**ELLIO T: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation**

Vito Walter Anelli*  
vitowalter.anelli@poliba.it  
Politecnico di Bari, Italy

Alejandro Bellogín  
alejandro.bellogin@uam.es  
Autónoma Madrid, Spain

Antonio Ferrara  
antonio.ferrara@poliba.it  
Politecnico di Bari, Italy

Daniele Malitesta  
daniele.malitesta@poliba.it  
Politecnico di Bari, Italy

Felice Antonio Merra  
felice.merra@poliba.it  
Politecnico di Bari, Italy

Claudio Pomo*  
claudio.pomo@poliba.it  
Politecnico di Bari, Italy

Francesco Maria Donini  
donini@unitus.it  
Università della Tuscia, Italy

Tommaso Di Noia  
tommaso.dinoia@poliba.it  
Politecnico di Bari, Italy

**ABSTRACT**

Recommender Systems have shown to be an effective way to alleviate the over-choice problem and provide accurate and tailored recommendations. However, the impressive number of proposed recommendation algorithms, splitting strategies, evaluation protocols, metrics, and tasks, has made rigorous experimental evaluation particularly challenging. Puzzled and frustrated by the continuous recreation of appropriate evaluation benchmarks, experimental pipelines, hyperparameter optimization, and evaluation procedures, we have developed an exhaustive framework to address such needs. ELLIO T is a comprehensive recommendation framework that aims to run and reproduce an entire experimental pipeline by processing a simple configuration file. The framework loads, filters, and splits the data considering a vast set of strategies (13 splitting methods and 8 filtering approaches, from temporal training-test splitting to nested K-folds Cross-Validation). ELLIO T optimizes hyperparameters (51 strategies) for several recommendation algorithms (50), selects the best models, compares them with the baselines providing intra-model statistics, computes metrics (36) spanning from accuracy to beyond-accuracy, bias, and fairness, and conducts statistical analysis (Wilcoxon and Paired t-test).

**CCS CONCEPTS**

• Information systems → Recommender systems; Collaborative filtering; • Computing methodologies → Learning from implicit feedback; Neural networks; Factorization methods.

**KEYWORDS**

Recommender Systems; Evaluation; Reproducibility; Bias; Fairness

**ACM Reference Format:**

Anelli et al. 2021. ELLIO T: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation. In Proceedings of the 44th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR ’21), July 11–15, 2021, Virtual Event, Canada. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3463245

**1 INTRODUCTION**

In the last decade, Recommendation Systems (RSs) have gained momentum as the pivotal choice for personalized decision-support systems. Recommendation is essentially a retrieval task where a catalog of items is ranked and the top-scoring items are presented to the user [58]. Once it was demonstrated their ability to provide personalized items to clients, both Academia and Industry devoted their attention to RSs [4, 13, 66, 67]. This collective effort resulted in an impressive number of recommendation algorithms, ranging from memory-based [85] to latent factor-based [21, 29, 55, 76], and deep learning-based methods [63, 97]. At the same time, the RS research community became conscious that accuracy was not sufficient to guarantee user satisfaction [71]. Novelty and diversity [17, 46, 93] came into play as new dimensions to be analyzed when comparing algorithms. However, this was only the first step in the direction of a more comprehensive evaluation of RSs. Indeed, more recently, the presence of biased [9, 107] and unfair [23, 25, 26] recommendations towards user groups and item categories has been widely investigated. In fact, RSs have been widely studied and applied in various domains and tasks, with different (and often contradicting in their hypotheses) splitting preprocessing strategies [16] fitting the specific scenario. Moreover, machine learning (and recently also deep learning) techniques are prominent in algorithmic research and require their hyperparameter optimization strategies and procedures [6, 92].

The abundance of possible choices generated much confusion about choosing the correct baselines, conducting the hyperparameter optimization and the experimental evaluation [81, 82], and reporting the details of the adopted procedure. Consequently, two major concerns arose: unreproducible evaluation and unfair comparisons [88]. On the one hand, the negative effect of unfair comparisons is that various proposed recommendation models have been compared with suboptimal baselines [22, 79]. On the other hand, in a recent study [22], it has been shown that only one-third of the published experimental results are, in fact, reproducible. Progressively, the RS community has welcomed the emergence of recommendation evaluation and even hyperparameter tuning frameworks [15, 24, 31, 88, 93]. However, facilitating reproducibility or extending the provided functionality would typically depend on developing bash scripts or programming on whatever language each framework is written.

This work introduces ELLIO T, a novel kind of recommendation framework, to overcome these obstacles. The framework analyzes
the recommendation problem from the researcher’s perspective. Indeed, ELLIOT conducts a whole experiment, from dataset loading to results gathering. The core idea is to feed the system with a simple and straightforward configuration file that drives the framework through the experimental setting choices. ELLIOT natively provides for widespread research evaluation features, like the analysis of multiple cut-offs and several RSs (50). According to the recommendation model, the framework allows, to date, the choice among 27 similarities, the definition of multiple neural architectures, and 51 hyperparameter tuning combined approaches, unleashing the full potential of the HyperOpt library [15]. To enable the evaluation for the diverse tasks and domains, ELLIOT supplies 36 metrics (including Accuracy, Error-based, Coverage, Novelty, Diversity, Bias, and Fairness metrics), 13 splitting strategies, and 8 prefiltering policies. The framework can also measure to what extent the RS results are significantly different from each other, providing the paired t-test and Wilcoxon statistical hypothesis tests. Finally, ELLIOT lets the researcher quickly build their models and include them in the experiment.

2 PRIOR WORK

Background. RS evaluation is an active, ever-growing research topic related to reproducibility, which is a cornerstone of the scientific process as identified by Konstan and Adomavicius [53]. Recently researchers have taken a closer look at this problem, in particular because depending on how well we evaluate and assess the efficacy of a system, the significance and impact of such results will increase.

Some researchers argue that to enhance reproducibility, and to facilitate fair comparisons between different works (either frameworks, research papers, or published artifacts), at least the following four stages must be identified within the evaluation protocol [81]: data splitting, item recommendations, candidate item generation, and performance measurement. In a recent work [11], these stages have been completed with dataset collection and statistical testing. Some of these stages can be further categorized, such as performance measurement, depending on the performance dimension to be analyzed (e.g., ranking vs error, accuracy vs diversity, and so on).

In fact, the importance and relevance of the aforementioned stages have been validated in recent works; however, even though most of the RS literature has been focused on the impact of the item recommendation stage as an isolated component, this is far from being the only driver that affects RS performance or the only component impacting on its potential for reproducibility. In particular, Meng et al. [72] survey recent works in the area and conclude that no standard splitting strategy exists, in terms of random vs temporal splits; furthermore, the authors found that the selection of the splitting strategy can have a strong impact on the results. Previously, Campos et al. [16] categorized and experimented with several variations of random and temporal splitting strategies, evidencing the same inconsistency in the results. Regarding the candidate item generation, it was first shown [10] that different strategies selecting the candidate items to be ranked by the recommendation algorithm may produce results that are orders of magnitude away from each other; this was later confirmed [81] in the context of benchmarking recommendation frameworks. Recent works [58, 62] evidenced that some of these strategies selecting the candidate items may introduce inconsistent measurements which should, hence, not be trusted.

Finally, depending on the recommendation task and main goal of the RS, several performance dimensions, sometimes contradicting, can be assessed. For a classical overview of these dimensions, we refer the reader to Gunawardana and Shani [32], where metrics accounting for prediction accuracy, coverage, confidence, trust, novelty, diversity, serendipity, and so on are defined and compared. However, to the best of our knowledge, there is no public implementation providing more than one or two of these dimensions. Moreover, recently the community has considered additional dimensions such as bias (in particular, popularity bias [1]) and fairness [26]. These dimensions are gaining attention, and several metrics addressing different subtleties are being proposed, but no clear winner or standard definition emerged so far – as a consequence, the community lacks an established implementation of these novel evaluation dimensions.

Related Frameworks. Reproducibility is the keystone of modern RSs research. Dacrema et al. [22] and Rendle et al. [78] have recently raised the need of comprehensive and fair recommender model evaluation. Their argument on the outperforming recommendation accuracy of latent-factor models over deep-neural ones, when an extensive hyper-parameter tuning was performed, made it essential the development of novel recommendation frameworks. Starting from 2011, Mymedialite [31], LensKit [24, 27], LightFM [59], RankSys [93], and Surprise [45], have formed the basic software for rapid prototyping and testing of recommendation models, thanks to an easy-to-use model execution and the implementation of standard accuracy, and beyond-accuracy, evaluation measures and splitting techniques. However, the outstanding success and the community interests in deep learning (DL) recommendation models, raised need for novel instruments. LibRec [33], Spotlight [60], and OpenRec [100] are the first open-source projects that made DL-based recommenders available – with less than a dozen of available models without filtering, splitting, and hyper-optimization tuning strategies. An important step towards more exhaustive and up-to-date set of model implementations have been released with RecQ [102], DeepRec [35], and Cornac [83] frameworks. However, they do not provide a general tool for extensive experiments on the pre-elaboration and the evaluation of a dataset. Indeed, after the reproducibility hype [22, 78], DaisyRec [88] and Recbole [105] raised the bar of framework capabilities, making available both large set of models, data filtering/splitting operations and, above all, hyper-parameter tuning features. However, we found a significant gap in splitting and filtering capabilities, in addition to a complete lack of two nowadays popular (even critical) aspects of recommendation performance: biases and fairness. Reviewing these related frameworks, emerged a striking lack of an open-source recommendation framework able to perform by design an extensive set of pre-elaboration operations, to support several hyperparameters optimization strategies and multiple sets of evaluation measures, which include bias and fairness ones, supported by statistical significance tests – a feature absent in other frameworks (as of February 2021). ELLIOT meets all these needs. Table 1 gives an overview of the frameworks and to which extent they satisfy the mentioned requirements.

3 ELLIOT

ELLIOiT is an extensible framework composed of eight functional modules, each of them responsible for a specific step of an experimental recommendation process. What happens under the hood
Figure 1 is transparent to the user, who is only expected to provide human-level experimental flow details using a customizable configuration file. Accordingly, Elliot builds the overall pipeline. The following sections deepen into the details of the eight Elliot’s modules and outline the preparation of a configuration file.

3.1 Data Preparation
The Data modules are responsible for handling and managing the experiment input, supporting various additional information, e.g., item features, visual embeddings, and images. After being loaded by the Loading module, the input data is taken over by Prefiltering and Splitting modules whose strategies are reported in Table 1.

3.1.1 Loading. RSS experiments could require different data sources such as user-item feedback or additional side information, e.g., the visual features of an item images. To fulfill these requirements, Elliot comes with different implementations of the Loading module. Additionally, the user can design computationally expensive prefiltering and splitting procedures that can be stored and loaded to save future computation. Data-driven extensions can handle additional data like visual features [19, 51], and semantic features extracted from knowledge graphs [7]. Once a side-information-aware Loading module is chosen, it filters out the items devoiding the required information to grant a fair comparison.

3.1.2 Prefiltering. After data loading, Elliot provides data filtering operations through two possible strategies. The first strategy implemented in the Prefiltering module is Filter-by-rating, which drops off a user-item interaction if the preference score is smaller than a given threshold. It can be (i) a Numerical value, e.g., 3.5, (ii) a Distributional detail, e.g., global rating average value, or (iii) a user-based distributional (User Dist.) value, e.g., user’s average rating value. The second prefiltering strategy, k-core, filters out users, items, or both, with less than k recorded interactions. The k-core strategy can proceed iteratively (Iterative k-core) on both users and items until the k-core filtering condition is met, i.e., all the users and items have at least k recorded interaction. Since reaching such condition might be intractable, Elliot allows specifying the maximum number of iterations (Iter-n-rounds). Finally, the Cold-Users filtering feature allows retaining cold-users only.

3.1.3 Splitting. If needed, the data is served to the Splitting module. In detail, Elliot provides (i) Temporal, (ii) Random, and (iii) Fix strategies. The Temporal strategy splits the user-item interactions based on the transaction timestamp, i.e., fixing the timestamp, finding the optimal one [8, 12], or adopting a hold-out (HO) mechanism. The Random strategy includes hold-out (HO), K-repeated hold-out (K-HO), and cross-validation (CV). Table 1 provides further configuration details. Finally, the Fix strategy exploits a precomputed splitting.

3.2 Recommendation Models
After data loading and pre-elaborations, Recommendation module (Figure 1) provides the functionalities to train (and restore) the Elliot recommendation models and the new ones integrated by users.

3.2.1 Implemented Models. Elliot integrates, to date, 50 recommendation models (see Table 1) partitioned into two sets. The first set includes 38 popular models implemented in at least two frameworks reviewed in this work (i.e., adopting a framework-wise popularity notion). Table 1 shows that Elliot is the framework covering the largest number of popular models, with 30 models out of 38, i.e., 79%. The second set comprises other well-known state-of-the-art recommendation models implemented in less than two frameworks, namely, BPRSLIM [73], ConvNCF [39], NPR [74], MultiDAE [63], and NAIS [41], graph-learning based, i.e., NGCF [96], and LightGCN [38], visual-based, i.e., VBPR [36], DeepStyle [65], DVBPR [51], ACF [19], and VNPR [74], adversarial-robust, i.e., APR [40] and AMR [89], generative adversarial network (GAN)-based, i.e., IRGAN [95] and CFGAN [18], content-aware, i.e., Attribute-I-\&U-KNN [31], VSM [5, 75], Wide & Deep [20], and KahFM [7] recommenders.

3.2.2 Hyper-parameter Tuning. Hyperparameter tuning is an ingredient of the recommendation model training that definitely influences its performance [78]. Elliot provides Grid Search, Simulated Annealing, Bayesian Optimization, and Random Search strategies. Furthermore, Elliot allows performing four traversing strategies across the search space defined in each recommendation model configuration. When the user details the possible hyperparameters (as a list) without specifying a search strategy, Elliot automatically performs an exhaustive Grid Search. Elliot may exploit the full potential of
Table 1: Overview of the Elliot and related frameworks functionalities.

### Data Elaboration and Model Optimization Strategies

| Prefiltering | Temporal | Random | Hyperparam. Tuning |
|--------------|----------|--------|-------------------|
| Filter-       |          |        |                   |
| by-rating     |          |        |                   |
| $k$-core      |          |        |                   |
|              |          |        |                   |
| LensKit Java [27] | 0         | ✓ ✓ ✓ ✓ | 10                |
| Mymedialite [31] | 0         | ✓ ✓ ✓ ✓ | 6                 |
| RankSys [93] | 0         | ✓ ✓ ✓ ✓ | 1                 |
| LibRec [33] | 0         | ✓ ✓ ✓ ✓ | 11                |
| Implicit [28] | ✓         | ✓ ✓ ✓ ✓ | 1                 |
| OpenRec [100] | 0         | ✓ ✓ ✓ ✓ | 1                 |
| RecQ [102] | 0         | ✓ ✓ ✓ ✓ | 3                 |
| DeepRec [35] | 0         | ✓ ✓ ✓ ✓ | 2                 |
| LensKit Python [24] | 0         | ✓ ✓ ✓ ✓ | 5                 |
| Surprise [43] | 0         | ✓ ✓ ✓ ✓ | 6                 |
| Corr [83] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 4                 |
| RecBole [103] | 0         | ✓ ✓ ✓ ✓ | 8                 |
| DaisyRec [88] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 4                 |
| Elliot        | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 13                |

### Recommender Models

#### Popular Models

| LensKit Java [27] | ✓         | ✓         | ✓         | ✓         | ✓         | ✓         | 0         |
| Mymedialite [31] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 6 18      |
| RankSys [93] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 8         |
| LibRec [33] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 30 55     |
| Implicit [28] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 0 4       |
| OpenRec [100] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 7 11      |
| RecQ [102] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 6 13      |
| DeepRec [35] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 5 20      |
| LensKit Python [24] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 6         |
| Surprise [43] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 1 7       |
| Corr [83] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 21 35     |
| RecBole [103] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 50 65     |
| DaisyRec [88] | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 0 14      |
| Elliot        | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | ✓ ✓ ✓ ✓ | 20 50     |

### Evaluation of Recommendation Performance

#### Metric Families

| LensKit Java [27] | Accuracy | Error | Coverage | Novelty | Diversity | Bias | Fairness | Tot. |
|-------------------|----------|-------|----------|---------|-----------|------|----------|------|
| Mymedialite [31] | 6 2      | 1     | 1        | 1       | 1         | 6    |          | 0    |
| RankSys [93] | 7 3 6 18 | 3     | 6 18     | 33      | 34        | 30   |          | 0    |
| LibRec [33] | 6 4      | 10    |          |         |           | 0    |          | 0    |
| Implicit [28] | 0         |       |          |         |           | 0    |          | 0    |
| OpenRec [100] | 4         | 4     |          |         |           | 0    |          | 0    |
| RecQ [102] | 6 2      | 8     |          |         |           | 0    |          | 0    |
| DeepRec [35] | 6 2      | 8     |          |         |           | 0    |          | 0    |
| LensKit Python [24] | 4 2      | 6     |          |         |           | 0    |          | 0    |
| Surprise [43] | 4         | 4     |          |         |           | 0    |          | 0    |
| Corr [83] | 8 3      | 11    |          |         |           | 0    |          | 0    |
| RecBole [105] | 8 3      | 11    |          |         |           | 0    |          | 0    |
| DaisyRec [88] | 8         | 8     |          |         |           | 0    |          | 0    |
| Elliot        | 11 3 2 3 | 10 4 | 36       |         |           | 2    |          | 0    |

#### Statistical Tests

| LensKit Java [27] | Paired t-test | Wilcoxon | Tot. |
|-------------------|---------------|----------|------|
| Mymedialite [31] |               |          |      |
| RankSys [93] |               |          |      |
| LibRec [33] |               |          |      |
| Implicit [28] |               |          |      |
| OpenRec [100] |               |          |      |
| RecQ [102] |               |          |      |
| DeepRec [35] |               |          |      |
| LensKit Python [24] |               |          |      |
| Surprise [43] |               |          |      |
| Corr [83] |               |          |      |
| RecBole [105] |               |          |      |
| DaisyRec [88] |               |          |      |
| Elliot        |               |          | 2    |
the HyperOpt [15] library by considering all its sampling strategies. Table 1 summarizes the available Search Strategies and Search Spaces.

3.3 Performance Evaluation

After the training phase, ELLIOT continues its operations, evaluating recommendations. Figure 1 indicates this phase with two distinct evaluation modules: Metrics and Statistical Tests.

3.3.1 Metrics. ELLIOT provides 36 evaluation metrics (see Table 1), partitioned into seven families: Accuracy [86, 106], Error, Coverage, Novelty [94], Diversity [103], Bias [2, 3, 91, 101, 109], and Fairness [23, 108]. It is worth mentioning that ELLIOT is the framework that exposes both the largest number of metrics and the only one considering bias and fairness measures. Moreover, the user can choose any metric to drive the model selection and the tuning.

3.3.2 Statistical Tests. Table 1 shows that the reviewed related frameworks miss statistical hypothesis tests. This is probably due to the need to compute fine-grained (e.g., per-user or per-partition) results and retain them for each recommendation model. It implies that the framework should be designed for multi-recommender evaluation and handling the fine-grained results. ELLIOT brings the opportunity to compute two statistical hypothesis tests, i.e., Wilcoxon and Paired t-test, activating a flag in the configuration file.

3.4 Framework Outcomes

When the experiment finishes, it is time for ELLIOT to collect the results through the Output module in Figure 1. ELLIOT gives the possibility to store three classes of output reports: (i) Performance Tables, (ii) Model Weights, and (iii) Recommendation Lists. The former consist of spreadsheets (in a tab-separated-value format) with all the metric values computed on the test set for every recommendation model specified in the configuration file. The tables comprise cut-off specific and model-specific tables (i.e., considering each combination of the explored parameters). The user can also choose to store tables with the triple format, i.e., <Model, Metric, Value>. Tables also include cut-off-specific statistical hypothesis tests and a JSON file that summarizes the best model parameters. Optionally, ELLIOT saves model weights to avoid future re-training of the recommender. Finally, ELLIOT stores the top-k recommendation lists for each model adopting a tab-separated <User, Item, Predicted Score> triple-based format.

3.5 Preparation of the Experiment

The operation of ELLIOT is triggered by a single configuration file written in YAML. Configuration 1 shows a toy example of a configuration file. The first section details the data loading, filtering, and splitting information as defined in Section 3.1. The models section represents the recommendation models configuration, e.g., Item-KNN, described in Section 3.2.1. Here, the model-specific hyperparameter optimization strategies are specified (Section 3.2.2), e.g., the grid-search in Configuration 1. The evaluation section details the evaluation strategy with the desired metrics (Section 3.3), e.g., nDCG in the toy example. Finally, save_recs and top_k keys detail, for example, the Output module abilities described in Section 3.4. It is worth noticing that, to the best of our knowledge, ELLIOT is the only framework able to run an extensive set of reproducible experiments by merely preparing a single configuration file. Section 4 exemplifies two real experimental scenarios commenting on the salient parts of the configuration files.

4 EXPERIMENTAL SCENARIOS

We illustrate how to prepare, execute and evaluate a basic and a more advanced experimental scenario with ELLIOT.

4.1 Basic Configuration

Experiment. In the first scenario, the experiments require comparing a group of RSs whose parameters are optimized via a grid-search. Configuration 2 specifies the data loading information, i.e., semantic features source files, in addition to the filtering and splitting strategies. In particular, the latter supplies an entirely automated way of preprocessing the dataset, which is often a time-consuming and non-easily-reproducible phase. The simple_metrics field allows computing accuracy and beyond-accuracy metrics, with two top-k cut-off values (5 and 10) by merely inserting the list of desired measures, e.g., [Precision, nDCG, ...]. The knowledge-aware recommendation model, AttributeItemKNN, is compared against two baselines: Random and ItemKNN, along with a user-implemented model that is external MostPop. The configuration makes use of Elliot’s feature of conducting a grid search-based hyperparameter optimization strategy by merely passing a list of possible hyperparameter values, e.g., neighbors: [50, 100]. The reported models are selected according to nDCG@10.

Results. Table 2 displays a portion of experimental results generated by feeding ELLIOT with the configuration file. The table reports four metric values computed on recommendation lists at cutoffs 5 and 10 generated by the models selected after the hyperparameter tuning phase. For instance, Attribute-I-kNN model reports values for the configuration with neighbors set to 100 and similarity set to Braycurtis. Table 2 confirms some common findings: the item coverage value (JCov@10) of an Attribute-I-kNN model is higher than the one measured on I-KNN, and I-KNN is the most accurate model.

4.2 Advanced Configuration

Experiment. The second scenario depicts a more complex experimental setting. In Configuration 3, the user specifies an elaborate data splitting strategy, i.e., random_subsampling (for test splitting) and random_cross_validation (for model selection), by setting few splitting configuration fields. Configuration 3 does not provide a cut-off value, and thus a top-k field value of 50 is assumed as the cut-off. Moreover, the evaluation section includes the UserMADrating
Elliot is a framework that examines the recommendation process from an RS researcher’s perspective. It requires the user just to compile a flexible configuration file to conduct a rigorous and reproducible experimental evaluation. The framework provides several loading, prefiltering, splitting, hyperparameter optimization strategies, recommendation models, and statistical hypothesis tests. Elliot reports can be directly analyzed and inserted into research papers. We reviewed the RS evaluation literature, positioning Elliot among the existing frameworks, and highlighting its advantages and limitations. Next, we explored the framework architecture and how to build a working (and reproducible) experimental benchmark. To the best of our knowledge, Elliot is the first recommendation framework that provides a full multi-recommender experimental pipeline based on a simple configuration file. We plan to extend soon Elliot in various directions to include: sequential recommendation scenarios, adversarial attacks, reinforcement learning-based recommendation systems, differential privacy facilities, sampled evaluation, and distributed recommendation.

ACKNOWLEDGMENTS

The authors acknowledge partial support of the projects: Servizi Locali 2.0, PON ARS01_00876 Bio-D, PON ARS01_00821 FLET4.0, PON ARS01_00917 OK-INSAID, H2020 PASSPARTOUT, PID2019-108965GB-I00

Table 2: Experimental results for Configuration 2.

| Model        | nDCG@5 | ICov@5 | nDCG@10 | ICov@10 |
|--------------|--------|--------|---------|--------|
| Random       | 0.0098 | 3197   | 0.0056  | 3197   |
| MostPop      | 0.0699 | 68     | 0.0728  | 96     |
| I-KNN        | 0.0791 | 448    | 0.0837  | 710    |
| Attribute-I-KNN | 0.0464 | 1575   | 0.0485  | 2102   |

Table 3: Experimental results for Configuration 3.

| Model        | nDCG@5 | ARP@50 | ACLT@50 | UMADy@50 |
|--------------|--------|--------|---------|----------|
| BPRMF        | 0.2390 | 1096   | 0.0420  | 0.0516   |
| NeuMF        | 0.2585 | 919    | 0.8616  | 0.0032   |
| MultiVAE     | 0.2922 | 755    | 3.2871† | 0.1588†  |

†p-value ≤ 0.001 using Wilcoxon-test
[103] ChengXiang Zhai, William W. Cohen, and John D. Lafferty. 2003. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 – August 1, 2003, Toronto, Canada, Charles L. A. Clarke, Gordon V. Cormack, Jamie Callan, David Hawking, and Alan F. Smeaton (Eds.). ACM, 10–17.

[104] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. 2006. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. In Proceedings of the Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA, Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava (Eds.). SIAM, 549–553.

[105] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. CoRR abs/2011.01731 (2020).

[106] Guorui Zhou, Xiaoguang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018, Yike Guo and Faisal Farooq (Eds.). ACM, 1059–1068.

[107] Ziwei Zhu, Yun He, Xing Zhao, Yin Zhang, Jianling Wang, and James Caverlee. 2021. Popularity-Opportunity Bias in Collaborative Filtering. In Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining (WSDM ’21), March 8–12, 2021, Virtual Event, Israel. ACM.

[108] Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-Aware Tensor-Based Recommendation. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018, Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang (Eds.). ACM, 1153–1162.

[109] Ziwei Zhu, Jianling Wang, and James Caverlee. 2020. Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems. In Proceedings of the 33rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25–30, 2020, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 449–458.