: (1) the data construction (OBP); (2) the simulation of the recommenders (iRec); (3) and the experimental evaluation (OBP and iRec). In addition, our tool also aims at maintaining the practicality of new implementations existing in each of the tools.

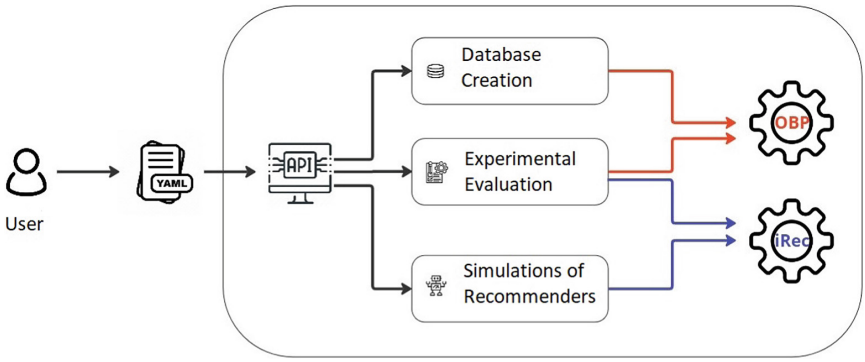

Fig. 3. Integration API between iRec and OBP

The first module uses OBP and consists of preparing the data to run the experiments, including creating, organizing, and parameterizing the synthetic data in a format suitable for analysis. We included two new parameters to the dataset construction step beyond those already provided by OBP to improve the

flexibility and adaptability of the tool. The first parameter allows it to indicate the number of distinct users that will be simulated in the analysis, allowing the creation of datasets with different sizes. The second parameter allows it to define the context vector of each user and additional information used by the models to make more precise recommendations. With the addition of these parameters, it is possible to simulate more complex and realistic scenarios in evaluating recommenders in MAB models. The separation of data into training and testing is carried out using iRec, with different possibilities.

In the second module, the prepared data is used to train and test the recommender models implemented by iRec, which offers several recommendation algorithms, from classical approaches to more recent algorithms based on deep learning. The highly customizable framework offers an intuitive API that facilitates integration with other software. By using iRec in this second module, it is possible to evaluate different recommender models based on MAB.

Finally, the third module evaluates the implemented recommendation algorithms based on the logs generated in the previous module. To this end, we used MAB performance evaluation techniques, divided into two groups: (1) the traditional metrics, such as precision and recall, performed by iRec; and (2) the counterfactual estimators, such as DM, IPS, and DR, performed by OBP.

The configurations for using the tool are composed of several *yaml* files responsible for defining all the module parameters. *Yaml* is a human-readable data serialization language commonly used for configuration files. This way, a researcher can define the dataset parameters, the policy used in its creation, the simulation's recommenders, all the offline and online evaluation metrics, and more. Configuration 1 depicts an example of its use.

syntheticData.yaml

```
SyntheticData:
n_actions: 1000
dim_context: 23
reward_type: continuos
reward_function: Linear
min_reward: 1
max_reward: 5
n_users: 943
users_context_file: path
random_state: 12345
obp_parameters: None
splitting:
  strategy: temporal
  train_size: 0.8
  test_consumes: 5
```

experimentalSetup.yaml

```
ExperimentalSetup:
  irec_parameters_file: path
  input_file: path
  output_file: path
```

evaluationSetup.yaml

```
EvaluationSetup:
  irec_parameters_file: "path"
  ope_estimators: [IPW,DM,DR]
  regression_model:
```

Configuration 1. Synthetic Data, Experimental Setup and Evaluation Setup

We defined the specifications for the generated data in Configuration 1 (Synthetic Data). It is possible to specify the number of actions (n_actions), the dimension of the user context (dim_context), the type of generated reward (reward_type), the policy that will be implemented (reward_function), and the limits of the generated returns, in case it is continuous. In addition, it is possible to define the number of users (n_users) and provide a file with the contexts of each user (user_context_file) or let the system generate the contexts. It is also possible to define a randomness index (random_state) and include extra parameters available through OBP (obp_parameters). Finally, it is possible to select the data splitting strategy.

Configuration 2 (Experimental Setup) follows the pattern used by iRec and its configuration file of the recommendation agent module (irec_parameters_file). It adds two parameters for the path of the input and output files that will be used. In this sense, Configuration 3 (Evaluation Setup) maintains the format used by iRec in the experimental evaluation module to define the metrics that will be used (irec_parameters_file), along with the counterfactual estimators for analysis (ope_estimators) and the regression model (regression_model), if the DM or DR estimators are used.

4 Experiments

This section presents an experimental evaluation demonstrating the potential and richness of analyses that can be performed when we combine the two main evaluation frameworks for interactive recommenders. While iRec provides a huge amount of MAB models, offline evaluation metrics, and data collections – in addition to a whole framework of statistical comparisons, OBP offers the possibility of creating synthetic and unbiased data collections – in addition from estimators to counterfactual assessments. Using our integration tool of these two frameworks, we divided our experiments between the traditional and the counterfactual evaluation, comparing the obtained results.

4.1 Traditional Evaluation

In our experiments, we first selected three datasets made available by iRec: Netflix (movies), Good Books (books), and Yahoo Music (music). We compared the performance of the following algorithms, previously described in Sect. 2.1 and made available in iRec: e-Greedy, UCB, TS, Linear e-Greedy, and Linear UCB, considering three metrics implemented by iRec: Hits: the number of recommendations that matched the user's history (relevance); ILD: measured by the Pearson correlation of the item attribute vectors among the recommended items list (diversity); and Users Coverage: represented by the percentage of distinct users who have a relationship with the recommended items (coverage).

Table 2. Performance of bandit models in the offline scenario. Results were compared using the Wilcoxon test with a p-value of 0.05. The symbol ▲ indicates statistical gains, and the symbol • represents ties

Dataset	Yahoo Music			Netflix			Good Books		
Metric	Hits			Hits			Hits		
T	10	50	100	10	50	100	10	50	100
e-Greedy	1.460	7.424	13.360	1.320	5.390	9.936	0.800	3.072	5.633
UCB	1.358	7.330	13.277	1.284	5.440	9.915	0.764	2.984	5.493
TS	1.907	8.356	14.720	1.882	7.498	12.959	1.361	4.528	7.216
Linear e-Greedy	0.011	0.316	1.059	0.158	2.303	6.037	0.060	0.815	2.543
Linear UCB	**3.157**▲	**15.514**▲	**25.361**▲	**1.980**▲	**12.076**▲	**22.361**▲	**1.586**▲	**6.848**▲	**12.593**▲
Metric	ILD			ILD			ILD		
T	10	50	100	10	50	100	10	50	100
e-Greedy	**0.461**•	0.462	0.465	0.401	0.416	0.421	0.489	0.494	0.495
UCB	**0.465**•	0.463	0.465	0.404	0.416	0.421	**0.490**▲	**0.495**▲	**0.495**▲
TS	0.431	0.452	0.459	0.336	0.373	0.387	0.467	0.487	0.492
Linear e-Greedy	**0.466**▲	**0.478**▲	**0.488**▲	**0.481**▲	**0.482**▲	**0.481**▲	0.487	0.488	0.488
Linear UCB	0.387	0.418	0.436	0.375	0.387	0.394	0.428	0.469	0.476
Metric	UsersCoverage			UsersCoverage			UsersCoverage		
T	10	50	100	10	50	100	10	50	100
e-Greedy	0.684	0.960	0.985	0.545	0.857	0.954	0.499	0.853	0.932
UCB	0.664	0.960	0.986	0.544	0.872	0.949	0.483	0.842	0.930
TS	0.748	0.960	0.986	**0.631**▲	0.895	0.957	**0.636**▲	**0.893**▲	**0.944**▲
Linear e-Greedy	0.005	0.038	0.075	0.078	0.129	0.244	0.037	0.181	0.306
Linear UCB	**0.806**▲	**0.967**▲	**0.990**▲	0.588	**0.920**▲	**0.963**▲	0.489	0.819	0.915

Table 2, automatically generated by iRec, shows the experimental results generated with the configurations described earlier. It is important to note that none obtained the best results for all evaluated metrics, demonstrating the importance of using different evaluation metrics appropriate to the recommendation scenario. RSs should be able to provide relevant, diverse, and novel items, meeting the consumption needs of most users. Regarding relevance (hits), the superiority of Linear UCB is clear, regardless of the recommended list size. Considering diversity, Linear e-Greedy presents the best results for the Yahoo Music and Netflix collections, while UCB was superior for the Good Books collection. However, contrary to the perspective of relevance, the superiority of these algorithms is not so disparate from the perspective of diversity, reinforcing the need for statistical tests like the one used by iRec (i.e., Wilcoxon). From the coverage perspective, we again observe the superiority of Linear UCB. However, in this case, we have TS with a performance very close to Linear UCB, even reaching statistical superiority in the Good Books collection.

4.2 Counterfactual Evaluation

Counterfactual evaluation allows an RSs evaluation without the bias of selection and exposure. Based on pre-existing collections, it estimates how the performance of a new strategy would be if it were used in place of the original policy through estimators. Current counterfactual estimators require knowing the rec-

ommendation policy used in production to create the dataset, which is unavailable for offline datasets. In this sense, we created a synthetic recommendation dataset using OBP to produce ratings from 1 to 5, following the same pattern as the traditional MovieLens 100k (100 thousand ratings for the same number of users). Each context contains the user ID and its characteristics (e.g., gender, age, and occupation). A linear policy makes recommendations. In this case, we evaluated the same RSs from the previous experiment, which was only possible after integrating iRec with OBP. We configured them to make 100 recommendations for each user. Three distinct estimators were used to evaluate each RS, provided by OBP and described in Sect. 2.2 (i.e., IPS, DM, and DR).

Table 3 shows the results. The first and second columns refer to the estimated mean value of the policy and the confidence interval made by the selected metrics for each recommender. The original policy (linear) used to create the dataset influenced the DM, and IPS has a higher variance than the other methods. DR is the fairest and most consistent value. The third column shows the relative value of the policy – how much this new policy improved the original policy. Values greater than one mean that the policy would perform better if it replaced the original one. When evaluating the results, we observed that the Linear UCB and UCB strategies perform best, reinforcing the results obtained in the traditional evaluation. However, TS highlighted previously had a lower-than-expected performance in the counterfactual evaluation, being unable, in some cases, to even outperform the original recommender (linear). This fact demonstrates how TS, to some extent, benefits from the pre-existing bias in pre-existing datasets.

Table 3. Counterfactual estimators for bandit models on synthetic data. DR is the most unbiased and consistent estimator to ensure reliable evaluation. The best results are in bold text.

Dataset	Synthetic Dataset								
Measure	Estimated policy value			95.0% CI (lower) – 95.0% CI (upper)			Relative policy value		
Estimators	IPS	DM	DR	IPS	DM	DR	IPS	DM	DR
e-Greedy	6.244	4.468	4.495	2.854–10.154	4.458–4.477	4.432–4.596	1.901	1.36	1.368
UCB	8.318	4.484	4.438	3.821–13.279	4.474–4.492	4.397–4.473	**2.532**	1.365	1.351
TS	3.248	4.303	4.312	1.511–5.274	4.294–4.312	4.273–4.367	0.988	1.31	1.312
Linear e-Greedy	2.874	3.973	3.916	1.086–5.133	3.945–4.0	3.828–3.981	0.875	1.209	1.192
Linear UCB	3.35	4.893	4.89	1.192–5.952	4.89–4.895	4.864–4.918	1.02	**1.489**	**1.488**

5 Conclusion and Future Works

This work filled a gap in the literature regarding the evaluation of interactive RSs based on MAB approaches by integrating two of the most important evaluation frameworks: iRec and OBP. The first one is focused on offline evaluations, providing a complete set of collections, metrics, and RSs. The second one offers methodologies and metrics focused on counterfactual evaluations. Our integration allows any bandit solution to be evaluated offline and from a counterfactual

perspective, including synthetic collections free from the bias of pre-selected collections. We evaluated the integration by considering four collections (three selected and one synthetic), six evaluation metrics (three offline and three counterfactual), and four state-of-the-art MAB algorithms. Our first observation is that evaluating the results regarding only one metric in the short, medium, or long term is not enough to assert that one model is better than another. Another important observation in our results concerns how much some algorithms "benefit" from the data bias, leading to erroneous conclusions about their performance. In these cases, the effectiveness of a model is more associated with how well it can approach the recommendations made by the recommender that originated the collection than effectively satisfying the user. We intend to conduct an even broader evaluation in future work, considering many other MAB models, metrics, and data collections.

References

1. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowl.-Based Syst. **46**, 109–132 (2013)
2. Chapelle, O., Li, L.: An empirical evaluation of thompson sampling. In: Advances in Neural Information Processing Systems, pp. 2249–2257 (2011)
3. Liu, Y., Yen, J.N., Yuan, B., Shi, R., Yan, P., Lin, C.J.: Practical counterfactual policy learning for top-k recommendations. In: ACM SIGKDD, pp. 1141–1151 (2022)
4. Pan, W., Cui, S., Wen, H., Chen, K., Zhang, C., Wang, F.: Correcting the user feedback-loop bias for recommendation systems. arXiv preprint arXiv:2109.06037 (2021)
5. Saito, Y., Aihara, S., Matsutani, M., Narita, Y.: Open bandit dataset and pipeline: towards realistic and reproducible off-policy evaluation. arXiv preprint arXiv:2008.07146 (2020)
6. Sanz-Cruzado, J., Castells, P., López, E.: A simple multi-armed nearest-neighbor bandit for interactive recommendation. In: RecSys, pp. 358–362 (2019)
7. Shams, S., Anderson, D., Leith, D.: Cluster-based bandits: fast cold-start for recommender system new users (2021)
8. Silva, T., Silva, N., Werneck, H., Mito, C., Pereira, A.C., Rocha, L.: irec: an interactive recommendation framework. In: SIGIR, pp. 3165–3175 (2022)
9. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (2018)
10. Wang, H., Wu, Q., Wang, H.: Factorization bandits for interactive recommendation. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
11. Wang, Q., et al.: Online interactive collaborative filtering using multi-armed bandit with dependent arms. IEEE Trans. Knowl. Data Eng. **31**(8), 1569–1580 (2018)
12. Wu, Q., Iyer, N., Wang, H.: Learning contextual bandits in a non-stationary environment. In: SIGIR, pp. 495–504 (2018)
13. Yang, Y., Xia, X., Lo, D., Grundy, J.: A survey on deep learning for software engineering. ACM Comput. Surv. (CSUR) **54**(10s), 1–73 (2022)
14. Zhao, X., Zhang, W., Wang, J.: Interactive collaborative filtering. In: Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, pp. 1411–1420 (2013)
15. Zhou, S., et al.: Interactive recommender system via knowledge graph-enhanced reinforcement learning. In: SIGIR, pp. 179–188 (2020)