IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 December 2024, accepted 3 February 2025, date of publication 11 February 2025, date of current version 18 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3541014

RESEARCH ARTICLE

A Comparative Evaluation of Recommender Systems Tools

AYOUB AKHADAM[1], OUMAYMA KBIBCHI[2], LOUBNA MEKOUAR[3], (Senior Member, IEEE), AND YOUSSEF IRAQI[3], (Senior Member, IEEE)

[1]TICLab, College of Engineering and Architecture, International University of Rabat, Rabat 11103, Morocco
[2]ENSIAS, Rabat 10000, Morocco
[3]School of Computer Science, Mohammed VI Polytechnic University, Ben Guerir 43150, Morocco

Corresponding author: Loubna Mekouar (Loubna.Mekouar@um6p.ma)

This work was supported by Mohammed VI Polytechnic University, Morocco, under the Start-Up Grant Award.

ABSTRACT Due to the vast flow of information on the Internet, easy and effective access to information has become crucial. Recommender systems are important in information filtering, as they significantly impact large-scale internet web services such as YouTube, Netflix, and Amazon. As the demand for personalized recommendations continues to grow, researchers and practitioners alike strive to develop tools specifically designed for this purpose to meet the increasing need. In this work, we address the challenges associated with selecting software frameworks and Machine Learning (ML) algorithms for Recommender Systems (RSs), thus, we offer a detailed comparison of 42 open-source RS software to provide insights into their different features and capabilities. Furthermore, the paper presents a concise overview of various ML algorithms to generate recommendations, reviews the most used performance metrics to evaluate RS, and then compares several ML algorithms provided by four popular recommendation tools: Microsoft Recommenders, Lenskit, Turi Create, and Cornac.

INDEX TERMS Recommender systems, recommender system tools, machine learning, collaborative filtering, Microsoft recommenders, Turi create, Lenskit, Cornac.

I. INTRODUCTION

The advent of major online platforms such as YouTube, Amazon, and Netflix has propelled Recommender Systems (RSs) to the forefront of contemporary digital industries, transforming them from optional features into essential user experience components. Due to the vast amount of online information, users may face many possibilities when selecting the right items. RSs are designed to help users find relevant items they might like by suggesting products to buy, movies to watch, articles to read, and much more. Additionally, the dynamic nature of user preferences highlights the importance of user profiling, which enables systems to adapt to changing interests. Various profile construction techniques were investigated and a novel RS architecture with a sharable user profile was developed [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

The proposed framework can be adapted to the context of the metaverse. RSs have been investigated in different contexts, including e-commerce applications, peer-to-peer systems [2], [3], and Blockchain [4], [5]. Before computing the recommendations, designing an RS begins with collecting opinions and preferences data. The data collected can be explicit or implicit. Explicit data are given directly from users, such as rating stars, while implicit data is inferred from the user activity (purchase data, video viewing data, browsing history). The next step is selecting the relevant data; for this purpose, data preprocessing is required. Finally, algorithms are then used to compute the recommendations. Over the years, the field of RSs has witnessed significant advancements with the integration of Deep Learning (DL) [6], and more recently, Large Language Models (LLMs) [7]. DL has introduced powerful architectures like graph neural networks [8], and attention mechanisms that excel in capturing complex relationships and sequential patterns [9].

Similarly, LLMs have brought conversational recommendation capabilities, enabling more context-aware user interactions [10]. Despite these advancements, Machine Learning (ML) techniques continue to serve as the backbone of many production RSs due to their simplicity, interpretability, computational efficiency, and faster training and deploying times. Their interpretable nature allows practitioners to understand model behavior which is crucial in applications requiring transparency. Moreover, ML often provide a strong baseline for benchmarking new methods, making them an essential component of RS development. RSs have evolved significantly, with various frameworks being developed by both academic researchers and large corporations. While numerous tools for building these systems have emerged, selecting the most appropriate solution remains challenging. Although these frameworks may employ similar algorithms, they differ in implementation, data management strategies, and evaluation methods. The choice of an appropriate software framework and ML algorithm for an RS is difficult and depends on specific requirements and the desired level of accuracy, efficiency, and interoperability. This paper addresses these challenges by providing a comparative analysis of RS tools and their underlying ML algorithms to help practitioners and researchers select the most suitable solution for their specific user cases.

Thus, this paper provides the following contributions:

- A concise overview of different ML algorithms used in RSs.
- A comprehensive comparison of open-source RS software, examining their supported algorithms, and features.
- An evaluation of reviewed tools across multiple dimensions, including programming language, recommendation techniques, usability, flexibility, scalability, efficiency, extensibility, and modularity.
- A presentation of the most frequently used performance metrics to evaluate RSs.
- Conducting experiments using different recommender frameworks: Lenskit [11], Microsoft Recommenders [12], Turi Create [13], and Cornac [14] to assess the performance of different ML algorithms.
- Conducting simulations using several datasets including MovieLens [15], Jester [16], BookCrossing [17], and Epinions [18], which have been commonly used in RS research.
- Analysis of the results to provide new insights on the performance of different ML algorithms in the context of RSs.

This paper is organized as follows: Section II briefly introduces RSs. A brief overview of ML algorithms used in RS is given in Section III. Section IV presents a comprehensive review of open-source RS software libraries. Section V presents the evaluation methodology of the software. Section VI classifies the most used evaluation metrics. Section VII presents the experimental implementation, while Section VIII explains and analyzes the results. Section IX concludes the paper.

II. RECOMMENDER SYSTEMS

RSs have one of two tasks: prediction or top-N recommendation. Prediction refers to estimating how much a user would rate a new item to determine whether or not he would like it. Top-N recommendations are tasks that recommend a list of items that users may like [19]. RSs algorithms are mainly categorized into three different groups: Content-Based (CB), Collaborative Filtering (CF), and hybrid RSs.

A. CONTENT-BASED RECOMMENDER SYSTEMS

CB systems look at the attributes of an item rated highly by users in the past and then recommend similar items. For example, while recommending movies, features such as actors and genre may be considered. When a person expresses an interest in action films, a CB algorithm suggests the most appropriate action films. A CB system can recommend new items and does not require data from other users. However, it is difficult to find the appropriate features.

B. COLLABORATIVE FILTERING RECOMMENDER SYSTEMS

This class of RSs provides recommendations to the target user, to whom recommendations are made based on similar users' preferences. The core idea of CF is that users with similar preferences in the past are likely to have similar preferences in the future. CF can be divided into memory-based and model-based techniques. Memory-based methods compute the similarity between users or items based on user ratings of historical data. Model-based relies on ML models to make predictions. One of the main advantages of CF recommenders is making predictions without requiring prior explicit knowledge about items or their characteristics. On the other hand, sparsity and cold start are the main issues. The cold start problem frequently occurs when a new item or user enters the dataset. For example, when a new user arrives, the system is unaware of his specific preferences as there is no user history. Sparsity is due to insufficient rating data because only a few users rate most of the items.

C. HYBRID RECOMMENDER SYSTEMS

Hybrid methods were proposed to solve the limitations of using each technique separately. They combine two or more recommendation techniques to improve performance. Weighted, mixed, switching, feature combination, feature augmentation, cascade, and meta-level are the seven main hybridization processes used in RSs to generate hybrid RS [20], [21].

III. MACHINE LEARNING ALGORITHMS

For a comprehensive understanding of ML techniques commonly employed in RS, Table 16 in section IV provides a detailed comparison of open-source RS software, along with the algorithms they employs. In this section, we focus on explaining the essential ML algorithms that form the

basis of these systems, in addition to recent advances in RS algorithms.

ML models are developing rapidly due to the increased available data and the need to extract information and knowledge from this data. Different ML models are used in the RS domain to provide people with accurate recommendations and suggest appropriate items and ongoing research is still being conducted. Portugal et al. [22] conduct a systematic review of the literature to investigate the use of ML algorithms in RSs. The paper concludes that both supervised and unsupervised ML algorithms are being well-researched. Clustering algorithms, Ensemble, and Support Vector Machine (SVM) are among the most popular and used algorithms. The paper also highlights the use of neighborhood-based approaches among ML algorithms. In [20], Ko et al. examine various techniques used in RSs. Studies were conducted on text mining, KNN, clustering, matrix factorization, and neural networks. Over an extended period, text mining techniques for analyzing text information and clustering techniques for analyzing user or location data of a similar group for recommendation have been widely studied. However, recently, interest in the high possibility of applying the neural networks technology to an RS has increased. Alam et al. [23] compared various filtering techniques, including K-Nearest Neighbors (KNN), random tree, random forest, BayesNet, logistic regression, LibSVM, JRip, decision table, OneR, and randomized filter classifier. The comparison was based on six metrics: kappa statistic, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), F1 Score, Receiver Operating Characteristic (ROC) area, and accuracy. The kappa statistic is a measure that evaluates a model's performance by quantifying the agreement above chance, where a value of 1 indicates a perfect model. The classifiers are compared on the MovieLens dataset [15]. The paper concluded that the decision table is the best algorithm because it has the highest kappa and accuracy values and the lowest RMSE value.

Jalili et al. [19] conducted a comprehensive survey and comparative analysis of various memory-based and model-based CF algorithms, evaluating their performance across multiple metrics. Five datasets were used, including Movielens [15], LastFM [24], Epinions [18], BookCrossing [17], and Jester [16]. Analysis of the results indicated that no single algorithm consistently outperformed others across all performance metrics. The paper also assessed the algorithms' performance based on a unified metric that combines diversity, novelty, coverage, precision, and Rank-Biased Precision (RBP), for which the Probabilistic Matrix Factorization (PMF) is regarded as the most effective algorithm across all datasets, with the exception of the LastFM dataset [24], where it is the second-best algorithm. In [25], Nilashi et al. use dimensionality reduction and ontology techniques to address two significant drawbacks of recommendation systems: sparsity and scalability. They use Singular Value Decomposition (SVD) to find similar users and items in each cluster of users and items. The proposed method was tested on two real-world movie datasets: MovieLens [15] and IMDB [26]. The results demonstrated that the proposed method improves the sparsity and scalability issues in CF. He et al. [27] suggested a Neural Collaborative Filtering (NCF) model for building RS that takes advantage of the complexity and non-linearity of the Neural Network. It also demonstrates that NCF is a generalization of Matrix Factorization (MF). The proposed model experimented on two datasets: MovieLens [15] and Pinterest. To evaluate the performance, the paper considered two evaluation metrics: NDCG@10, which is a ranking measurement for the top 10 positions in the recommendation list, and Hit Ratio@10, which is the percentage of users for whom the right answer appears in the top 10 of the recommendation list. The NCF is compared with the following methods: ItemPop (a non-personalized method where items are ranked by their popularity), ItemKNN (the item-based CF), Bayesian Personalised Ranking (BPR) (an MF algorithm that uses implicit feedback to predict item ranking), and eALS (a state-of-the-art MF method for item recommendation). In all cases, the proposed model performs better than the other models. Below is a description of the most used ML algorithms.

A. K-NEAREST NEIGHBOR (KNN)

KNN is an ML algorithm that classifies data based on the closest distance by comparing the similarity of each data item. The most commonly used measures to compare similarities are Euclidean distance, Cosine similarity, and Pearson correlation [20]. User-based CF and Item-based CF are among the most used techniques that use KNN.

User-based performs recommendations based on similar users. After forming the neighborhood, the weights of various neighbors are used to predict the new rating for the target user-item pair. In other words, the higher the users' similarity to the target user, the more significant the impact of their ratings on the prediction of the rating of the target user [19].

In contrast, item-based filtering operates under the assumption that if a pair of items receive similar ratings from various users, they are also likely to receive similar ratings from other users. The item-based approach analyzes the rating matrix to identify the similarity between items and subsequently uses the weighted average of the target user's ratings for similar items to generate the prediction [19].

B. CLUSTERING

It is an unsupervised learning method widely used in the CF recommendation model to cluster the user or the item data into similar groups. The most frequent method is to cluster users based on their vector representations. In this case, clustering methods can be applied to users' vectors to group them. After clusters have been constructed, the ratings of other users in that group can be averaged and used to predict unknown items for a target user. The recommendation system employs many clustering techniques, and K Means is the most used one [20].

C. MATRIX FACTORIZATION

This approach is used as a CF method in RSs. The primary goal of this technique is to factorize the rating matrix into two low-rank matrices that express the user information and the item's preference. There are several methods based on MF such as SVD, Non-Negative Matrix Factorization (NMF), Probabilistic Matrix Factorization (PMF), Bayesian Probabilistic Matrix Factorization (BPMF), Non-Linear Probabilistic Matrix Factorization (NLPMF). According to [19], these approaches have various advantages, including higher accuracy, greater scalability, and a straightforward learning procedure. However, the difficulty in learning the model is the main drawback.

D. ENSEMBLE

The Ensemble is an ML technique that combines similar models to improve the results of a single model. As mentioned in [28], the core concept behind classifier ensembles is to build a set of classifiers that can predict class labels by combining their predictions. When classifiers are independent, classifier ensembles work. In this situation, the Ensemble will give results that are at least as bad as the Ensemble's worst classifier. Indeed, some research such as [29] demonstrates that an ensemble of several distinct and more straightforward algorithms outperforms any single algorithm. In RS, using ensembles of classifiers is a widespread technique. Because it integrates many classifiers, any hybrid approach can be called an Ensemble.

E. SUPPORT VECTOR MACHINE (SVM)

The SVM is a supervised classification method that determines the best hyperplane for separating data points and how well it generalizes to new data. This type of hyperplane is known as a maximum margin hyperplane. It optimizes the distance between the closest points in each class [30]. According to [31], using an SVM classifier, we can classify a user or an item into a specific group. This problem can be presented in two ways. The first approach is to consider each item as a separate classification problem. We can create a classifier based on item i to predict which group the active user belongs to. Each user u can be represented as a vector containing his ratings on items other than the item i. The second approach considers each user as a distinct class and items as training instances, represented as a vector containing ratings given by other users.

F. NAIVE BAYES

Naive Bayes is one of the most widely used supervised techniques for multi-class classification. It is based on Bayes' theorem and the naive assumption of conditional independence between all pairs of variables. The Naive Bayes Classifier model and CF can be combined to constitute an RS and predict if a particular user might like a specific item. Valdiviezo-Diaz et al. [32] proposed an approach based on Naive Bayes. Considering a set of independent variables, the Naive Bayes classifier computes the posterior probability for each class. The independent variables in the suggested technique will be the user ratings of items, and the plausible classes should be each possible rating value. The rating value that maximizes the probability of an item being rated by a given user can be used to make predictions.

G. DECISION AND REGRESSION TREES

Some of the most applied models in data classification are decision and regression trees. Decision trees are intended when the dependent variable is categorical, while regression trees are used when the dependent variable is numerical.

When developing RSs using decision trees, one can face various problems. The rating matrix is very sparse. If an element has an undefined attribute and then this attribute is used to do a split, in which branch will the tree place that element? Additionally, the predicted and observed entries are not separated as feature and class variables. In a rating matrix, which items are dependent or independent is not evident, as they are simply referred to as users and items. As stated in [33], the latter challenge can be solved using separate decision trees for each attribute by fixing one attribute to be dependent and the rest as independent. In other words, we will have a tree for each item then the tree is used to predict the rating of a given item for a user.

H. SLOPE ONE

Slope One is a CF algorithm proposed by Lemire and Maclachlan in [34]. The predictions in Slope One are based on the average difference between the ratings of items given by users who have rated the same items. The idea of Slope One is to find a linear relationship between item preferences in the form of $f(x) = x + b$. x represents the rating, and b is a constant that indicates the average difference between the ratings of two items. The name ''Slope One'' reflects the fact that x is multiplied by 1 in this case. Slope One does not consider the number of ratings observed when making predictions. Therefore, the weighted Slope One has been proposed. If a user has rated multiple items, predictions are computed using a weighted average, where the weight corresponds to the number of users who have rated both items.

I. NEURAL NETWORKS

Neural networks are a subset of ML. The human brain inspires their name and structure, and they mimic how biological neurons interact. Despite their relatively recent introduction to the RS field compared to other domains, neural networks have emerged as a major research focus, with numerous studies being conducted. It has been commonly utilized in research to get more data where understanding user preferences with historical data is challenging [20]. CF approaches can be adopted in the form of neural networks. The goal is to use neural networks to extract features from items in a general way, similar to CB recommendation methods, and then incorporate these features into a CF model [35]. Traditional recommendation algorithms, such as

CF, generally do not take the temporal dynamics and the sequence of activities into account when trying to simulate user behavior [36]. Over time, users' preferences change. Sequential recommendation algorithms can identify temporal patterns in users' browsing behavior and enhance predictions of future interests for improved recommendations. By tracking user interactions over time, these algorithms construct a logical sequence that models evolving preferences, with user input being largely influenced by its temporal order [36].

RSs have seen significant advancements, integrating more sophisticated DL techniques and external knowledge sources to improve recommendation performance and address the limitations of traditional methods. More recently, the adoption of LLMs has further enhanced their efficiency.

In [6], the authors provided a comprehensive overview of DL-based RSs, emphasizing their effectiveness in overcoming critical challenges such as the cold start problem, long-tail recommendations, and data sparsity. Among the key techniques discussed, Autoencoders play a crucial role in dimensionality reduction and feature extraction, thereby addressing missing values in user-item interactions. Convolutional Neural Networks (CNNs) are employed to process visual and textual data, enhancing content-based recommendation accuracy. Meanwhile, Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), are adept at modeling sequential dependencies in user interactions, making them particularly suitable for session-based recommendation tasks. Attention mechanisms further enhance personalization by dynamically adjusting the weight assigned to past interactions, allowing models to adapt to evolving user preferences.

For instance, Yuan et al. [9] introduces the ACA-GRU model, which integrates an attention mechanism with a GRU framework to enhance sequential RSs. Unlike traditional models that may overlook the importance of contextual and adjacent item information, ACA-GRU incorporates multiple types of contextual data such as input, transition, and static context, along with user-item interactions. Experimental evaluations on the MovieLens-100K, MovieLens-1M, and Netflix datasets demonstrate that ACA-GRU outperforms conventional approaches, including matrix factorization, item-based collaborative filtering, and non-context-aware sequential models. Reference [37] introduces a novel post-processing method called Dual Training Error-based Correction (DTEC) to enhance the performance of RSs. DTEC operates as a secondary stage that refines any model-based RS outputs by correcting training errors from user and item perspectives. This dual approach, combining User Training Error Correction (UTEC) and Item Training Error Correction (ITEC), optimizes recommendations by reducing the prediction error in the training set and applying learned corrections to test data. Reference [38] proposes a novel long-term RS using deep reinforcement learning (DRL) to address limitations of traditional CF models, which struggle with dynamic, sequential interactions and user cold-start issues. By modeling recommendations as a Markov Decision Process (MDP) and incorporating an RNN, the system adapts to user interactions over time.

Graph Neural Network (GNN) have become popular for their ability to model intricate relationships between users and items through graph structures, capturing high-order connectivity and user-item interactions [8]. Darban and Valipour [39] presented a Graph-based hybrid recommendation system (GHRS), which combines graph modeling with user demographic and geographic data to enhance movie recommendation accuracy. This approach uses an Autoencoder for feature extraction, integrating user ratings and auxiliary information to create robust features. These features facilitate more effective user clustering, addressing the cold-start problem and boosting overall recommendation performance compared to traditional methods.

As surveyed by [40], recent attention has focused on self-supervised learning (SSL) techniques, which enable models to learn representations without labeled data. Additionally, self-supervised hypergraph learning, which models higher-order relationships through hypergraph convolutional networks (HGNNs), has proven effective in group recommendation systems. Moreover, contrastive learning, which has been extensively studied in representation learning, has been applied in RSs to distinguish between similar and dissimilar user-item interactions. Further details on other recent RS techniques can be found in this survey [40].

Knowledge-based RSs differ from CF and CB RSs as they do not rely on historical user-item interactions. These systems are especially valuable in complex domains where user ratings are unavailable or insufficient. Knowledge-based RS leverage domain knowledge, whether externally sourced or extracted from datasets, to improve recommendation quality and mitigate the cold start problem. The integration of knowledge graphs (KGs) has become a central focus, with systems using structured entities and relationships (examples of Knowledge Graphs include YAGO [41], DBpedia [42], Freebase [43]) to enhance user-item interactions. Knowledge graph embeddings (e.g., TransE [44], TransH [45]) transform these entities and relations into low-dimensional vectors while maintaining their semantic meanings. According to [46], graph embedding-based methods excel in predicting explicit user-item interactions, particularly with large data scales. These methods are also superior for big data applications due to the reusability of learned embeddings. Other advanced methods like Knowledge-Aware Coupled GNN (KCGN) [47] and Knowledge Graph-based Intent Network (KGIN) [48] model high-order relationships and help improve recommendation quality and interpretability.

LLM-based RSs represent another significant advancement, leveraging the extensive knowledge and human-like interaction capabilities of models such as ChatGPT,[1] to improve RS accuracy and user experience. These models enhance RS by leveraging pre-trained knowledge, natural language understanding, and contextual reasoning. They

[1] https://chatgpt.com/

generate top-k recommendations, predict ratings, and provide explanations through task-specific prompting. They also improve explainability by generating natural language justifications for recommendations. Additionally, they augment data by synthesizing pseudo-user interactions, thereby mitigating data sparsity issues. [7].

For instance, Gao et al. [10] demonstrated a ChatGPT-based RS capable of providing cross-domain recommendations, showcasing LLMs' potential for engaging, context-aware interactions. Chu et al. [49] introduced RECSYSLLM, which uses fine-tuned LLMs with specialized training and inference techniques to deliver highly personalized recommendations. Moreover, in conversational recommender systems, LLMs can refine recommendations in real-time dialogues, adapting dynamically to user preferences. Additionally, domain-specific LLMs specialize in recommendations for sectors such as healthcare, finance, and law, outperforming general-purpose models when trained on high-quality domain data [7].

While deep learning-based recommender systems (DLRS) and LLM-based RSs offer substantial advancements in performance, adaptability, and personalization, they also introduce significant challenges in computational complexity, interpretability, and scalability when compared to traditional ML-based approaches. DLRS excel in capturing intricate, non-linear user-item relationships, leveraging multi-modal data sources such as images, text, and behavioral signals to improve recommendation accuracy and user experience. Similarly, LLMs enhance RS by leveraging pre-trained knowledge, natural language understanding, and contextual reasoning, enabling conversational and cross-domain recommendations. However, these advancements come at a cost. DLRS require extensive computational resources, high-memory GPUs, and large-scale datasets. Likewise, LLM demand massive computational power for both training and inference, further limiting their feasibility for scalable, real-time deployments.

Interpretability remains another key concern. DLRS function as ''black-box'' models, making it challenging to explain recommendation decisions, which is particularly problematic in high-stakes domains requiring transparency. Similarly, LLM-based RS face issues of hallucination, where they generate plausible yet factually incorrect recommendations, posing risks in healthcare, finance, and legal advisory systems [7]. In contrast, traditional ML-based RSs including CF, MF, and decision trees offer higher transparency by relying on explicit feature engineering, structured data, and rule-based decision-making. Additionally, task-specific ML models require less fine-tuning to generalize within their trained domains, unlike LLMs, which struggle with domain adaptation without extensive retraining.

IV. COMPARISON OF OPEN-SOURCE RECOMMENDER SYSTEM SOFTWARE

This work aims to provide an exhaustive list of software used in RSs. This section highlights a selection of open-source software options, which offer notable benefits including cost-effectiveness and transparency through accessible source code. A comprehensive list of open-source RS software is provided in Table 16 of the appendix. The table is adapted and extended from https://recommender-systems.com/resources/software-libraries/. We have supplemented this table with additional libraries to provide a comprehensive overview. Table 16 contains 42 software that help developers build and evaluate RSs.

Although there are numerous compelling frameworks available, the majority of them are either no longer actively developed, narrowly focused on a single aspect of RSs, or lack comprehensive documentation.

Neural RSs dominate the recommendation space, which aligns with the general trend toward more advanced models. The last check for the information in Table 16 was on December 02, 2024.

Table 16 presents a comparison of RS software based on several key characteristics. These characteristics include the number of stars on GitHub, the algorithms supported by each software, and the datasets available:

- Release date: The date the framework or library was released.
- Last update: The date of the most recent update of the software.
- Dataset: The dataset supported by the framework. Examples include MovieLens [15], LastFM [24], and IMDB [26], among others.
- Techniques: There are many techniques as mentioned in [50], it could be CF, CB, Demographic-based Filtering (DBF), Knowledge-Based Filtering (KBF), and Hybrid approaches. There are also some advanced techniques such as session/sequential-based recommendation.
- Algorithms: The models implemented by the software.
- Type of the feedback: Most RSs rely on two distinct types of data collection to predict users' interests and construct personalized user profiles:
  - Explicit feedback involves users directly expressing their interests through actions such as commenting, tagging, rating, liking, or bookmarking content. Ratings are often collected on a numerical scale (e.g., 1 to 5 stars).
  - Implicit feedback gathers user behavior data indirectly by observing their actions within the application. Examples include tracking items viewed, purchased, or interacted with, as well as analyzing interaction frequency, content consumption duration, and social network activity. Implicit feedback operates passively in the background, requiring no direct input from users.

To begin, we point out that some of the open-source software for RSs was created in a programming language other than Python. MyMediaLite [51] is a C# library that provides implementations of known algorithms as

well as basic procedures for evaluating recommendation performance. LibRec [52], on the other hand, is a Java library that implements a large number of algorithms, metrics, and data partition strategies.

The R package Recommenderlab is used to create and evaluate recommender algorithms. There are some software that are in desperate need of renovation, the ones that are not actively updated such as PREA, and MyMediaLite.

With the growing prominence of Python in data science applications, there is a high demand for Python-based RS frameworks. Beyond Cornac [14], other notable options include LightFM [53], which employs a factorization machine to model the interactions between user and item features. TensorRec, which permits non-linear interactions, and Surprise [54], which utilizes rating prediction techniques such as MF. Contrastingly, the Implicit framework focuses on algorithms for handling implicit feedback data. Several social and text-based recommender models are included in certain frameworks (e.g., LibRec), despite not being created expressly for multimodality. Nonetheless, they delegate the responsibility for dealing with auxiliary data to model implementations. Their data processing pipelines are not as standardized as those in Cornac, which is a constraint. Notably, other libraries rarely include graphic recommenders. LibRec also has more baseline and extension algorithms, such as NMF, than PREA and MyMediaLite. PREA only gives predicted error-based metrics for evaluating recommendation performance, while MyMediaLite does not provide novel measurements beyond accuracy. LibRec, on the other hand, offers both unique and standard accuracy-based measurements. Finally, this paper [52] shows that LibRec is much faster than PREA and MyMediaLite while still providing comparable recommendation results. The amount of performance improvement varies per algorithm.

Despite the existence of some open-source Python recommendation software like Suprise and OpenRec [35], [54], most of these tools only support traditional recommendation algorithms or have not been updated in a long time. Additionally, while the OpenRec library includes numerous deep neural network-based recommendation models, the number of such models is relatively limited compared to those available in DeepRec [35], [36].

Surprise, MyMedialite, and LibRec do not incorporate neural network-based models and cannot make use of GPU computing capabilities. Although OpenRec uses TensorFlow as its backend [35], it only implements four ranking-based recommendation techniques. DeepRec [36] incorporates over ten cutting-edge DL-based recommendation models. DeepRec is also the first library to provide sequence-aware models.

EasyRec is a web service that can be added to websites as an RS, but it does not have any advanced personalization algorithms. Instead, it is more of a framework for connecting a recommender service to an application.

V. EVALUATION OF OPEN-SOURCE RECOMMENDER SYSTEM SOFTWARE

We have identified the following measures that will be evaluated against a variety of existing software: Programming language, recommendation techniques, usability, flexibility, scalability, efficiency, extensibility, and modularity.

A. PROGRAMMING LANGUAGE

Python is a widely utilized programming language that exhibits numerous advantages, making it a prominent choice for constructing RSs. Its productivity allows programmers to write and test code expeditiously, and its extensive library of functions and modules enables developers to accomplish a wide range of tasks without the need to write new code from scratch. Furthermore, Python's simplicity in combining various data types makes it an excellent tool for data analysis, a crucial component in building effective RSs. Additionally, the availability of numerous ML libraries based on Python further reinforces its suitability for projects involving the development of RSs. Given the advantageous attributes of Python described, we consider it a suitable programming language for RS software development purposes.

B. RECOMMENDATION TECHNIQUES

As discussed in Section III, recommendations are generated using well-known state-of-the-art approaches from the RS literature, such as CF, item-based CF, and MF. This is partly because the techniques in question are readily available in comparable frameworks, but this is not the case for less traditional techniques. We compare the software based on the techniques used.

C. USABILITY

Usability in RS software can be characterized by the system's ability to be implemented with minimal code, allowing even non-experts in application development to create customized recommendation solutions.

D. FLEXIBILITY

The ability of software code to adapt to changing requirements, incorporate new techniques, and handle diverse data structures without extensive reworking is known as flexibility. In the context of RSs, this flexibility is crucial due to the dynamic nature of user preferences, data sources, and recommendation algorithms. A flexible RS can effortlessly integrate various methods, and tailor the recommendation strategies to enhance personalization [55], [56], [57].

E. SCALABILITY

The capacity of a system to scale up or down in performance and cost in response to changes in application and system processing demands. Because data in RS tends to grow over time, scalability is an important factor to consider [14], [56], [57]. Scalability is often tested by examining how the system performs and how many resources it consumes as the amount of data changes.

F. EFFICIENCY

Efficiency measures the resources invested to produce a specific output. In software development, ''efficiency'' refers to the ratio of resources used (such as time and effort) to the amount of software developed or requirements met. In developing RSs, efficiency enables developers to quickly prototype and experiment with different parameters to optimize performance. For instance, OpenRec [35] allows rapid experimentation, while RecBole [58] demonstrates efficiency by completing comprehensive ranking tasks on large datasets, like MovieLens-10M, in just around two seconds. This highlights how efficient implementations can significantly improve development speed and system responsiveness.

G. EXTENSIBILITY

Extensibility refers to a software system's capacity to enable and accept considerable expansions of its capabilities without requiring major code rewrites or modifications to its core design. Extensible systems give developers technology, tools, and languages that allow them to extend or add to their capabilities [35], [56].

H. MODULARITY

Modularity is the ability to break software into smaller components with standardized interfaces. The goal is to build products by merging reusable blocks of codes, so the developer only needs to develop a feature or functionality once and then reuse it as many times as possible [14], [35].

The best tools were selected based on the features discussed, in addition to their ability to address issues such as the lack of support for algorithmic modularity and lack of reliable back-end support [35]. Table 1 showcases the top 10 software solutions that align with the desired features, and have also received recent updates.

VI. EVALUATION METRICS

This section aims to provide a detailed explanation of key metrics used to assess the performance of RS. This will facilitate a thorough understanding of the most commonly used performance metrics employed in the various open-source RS software presented in Table 16.

As mentioned previously in Section II. RSs can perform one of two tasks: prediction or top-N recommendation. For this reason, RS can be evaluated based on the predicted ratings or the relevancy of recommended items. In [19], the authors classified the performance metrics into accuracy metrics, rank-based metrics, and diversity, novelty, and coverage metrics. A summary is presented below, while Table 2, presents a detailed summary of the evaluation metrics together with symbols, formulas, ranges, and the targets (whether the optimal value of the metric should be higher or lower).

A. ACCURACY METRICS

According to [19], accuracy metrics can be divided into three main groups: predictive accuracy metrics, rank accuracy metrics, and classification metrics.

1) PREDICTIVE ACCURACY METRICS

Different predictive accuracy metrics are used to assess the accuracy of a recommender's predicted ratings in relation to the actual ratings. The most common measures are MAE and RMSE. MAE represents the average difference between the predicted ratings and the actual ratings given by users for a set of items. In contrast, RMSE is a variant of MAE that emphasizes significant errors by using power two on the difference between predicted and actual ratings.

2) RANK ACCURACY METRICS

Rank accuracy metrics such as Pearson correlation and Spearman correlation coefficients are used to measure the relationship between the order of items in a recommendation list and the order each user assigns to the same item.

- Pearson correlation: It quantifies the linear relation between two lists of predicted ratings and a user's actual ratings.
- Spearman correlation: Unlike the Pearson correlation, the Spearman correlation measures the linear relationship between two lists of the rank of predicted ratings and actual ratings instead of the value of ratings.

3) CLASSIFICATION ACCURACY METRICS

The classification accuracy metric indicates how accurately the system can classify a pertinent item as good or a non-pertinent item as bad. Precision and recall are two of the most commonly used classification metrics to assess RS performance.

- Precision: The precision determines the ratio of items in the recommendation list that the user likes.
- Recall: The recall determines the ratio of the recommended relevant items out of the total number of relevant items.

B. RANK-BASED METRICS

Rank-based metrics focus on the position of items within the recommendation list rather than on comparing predicted ratings to actual ones. This measures group includes metrics such as Normalized Discounted Cumulative Gain (NDCG), Recovery Rate, and Rank Biased Precision (RBP).

- Normalized Discounted Cumulative Gain: The Discounted Cumulative Gain (DCG) metric evaluates an item's usefulness based on its position in a recommendation list. The basic idea behind this is that the more relevant items appear at the top of the recommendation list, the more valuable the RS is. DCG is then normalized by the ideal ranking based on user preferences to compute NDCG.
- Recovery Rate: The Recovery Rate considers the correct ranking of items to assess the performance.
- Rank Biased Precision: The RBP metric, like DCG, evaluates the recommendation list by assigning more weight to highly ranked relevant items. The key difference between DCG and RBP is that RBP uses a

TABLE 1. Top 10 RS libraries selected based on key features: scalability, modularity, flexibility, usability, efficiency, and extensibility.

Features / Software	Scalability	Modularity	Flexibility	Usability	Efficiency	Extensibility
Recommenders	✓	✓	✓	✓	✓	✓
Transformers4Rec	✓	✓	✓	-	✓	✓
Turi Create	✓	-	✓	✓	✓	-
Cornac	✓	✓	✓	-	✓	✓
Lenskit	-	✓	-	✓	✓	-
Surprise	-	-	-	✓	-	-
RecBole	✓	✓	✓	✓	✓	✓
TorchRec	✓	✓	✓	-	✓	✓
Implicit	-	✓	✓	-	-	-
BETA-Rec	✓	✓	✓	✓	-	✓

geometric discounting function, whereas DCG applies a logarithmic one.

C. DIVERSITY, NOVELTY, AND COVERAGE

A well-designed RS must suggest relevant items to users and provide better recommendations. Still, it would be desirable to bring users additional satisfaction by recommending novel and diverse items. To determine how many of the items in the recommendations list are new and surprising to the user, or whether the system suggests the same set of items to all users every time, metrics such as Novelty, Diversity, and Coverage are used.

1) DIVERSITY

Diversity refers to the variety of items on the recommendation list. It consists of two notions: intra-diversity and inter-diversity. The first one aims to determine how different the recommended items to a user are. The second one is used to indicate the degree of difference between all users' recommendation lists.

2) NOVELTY

Novelty often refers to the presence of new items in the recommendation list. The main idea behind the majority of RS approaches is that users will prefer items that are similar to what they have previously liked. Still, the recommender can be successful if it can recommend new and good items that will capture users' attention to like them. To measure the novelty metric, the popularity metric is used to compute the popularity of items in the recommendations list. The more the items are popular, the more likely they are consumed by the user, making them less novel.

3) COVERAGE

A good recommender should also cover a wide range of items. For this reason, another concept that has been proposed is Coverage. Coverage suggests that the system performs better as the proportion of items a recommender may propose increases.

- Catalogue Coverage: The portion of distinct items in users' top recommendations lists is referred to as Catalog Coverage.
- Entropy Coverage: Besides the number of items covered by a recommender, the percentage of different items offered may vary. Several items are regularly recommended to multiple users, while others appear less frequently in the recommendation lists. A variant of Coverage called Entropy Coverage is used to examine how often an item is suggested to users.

VII. IMPLEMENTATION

This section presents the procedures for generating recommendations, the recommender frameworks, the datasets used, and the performance metrics for evaluating the algorithms. Out of the top 10 tools evaluated based on measures such as programming language, recommendation techniques, usability, flexibility, scalability, and efficiency, we have selected four specific recommender frameworks for implementation: Lenskit [11], Microsoft Recommenders [12], Turi Create, and Cornac [14].

A. RECOMMENDER FRAMEWORKS

1) **LensKit:**[2] LensKit [11] is an open-source RS software that can be used for building, researching, and learning about RS—first released as a Java Framework. The LensKit for Python (LKPY) is the next generation of the LensKit project that supports running, training, and evaluating various RS. LKPY provides non-personalized and personalized algorithms such as KNN collaborative filtering and Matrix Factorization models.
2) **Microsoft Recommenders:**[3] The Microsoft Recommenders repository is an open-source project that provides several Python utilities and Jupyter notebooks that aid in designing, evaluating, and deploying RSs. It supports both Linux and Windows platforms. Some algorithms work on CPU/GPU environments, and others require a Spark environment. Several utilities are included in Recommenders, such as loading datasets in the format required by various algorithms, evaluating model outputs, splitting into training and

[2]https://github.com/lenskit/lkpy
[3]https://github.com/recommenders-team/recommenders

TABLE 2. Summarization of evaluation metrics. The range of the Intra-list Diversity depends on the used similarity function $s(I_i, I_j)$.

Symbol	Metric	Formula	Range	Target	References
MAE	Mean Absolute Error	$\frac{1}{N}\sum_{i=1}^{N}\lvert r_{ui}-\tilde{r}_{ui}\rvert$	$[0,+\infty[$	Lower	[19]–[23], [59]
RMSE	Root Mean Squared Error	$\frac{1}{N}\sum_{i=1}^{N}\sqrt{(r_{ui}-\tilde{r}_{ui})^2}$	$[0,+\infty[$	Lower	[19]–[23], [59]
PC	Pearson Correlation	$\frac{\sum_{i=1}^{N}(r_{ui}-\bar{r}_u)(\tilde{r}_{ui}-\bar{\tilde{r}}_u)}{\sqrt{\sum_{i=1}^{N}(r_{ui}-\bar{r}_u)^2(\tilde{r}_{ui}-\bar{\tilde{r}}_u)^2}}$	$[-1,1]$	Higher	[19]
SC	Spearman Correlation	$\frac{\sum_{i=1}^{N}(r_i-\bar{r}_u)(\hat{r}_i-\bar{\hat{r}}_u)}{\sqrt{\sum_{i=1}^{N}(r_i-\bar{r}_u)^2(\hat{r}_i-\bar{\hat{r}}_u)^2}}$	$[-1,1]$	Higher	[19]
P	Precision	$\frac{\lvert C_u\cap R_u\rvert}{\lvert R_u\rvert}$	$[0,1]$	Higher	[19]–[22]
R	Recall	$\frac{\lvert C_u\cap R_u\rvert}{\lvert C_u\rvert}$	$[0,1]$	Higher	[19]–[22]
F1	F1 score	$\frac{2PR}{P+R}$	$[0,1]$	Higher	[19]–[23]
NDCG	Normalized Discounted Cumulative Gain	$\frac{\sum_{i=1}^{b}R_i+\sum_{i=b+1}^{N}\frac{R_i}{log_b r_i}}{DCG_{max}}$	$[0,1]$	Higher	[19], [22]
RBP	Rank-Biased Precision	$(1-p)\sum_{i=1}^{N}(R_i\times p^{i-1})$	$[0,1]$	Higher	[19]
RR	Recovery Rate	$\frac{1}{\lvert N_R\rvert}\sum_{i\in N_R}\frac{r_i}{C_i}$	$[0,1]$	Lower	[19]
ID	Intra-list Diversity	$\frac{\frac{1}{N(N-1)}\times\sum_{i\neq j}s(I_i,I_j)}{m}$		Higher	[19]
H	Hamming distance (diversity)	$\frac{1}{m(m-1)}\sum_{u=1}^{m}\sum_{u\neq u'}1-(\frac{Q_{uu'}}{N})$	$[0,\frac{1}{2}]$	Higher	[19]
Pop	Popularity	$\frac{1}{m\times N}\sum_{u=1}^{m}\sum_{i=1}^{N}d_i$	$[0,1]$	Lower	[19]
C	Catalogue Coverage	$\frac{I_r}{n}$	$[0,1]$	Higher	[19]
EC	Entropy Coverage	$-\sum_{i=1}^{m}p_i log_2 p_i$	$[0,+\infty[$	Higher	[19]

test data, and several state-of-the-art algorithms are implemented [12], [60].

3) **Turi Create:**[4] Turi Create is an Apple open-source project that simplifies the development of different ML models. It provides toolkits for image classification, object detection, recommendations, and several others. Turi Create recommender offers a unified interface for training and making recommendations using various recommender models. It implements algorithms like Item Similarity, Factorization Recommender, and Ranking Factorization Recommender.
4) **Cornac:**[5] Cornac is an open-source Python framework for multimodal RS. It is unique in that it emphasizes recommendation models that exploit additional data, such as item textual descriptions and product images. In addition to essential utilities for accessing, developing, evaluating, and comparing recommender models, it also comes with extensive documentation, tutorials, and various built-in datasets [14].

[4]https://github.com/apple/Turi Create

[5]https://github.com/PreferredAI/cornac

TABLE 3. Key notations.

Symbol	Meaning
I_u	The set of items rated by user u.
$I_{uu'}$	The set of items rated by both user u and u'.
R_u	List of recommendations for user u.
N	Number of items in the recommendations list.
n	The total number of items in the system.
m	The total number of users in the system.
C_u	List of items the user consumed.
r_{ui}	The actual rating of user u for item i.
$\tilde{r}_{ui}$	The predicted rating of user u for item i.
$\bar{r}_u$	The average of actual ratings of user u.
$\bar{\tilde{r}}_u$	The average of predicted ratings of user u.
r_i	The rank of the item i in the recommendations list.
b	The base of the logarithm.
R_i	The relevancy of item i in the recommendation list.
DCG_{max}	The maximum possible gain.
N_R	The set of relevant items in the actual rating list of user u.
C_i	The number of candidate items to recommend to user u.
$s(I_i, I_j)$	The similarity between items i and j.
$Q_{uu'}$	The number of common item in both u and u''s recommendation lists.
d_i	The degree of item i (has the user already consumed the item?).
I_r	The total number of distinct items in top-N lists of users.
p_i	The percentage of recommendation lists containing item i.

Based on our experience, the features and drawbacks of each RS tool are given in Table 4.

B. DATASETS

We used the following six datasets: MovieLens 100K, MovieLens 1M, MovieLens 10M [15], Jester [16], Bookcrossing [17], and Epinions [18]. These datasets are widely employed by the research community and constitute the most frequently utilized dataset in the recommender software presented in Table 16.

1) **MovieLens:**[6] MovieLens is a dataset collected by GroupLens Research from the MovieLens website across varying periods based on the size of the dataset. It has been widely used to evaluate RS algorithms. MovieLens dataset describes the user's preferences for movies in terms of ratings. The ratings are on a scale of 0.5 to 5 stars, except for MovieLens 1M, which contains ratings from 1 to 5. In all datasets, each user has rated at least 20 movies [15].
2) **Jester:** Jester[7] is a joke dataset that was created by the Jester Research project at the University of California, Berkeley Laboratory for Automation Science and Engineering. It has ratings for 100 jokes. Ratings are values from -10.00 to +10.00 [16].
3) **Bookcrossing:**[8] BookCrossing is a dataset that can be used for book recommendations. Cai-Nicolas Ziegler collected it from the Book-Crossing community. The dataset contains either explicit ratings, given on a scale of 1-10 (with higher values indicating higher appreciation), or implicit, characterized by a 0-1 scale [17].
4) **Epinions:**[9] Epinions was collected by Paolo Massa in a 5-week crawl (November/December 2003) from the Epinions.com website. It includes users' opinions about commercial items as ratings ranging from 1 to 5 [18].

The characteristics of the datasets, including the number of ratings, users, and rated items, are summarized in Table 5.

C. PROCEDURE OF IMPLEMENTATION

In this work, we implemented different memory-based and model-based algorithms provided by the recommender frameworks. A brief description of the implemented algorithms is given in Table 6. Different similarity metrics were utilized to implement some of the algorithms, and a detailed

[6]https://grouplens.org/datasets/movielens/
[7]https://grouplens.org/datasets/jester/
[8]https://grouplens.org/datasets/book-crossing/
[9]https://www.kaggle.com/datasets/masoud3/epinions-trust-network

TABLE 4. RS tools, features, and drawbacks.

Tool	Year of first release	Used version	Backed by	Features	Drawbacks
LensKit [61]	2018	0.14.1 Mar 11, 2022	Boise State University	- It is easy to use. - Minimal dependencies and simpler setup. - The documentation is easy to understand and implement.	- There is not much control over the customization of models. - Data splitting strategies are limited. - Limited Advanced Algorithms.
Recommenders [62]	2021	1.1.0 April 1, 2022	Microsoft	- Provides implementations of a wide range of algorithms. - The documentation is easy to understand and implement. - Scalable and flexible.	- Some algorithms like SAR are high space complexity. - Complex Setup. - Heavy Dependencies.
Turi Create [13]	2017	6.4.1 Sep 30, 2020	Apple	- Simplifies the development of Machine Learning models in a few lines of code. - It provides clear and easy documentation.	- It lacks performance metrics, only RMSE, precision, and recall are available. - Only one data splitting strategy is provided. - There is not much control over the customization of models.
Cornac [63]	2018	1.14.2 Feb 19, 2022	Preferred.AI	- Cornac models can be implemented in a few lines of code.	- There is not much control over the customization of models. - Data splitting strategies are limited.

TABLE 5. The number of ratings, the number of items, and the number of users in each evaluation dataset.

Dataset	#Ratings	#Items	#Users	Reference
MovieLens 100k	100,004	9,066	671	[15]
MovieLens 1M	1,000,209	3,706	6,040	[15]
MovieLens 10M	10,000,054	10,677	69,878	[15]
Jester	1,810,455	100	24,983	[16]
BookCrossing	87,805	59,401	310	[17]
Epinions	127,498	63,310	480	[18]

description of each metric and its corresponding formula can be found in Table 7. Fig. 1 shows a flow chart outlining the methodology used in the implementation process.

All experiments were conducted on the Toubkal High-Performance Computing system[10] at the University Mohammed VI Polytechnic, Morocco. The setup featured an Intel Xeon Platinum 8276 CPU (112 cores, ∼1.47 TiB RAM) and an NVIDIA A100-SXM4-80GB GPU, running on Rocky Linux. The implementation is conducted using Python 3.8.5. The current versions of the recommender frameworks at the time of implementation were: Recommenders 1.1.0, Lenskit 0.14.1, Turi Create 6.4.1, and Cornac 1.14.2. The code for the conducted experiment can be found on GitHub.[11]

[10]https://toubkal.um6p.ma/
[11]https://github.com/ayoubakh/ML-algorithms-in-RS

D. PREPROCESSING

Some of the datasets used in the implementation were first preprocessed:

- Within the BookCrossing dataset, users have rated some of the books that are not part of the original books dataset. We construct a new rating dataset corresponding to users and books in the actual users and books dataset. As the BookCrossing dataset includes both explicit and implicit ratings, we only considered the explicit ratings. To reduce memory consumption, we considered users who rated at least 125 books and books with at least 125 ratings.
- In the Epinions dataset, ratings are from 1 to 5, but there are some ratings of the value of 6 that we treat as 5. We also reduced the Epinions dataset size by considering users who rated at least 150 items and those with at least 150 ratings.
- In the Jester dataset, we first created a column for user IDs and then dropped the column that contains the number of jokes rated by each user. Rather than having a column for each joke, we reformatted the dataset with columns for user ID, joke ID, and rating. In the Jester dataset, the value 99 corresponds to a null rating, i.e., the user did not rate this joke, so we removed the rows with a null rating.

E. SPLITTING STRATEGIES

Each recommender tool supports distinct splitting strategies, and a unified approach is not always feasible. Therefore, the

TABLE 6. The implemented algorithms of different RS tools.

Tool	Algorithm	Description
LensKit [62]	UserUser	Implement the user-user collaborative filtering algorithm using centered-mean cosine similarity.
	ItemItem	Implement the item-item collaborative filtering algorithm using centered-mean cosine similarity.
	SVD	A regularized biased Matrix Factorization technique trained with featurewise stochastic gradient descent which is used to predict ratings.
	ALS	Implements biased Matrix Factorization for explicit feedback learned with alternating least squares and optimized with coordinate descent.
Recommenders [63]	SAR	Smart Adaptive Recommendations (SAR) is a personalized recommendation algorithm based on user transaction history that is designed to rank top items for each user.
	ALS	Alternating Least Squares (ALS) is a collaborative filtering algorithm provided by Spark MLlib that can be used for training an MF model.
	BPR	BPR is a MF algorithm used for predicting item ranking with implicit feedback.
	SVD	SVD is a MF algorithm for predicting explicit rating feedback implemented by Surprise library.
Turi Create [13]	Item Similarity	A model that ranks items based on their similarity to other items observed for the user in question.
	Factorization Recommender	A matrix factorization algorithm that learns latent factors for each user and item and then uses them to predict rating.
	Ranking Factorization Recommender	A Ranking Factorization Recommender learns latent factors for each user and item and uses them to rank recommended items.
Cornac [64]	UserKNN	User-Based Nearest Neighbor.
	ItemKNN	Item-Based Nearest Neighbor.
	SVD	The implementation is based on Matrix Factorization with biases.
	BPR	Bayesian Personalized Ranking.

TABLE 7. Similarity metrics.

Similarity Metrics	Description	Formula
Cosine Similarity	Measures the similarity using the cosine of the angle between two vectors in a multidimensional space.	$COS(u,u') = \frac{\sum_{i \in I_{uu'}} r_{ui} \times r_{u'i}}{\sqrt{\sum_{i \in I_{uu'}} (r_{ui})^2} \sqrt{\sum_{i \in I_{uu'}} (r_{u'i})^2}}$
Jaccard Coefficient	The Jaccard measures the similarity between two sets by counting the number of common items and dividing by the total number of unique items between them.	$JC(u,u') = \frac{\lvert I_u \cap I_{u'} \rvert}{\lvert I_u \cup I_{u'} \rvert}$
Centered-Mean Cosine / Pearson Correlation	Centered cosine similarity normalizes the ratings across all the users. All the ratings for a user are subtracted from his average rating.	$CMC(u,u') = \frac{\sum_{i \in I_{uu'}} (r_{ui} - \bar{r}_u) \times (r_{u'i} - \bar{r}_{u'})}{\sqrt{\sum_{i \in I_{uu'}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uu'}} (r_{u'i} - \bar{r}_{u'})^2}}$

most appropriate strategy was chosen for each framework as the study focuses on comparing algorithms within their respective frameworks. Notably, Turi Create supports only the *Random Split by User*, which was adopted for evaluations in this framework. The following are the splitting strategies used for each tool:

- **Recommenders:** Stratified Split, which preserves user-item interaction distributions to ensure balanced user-centric evaluations.
- **Lenskit:** User-Based Cross-Validation, which ensures robust, user-focused evaluations by splitting data at the user level across multiple folds.
- **Turi Create:** Random Split by User, which selects test users and allocates a proportion of their items to the test set while retaining sufficient training data for personalized recommendations.
- **Cornac:** Ratio Split, which divides the data into training and test sets based on a specified ratio

F. PERFORMANCE EVALUATION

The considered recommender frameworks provide several predictions and top-N evaluation metrics to evaluate the models. In our implementation, we evaluate algorithms based on Normalized Root Mean Squared Error (NRMSE) as

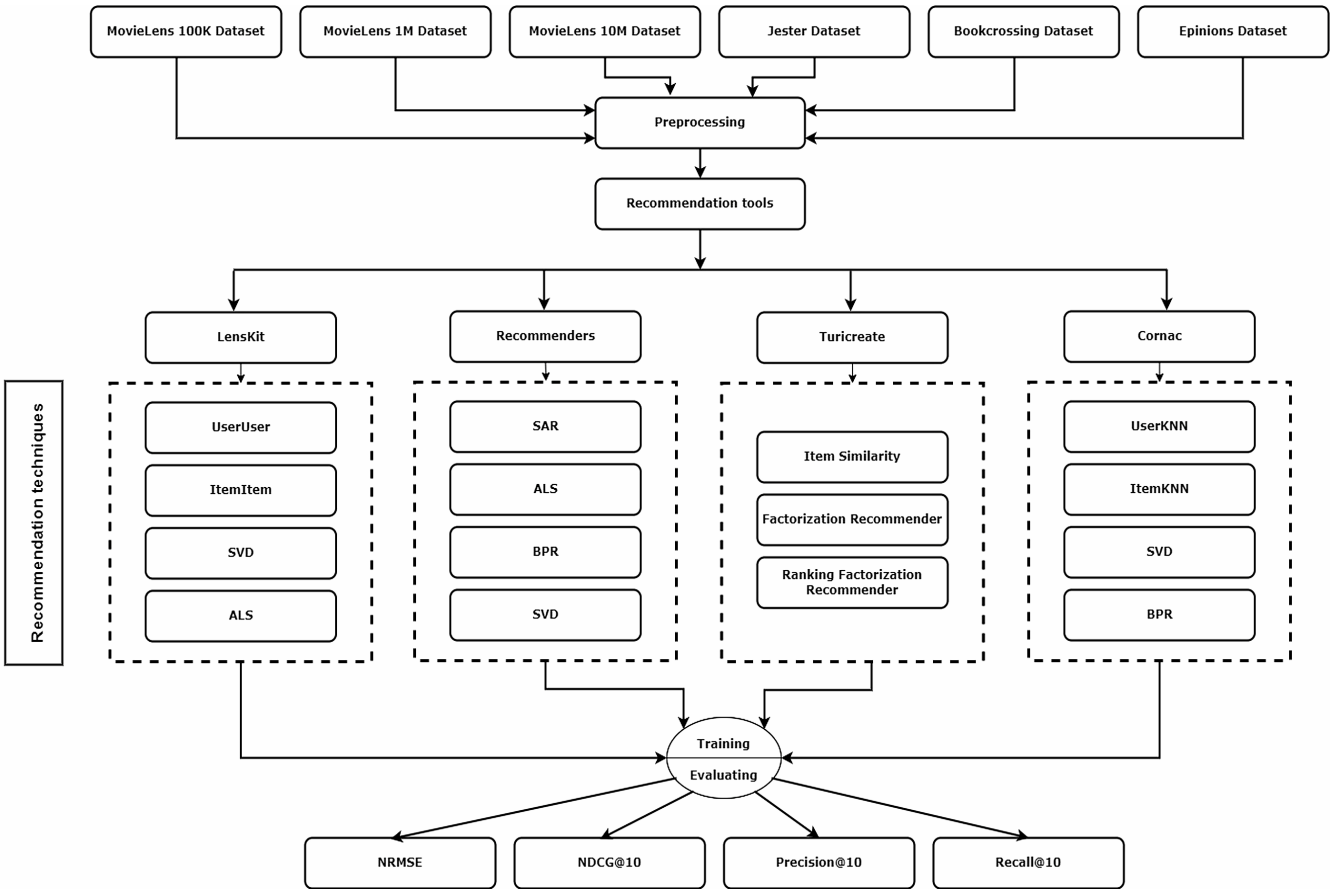

FIGURE 1. The architecture of the comparative analysis of different ML algorithms across various RS tools.

we compare them on datasets with different rating scales. We normalize the total RMSE by dividing overall users' mean RMSE value by the maximum rating minus the minimum rating. We used three top-N evaluation metrics: NDCG@10, Precision@10, and Recall@10, where 10 is the size of the recommendation list. The training time and testing time are also considered.

VIII. RESULTS AND DISCUSSIONS

Figures 2, 3, 4, 5, 6, and 7 show the performance of Lenskit, Recommenders, Turi Create, and Cornac models, based on NDCG@10, Precision@10, Recall@10, and NRMSE, on MovieLens 100K, MovieLens 1M, MovieLens 10M, Jester, Bookcrossing, and Epinions, respectively. The x-axis represents the models, and the y-axis represents the values of NDCG@10, Precision@10, Recall@10, and NRMSE.

In the first place, we can make comparisons between the algorithms of each recommender framework:

Using Lenskit, the ItemItem model performs well based on NRMSE in MovieLens 100K, MovieLens 1M, and Jester datasets. ALS and SVD achieved the lowest NRMSE values for Bookcrossing and Epinions, respectively. However, ALS and SVD perform well in terms of top-N evaluation metrics in all datasets except the Jester dataset, where the UserUser is the best based on Precision and Recall, and the ItemItem is the best based on NDCG.

Using Recommenders software, SVD is the best algorithm in all datasets based on NRMSE, as it always gives the lowest values except for the Jester dataset, where ALS is the best. Concerning the top-N evaluation metrics, BPR is the top performer based on NDCG, Precision, and Recall, followed by SAR as the second-best algorithm for all datasets except the Jester dataset, where SVD is the second-best algorithm, and the Bookcrossing dataset where ALS is the second-best algorithm. One may also notice that BPR gives high values for the ranking metrics, especially on the Jester dataset (0.9465, 0.9170, and 0.5640 for NDCG@10, precision@10, and Recall@10, respectively), which is expected since BPR is designed to predict item ranking.

For Turi Create, we implemented a Factorization Recommender, a Ranking Factorization Recommender, and Item Similarity models using three different similarity metrics: the Jaccard coefficient, Cosine similarity, and Pearson correlation. Since the Factorization Recommender uses MF to predict ratings, we can notice that the Factorization Recommender model is the best at predicting ratings by giving the lowest NRMSE values. The Item similarity models using either Jaccard or Cosine similarity are the best concern-

TABLE 8. Performance of algorithms on MovieLens 100K for each RS tool based on NRMSE, NDCG@10, precision@10, recall@10, and training and testing time in seconds.

Tool	Algorithm	NRMSE	NDCG@10	Precision@10	Recall@10	Train time (s)	Test time (s)
LensKit	UserUser	0.2336	0.0042	0.0062	0.0069	**0.0296**	3.4640
	ItemItem	**0.2241**	0.0361	0.0382	0.0391	3.9009	2.4800
	ALS	0.2277	**0.0830**	**0.0779**	0.0814	0.8168	1.8988
	SVD	0.2341	0.0386	0.0399	0.0424	3.372	**1.5704**
Recommender	SAR	0.2239	0.3864	0.3324	0.1771	**0.3156**	**0.1503**
	ALS	0.2462	0.0083	0.0090	0.0027	1.4482	9.3832
	BPR	NE	**0.4154**	**0.3651**	**0.1878**	1.9911	1.2959
	SVD	**0.2108**	0.1105	0.1004	0.0353	9.4600	14.4215
Turi Create	Factorization	**0.2497**	NE	0.0592	0.0270	2.6672	NE
	Ranking Factorization	0.3252	NE	0.2091	0.1217	6.2567	NE
	Item Similarity (Jaccard)	0.9122	NE	0.2718	**0.1840**	**0.2226**	NE
	Item Similarity (Cosine)	0.8602	NE	**0.2844**	0.1832	0.2367	NE
	Item Similarity (Pearson)	0.2584	NE	0.0008	0.0005	0.3756	NE
Cornac	UserKNN	0.2133	0.0006	0.0007	0.0006	0.1326	6.2750
	ItemKNN	0.2229	0.0141	0.0144	0.0047	0.1827	11.2737
	SVD	**0.2099**	0.0791	0.0753	0.0294	**0.0552**	**1.0132**
	BPR	0.5082	**0.1383**	**0.1282**	**0.0769**	1.1138	1.0180

TABLE 9. Performance of algorithms on MovieLens 1M for each RS tool based on NRMSE, NDCG@10, precision@10, recall@10, and training and testing time in seconds.

Tool	Algorithm	NRMSE	NDCG@10	Precision@10	Recall@10	Train time (s)	Test time (s)
LensKit	UserUser	0.2225	0.0236	0.0312	0.0314	**0.2080**	5.1658
	ItemItem	**0.2117**	0.0348	0.0395	0.0405	9.8073	5.7713
	ALS	0.2157	**0.0707**	**0.0671**	0.0700	3.6052	2.5792
	SVD	0.2128	0.0574	0.0579	0.0593	35.4857	**2.3430**
Recommender	SAR	0.2343	0.3282	0.2976	0.1089	2.9490	**3.1286**
	ALS	0.2241	0.0020	0.0025	0.0004	**2.5379**	16.4152
	BPR	NE	**0.4050**	**0.3721**	**0.1332**	21.6974	19.9647
	SVD	**0.2232**	0.0995	0.0902	0.0249	97.0210	202.7455
Turi Create	Factorization	**0.2285**	NE	0.0816	0.0297	17.0913	NE
	Ranking Factorization	0.2570	NE	0.1718	0.0660	76.8326	NE
	Item Similarity (Jaccard)	0.8952	NE	**0.2939**	**0.1274**	3.9503	NE
	Item Similarity (Cosine)	0.8952	NE	**0.2939**	**0.1274**	**3.8973**	NE
	Item Similarity (Pearson)	0.2467	NE	0.0000	0.0000	7.9121	NE
Cornac	UserKNN	0.2380	0.0003	0.0003	0.0000	3.5349	170.4192
	ItemKNN	0.2473	0.0157	0.0190	0.0031	1.5009	272.2339
	SVD	**0.2276**	0.0505	0.0518	0.0181	**0.5428**	9.8489
	BPR	0.5131	**0.1246**	**0.1168**	**0.0435**	14.5875	**9.8376**

TABLE 10. Performance of algorithms on MovieLens 10M for each RS tool based on NRMSE, NDCG@10, precision@10, recall@10, and training and testing time in seconds.

Tool	Algorithm	NRMSE	NDCG@10	Precision@10	Recall@10	Train time (s)	Test time (s)
LensKit	UserUser	0.1852	0.0039	0.0058	0.0058	**2.0387**	130.2260
	ItemItem	0.1767	0.0138	0.0168	0.0170	37.8046	85.0292
	ALS	0.1823	**0.0591**	**0.0510**	**0.0542**	34.5935	10.5969
	SVD	**0.1742**	0.0437	0.0402	0.0419	573.3138	**10.4667**
Recommender	SAR	0.2161	0.3569	0.3106	**0.1595**	31.5727	**111.8086**
	ALS	0.1817	0.0000	0.0000	0.0000	**5.8999**	164.2052
	BPR	NE	**0.3883**	**0.3527**	0.1444	309.5321	766.0699
	SVD	**0.1808**	0.0853	0.0738	0.0228	993.2828	9647.9206
Turi Create	Factorization	**0.1864**	NE	0.0600	0.0194	182.1749	NE
	Ranking Factorization	0.2055	NE	0.1512	0.0641	777.7044	NE
	Item Similarity (Jaccard)	0.7998	NE	0.2711	0.1588	**56.6016**	NE
	Item Similarity (Cosine)	0.7677	NE	**0.2854**	**0.1614**	60.5991	NE
	Item Similarity (Pearson)	0.2087	NE	0.0000	0.0000	116.2403	NE
Cornac	UserKNN	0.2024	0.0001	0.0001	0.0000	497.1349	33504.9271
	ItemKNN	0.2067	0.0190	0.0210	0.0036	17.0317	20663.9767
	SVD	**0.1931**	0.0483	0.0330	0.0138	**7.6220**	**174.0976**
	BPR	0.3229	**0.1192**	**0.1109**	**0.0562**	197.0230	179.2497

ing ranking recommendations except for the Bookcrossing dataset, where Ranking Factorization ranks first and Item similarity using the Jaccard coefficient ranks second.

Using Cornac, we notice that SVD is the best in predicting ratings, as it gives lower NRMSE values in all datasets except Jester, where ItemKNN is the best.

TABLE 11. Performance of algorithms on Jester for each RS tool based on NRMSE, NDCG@10, precision@10, recall@10, and training and testing time in seconds.

Tool	Algorithm	NRMSE	NDCG@10	Precision@10	Recall@10	Train time (s)	Test time (s)
LensKit	UserUser	0.2066	0.3853	**0.5916**	**0.6045**	**0.2825**	16.4691
	ItemItem	**0.2059**	**0.4857**	0.5549	0.5685	0.9849	7.7198
	ALS	0.2263	0.4172	0.5104	0.5179	9.2545	4.9957
	SVD	0.2230	0.3318	0.4803	0.4863	86.6464	**4.6143**
Recommender	SAR	0.4186	0.0482	0.0643	0.0467	6.0379	**0.4089**
	ALS	**0.2131**	0.5839	0.5706	0.3027	**4.0749**	5.7465
	BPR	NE	**0.9465**	**0.9170**	**0.5640**	26.7691	2.4808
	SVD	0.2396	0.6233	0.5994	0.3262	172.5195	24.0465
Turi Create	Factorization	**0.2098**	NE	0.5430	0.3566	18.9673	NE
	Ranking Factorization	0.2759	NE	0.8202	0.6015	40.6944	NE
	Item Similarity (Jaccard)	0.2634	NE	**0.8820**	**0.6542**	**2.5571**	NE
	Item Similarity (Cosine)	0.2499	NE	0.6495	0.4520	2.7951	NE
	Item Similarity (Pearson)	0.2385	NE	0.5453	0.3717	4.2846	NE
Cornac	UserKNN	0.2017	0.0675	0.0671	0.0862	102.7782	463.2912
	ItemKNN	**0.1995**	0.1315	0.1244	0.1376	**0.7128**	66.0800
	SVD	0.2088	**0.2088**	**0.1684**	**0.1891**	1.2734	**17.5475**
	BPR	0.2524	0.1353	0.1170	0.1387	20.7437	17.7536

TABLE 12. Performance of algorithms on Bookcrossing for each RS tool based on NRMSE, NDCG@10, precision@10, recall@10, and training and testing time in seconds.

Tool	Algorithm	NRMSE	NDCG@10	Precision@10	Recall@10	Train time (s)	Test time (s)
LensKit	UserUser	0.2106	0.0003	0.0007	0.0007	**0.0247**	3.8772
	ItemItem	0.1902	0.0010	0.0013	0.0013	0.5000	2.9667
	ALS	**0.1669**	**0.0054**	**0.0042**	**0.0042**	14.6721	2.3585
	SVD	0.1703	0.0035	**0.0042**	**0.0042**	4.4174	**1.9728**
Recommender	SAR	0.8165	0.0011	0.0010	0.0002	51.3343	**10.3280**
	ALS	0.6552	0.0036	0.0039	0.0008	2.5943	12.4974
	BPR	NE	**0.0338**	**0.0300**	**0.0057**	**2.0758**	13.8254
	SVD	0.3765	0.0024	0.0029	0.0003	9.25354	130.0027
Turi Create	Factorization	**0.1860**	NE	0.0019	0.0002	**2.0461**	NE
	Ranking Factorization	0.5731	NE	**0.0152**	0.0024	3.1429	NE
	Item Similarity (Jaccard)	0.8941	NE	0.0142	**0.0029**	31.9207	NE
	Item Similarity (Cosine)	0.8937	NE	0.0094	0.0020	29.7706	NE
	Item Similarity (Pearson)	0.6936	NE	0.0006	0.0001	28.8523	NE
Cornac	UserKNN	0.1831	0.0014	0.0010	0.0003	**0.0260**	8.4518
	ItemKNN	0.1680	0.0038	0.0039	0.0019	34.7913	457.9463
	SVD	**0.1578**	0.0214	0.0194	0.0076	0.0870	2.2480
	BPR	0.7931	**0.0218**	**0.0197**	**0.0085**	1.3673	**2.2443**

TABLE 13. Performance of algorithms on Epinions for each RS tool based on NRMSE, NDCG@10, precision@10, recall@10, and training and testing time in seconds.

Tool	Algorithm	NRMSE	NDCG@10	Precision@10	Recall@10	Train time (s)	Test time (s)
LensKit	UserUser	0.2885	0.0015	0.0012	0.0012	**0.0334**	4.0252
	ItemItem	0.2774	0.0005	0.0004	0.0004	0.5401	2.9702
	ALS	0.2608	**0.0071**	**0.0073**	**0.0073**	16.0045	2.3730
	SVD	**0.2542**	0.0052	0.0046	0.0046	6.1286	**1.9642**
Recommender	SAR	0.6765	0.0988	0.0948	0.0147	107.0749	14.4924
	ALS	0.6058	0.0035	0.0033	0.0005	**2.9482**	**13.7424**
	BPR	NE	**0.1073**	**0.0994**	**0.0157**	2.9946	23.5971
	SVD	**0.2695**	0.0094	0.0096	0.0016	13.4988	317.1890
Turi Create	Factorization	**0.3145**	NE	0.0017	0.0003	**1.4785**	NE
	Ranking Factorization	0.6368	NE	0.0233	0.0044	4.3023	NE
	Item Similarity (Jaccard)	1.0188	NE	**0.0552**	**0.0107**	31.8254	NE
	Item Similarity (Cosine)	1.0177	NE	0.0456	0.0089	33.0346	NE
	Item Similarity (Pearson)	0.6817	NE	0.0004	0.0001	28.2870	NE
Cornac	UserKNN	0.2694	0.0001	0.0002	0.0000	**0.0258**	15.4136
	ItemKNN	0.2954	0.0019	0.0019	0.0005	38.0062	747.9754
	SVD	**0.2681**	0.0255	0.0240	0.0055	0.1160	3.8693
	BPR	0.7767	**0.0518**	**0.0446**	**0.0118**	1.8834	**3.8455**

Regarding ranking recommendations, BPR consistently ranks first and SVD second in all datasets except Jester, where SVD is the best, and ItemKNN is the second best.

The results of the implementation are summarized in Tables 8, 9, 10, 11, 12, and 13 for MovieLens 100k, MovieLens 1M, MovieLens 10M, Jester, Bookcrossing, and Epinions datasets, respectively.

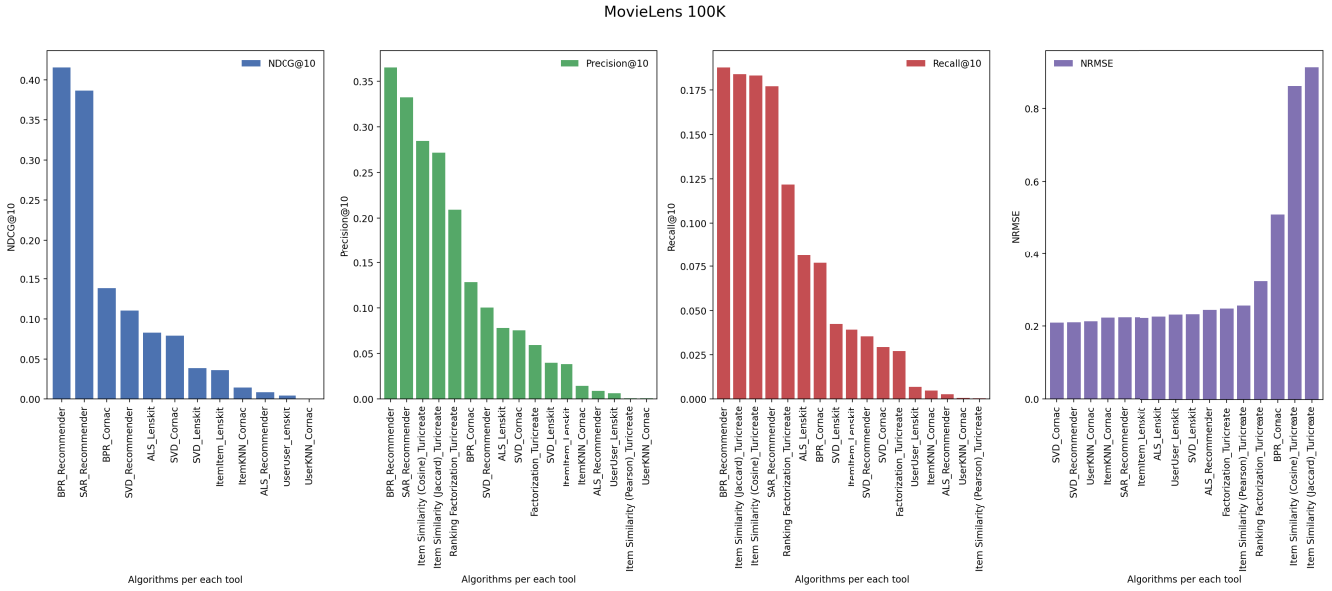

FIGURE 2. Performance of algorithms based on NDCG@10, Precision@10, and Recall@10, and NRMSE on MovieLens 100k.

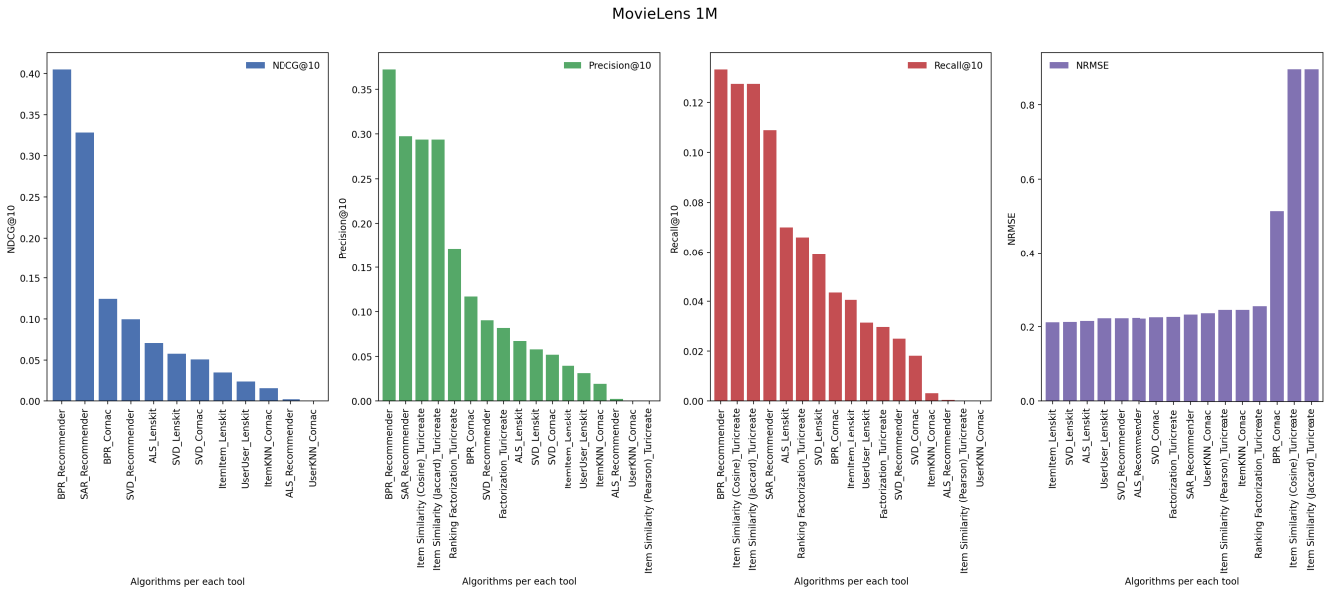

FIGURE 3. Performance of algorithms based on NDCG@10, Precision@10, and Recall@10, and NRMSE on MovieLens 1M.

For each dataset and each recommender tool, the performance metrics for the top-performer algorithm are written in bold, while those for the second top-performer algorithm are underlined. Turi Create does not provide an NDCG metric to evaluate algorithms, and we directly evaluate algorithms after training (testing time is unknown). We cannot also assess BPR (implemented by Recommenders) based on RMSE. To this end, we fill in those missing entries with NE (Not Evaluated).

The results demonstrate that the best algorithms are spread across multiple evaluation metrics. For example, while SVD (Lenskit) performs well on MovieLens 10M and Epinions in terms of NRMSE, no other algorithm performs well based on that metric on more than one dataset. Still, one may notice that MF models: SVD and Factorization Recommender, in the case of Turi Create, give the lowest values of NRMSE, which demonstrates the effectiveness of MF algorithms in predicting the ratings. One reason that makes the MF models effective in predicting the ratings well is its learning procedure that decomposes the user-item interaction matrix into two low-dimensional user and item latent factor matrices. These latent factors can capture complex patterns that influence

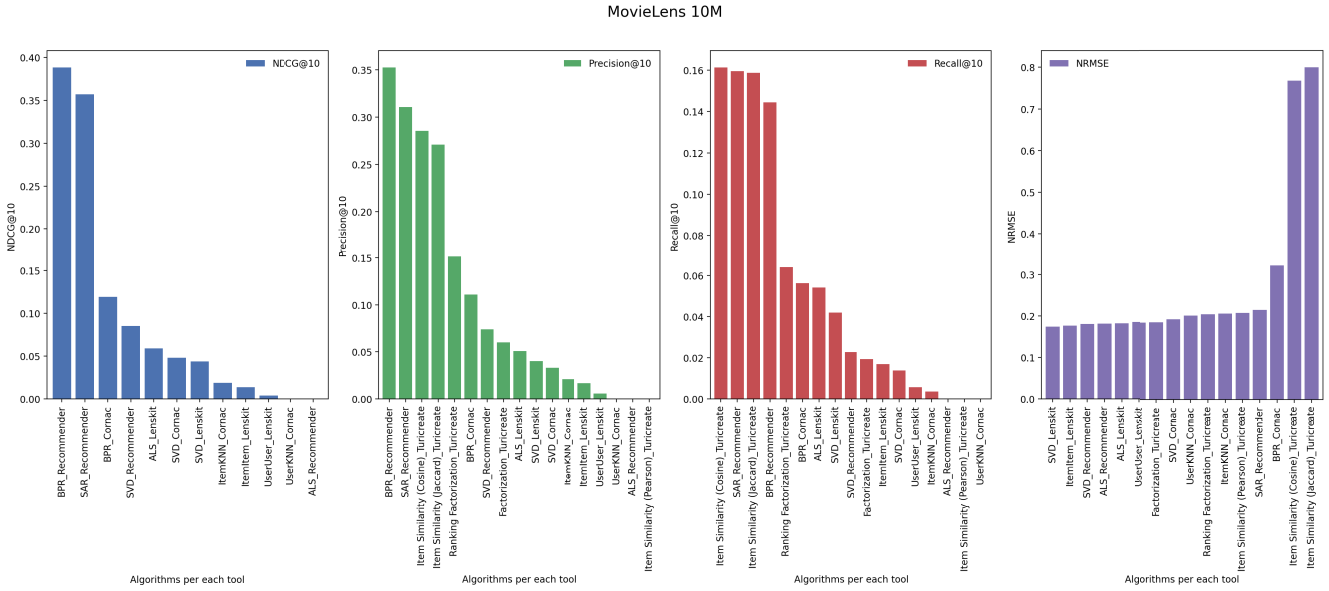

FIGURE 4. Performance of algorithms based on NDCG@10, Precision@10, and Recall@10, and NRMSE on MovieLens 10M.

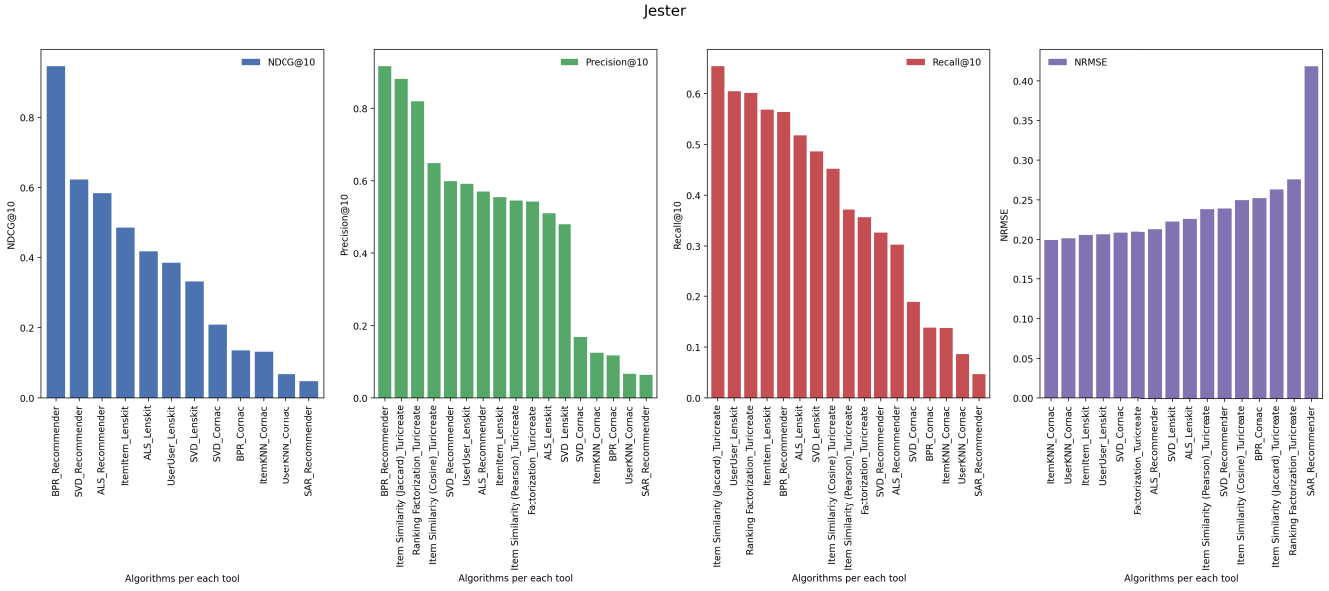

FIGURE 5. Performance of algorithms based on NDCG@10, Precision@10, and Recall@10, and NRMSE on Jester.

the user's preferences, resulting in more accurate rating predictions. As mentioned previously in the paper, MF is adequate for handling sparse matrices; this also allows MF algorithms to be more accurate compared to other algorithms.

On the other hand, when ranking recommendations, the BPR (Recommenders) ranks first as it always gives higher values for NDCG@10 and Precision@10. In contrast, SAR ranks second except for the Jester and Bookcrossing datasets. For Jester, SVD (Recommenders) and ItemItem similarity using the Jaccard coefficient rank second in terms of NDCG@10 and Precision@10, respectively, and BPR (Cornac) ranks second on Bookcrossing. The performance of algorithms based on Recall@10 is distributed across different datasets, but BPR (Recommenders) gives the highest values in MovieLens 100K, MovieLens 1M, and Epinions.

We can also make pairwise comparisons between the algorithms. For example, item-based models often perform better than user-based models, especially in Jester datasets, where the number of unique items is small compared to others since Jester has only 100 unique items. This is because items are simpler than users, although users often have a variety of preferences. It is simpler to identify items with similar features than to find users who share the same preferences.

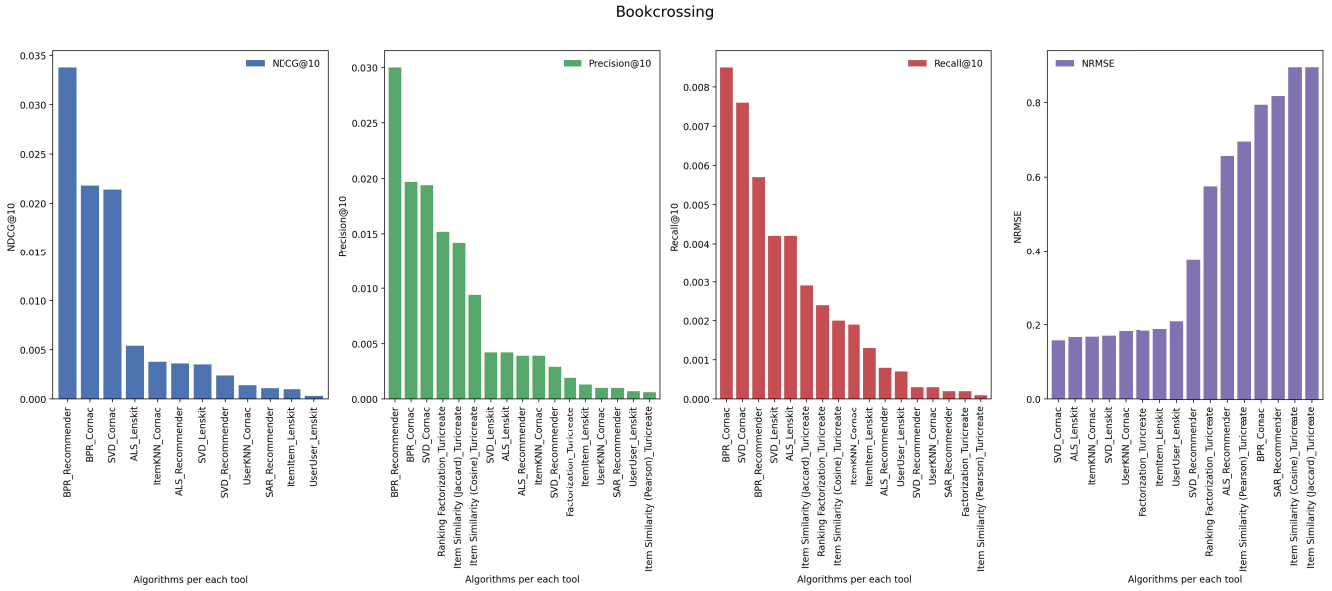

FIGURE 6. Performance of algorithms based on NDCG@10, Precision@10, and Recall@10, and NRMSE on Bookcrossing.

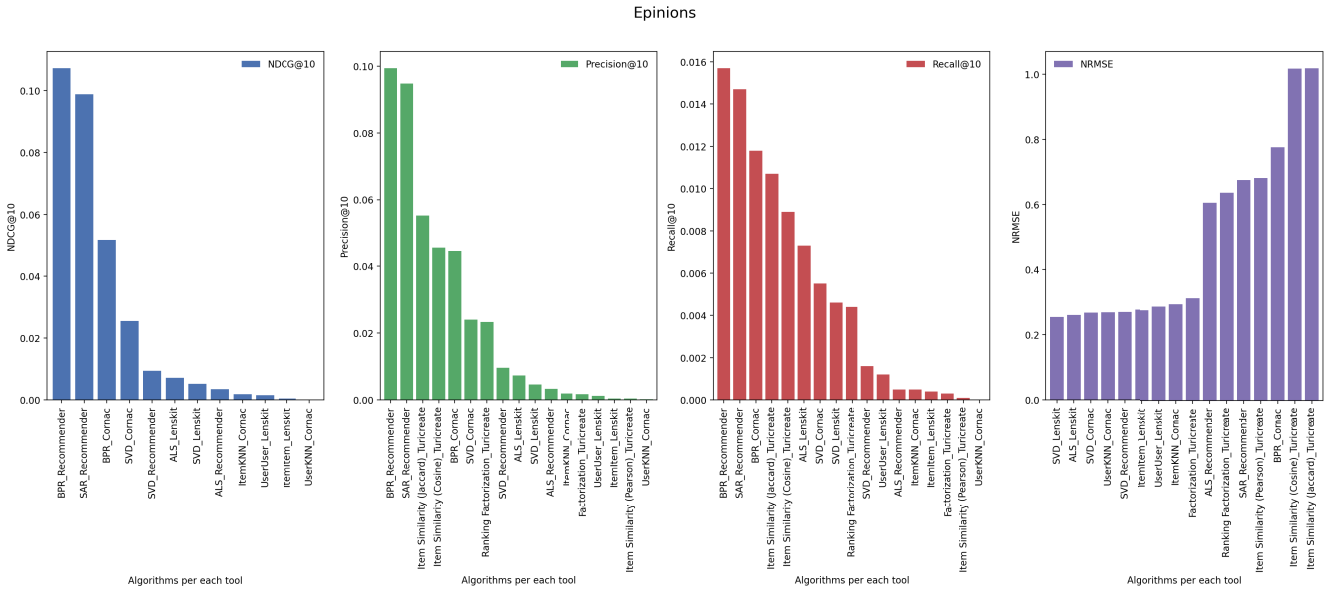

FIGURE 7. Performance of algorithms based on NDCG@10, Precision@10, and Recall@10, and NRMSE on Epinions.

For better visualization of the results, a summary of the best algorithm for each recommender tool on each dataset based on each performance metric is given in Table 14.

In evaluating recommender algorithms across different frameworks, each tool exhibits distinct strengths in various tasks. Figure 8 compares the average rank of algorithms across all datasets and metrics for each tool, where a lower rank indicates better performance. Since some algorithms lack values for specific metrics (e.g., NDCG for Turi Create and NRMSE for BPR), ranks were calculated only for the available data to ensure a fair assessment without penalizing for missing values. Furthermore, precision and recall were combined into a single F1-Score to capture accuracy and completeness in one metric.

In Lenskit, ALS is the most robust for relevance and classification tasks. At the same time, ItemItem excels in accuracy, with SVD offering balanced performance and UserUser underperforming, highlighting Lenskit's alignment with matrix factorization and collaborative filtering approaches. In Recommenders, BPR outperforms others in relevance and classification, while SVD excels in numerical accuracy. SAR provides consistent, general-purpose performance, though ALS shows weaker results in relevance and classification. Turi Create demonstrates that factorization

TABLE 14. Best algorithm for each RS tool on each dataset for each performance metric.

Dataset	Performance Metric	LensKit	Recommenders	Turi Create	Cornac
MovieLens 100K	NRMSE	ItemItem (0.2241)	SVD (0.2108)	Factorization (0.2497)	SVD (0.2099)
	NDCG@10	ALS (0.0830)	BPR (0.4154)		BPR (0.1383)
	Precision@10	ALS (0.0779)	BPR (0.3651)	Item Similarity (Cosine) (0.2844)	BPR (0.1282)
	Recall@10	ALS (0.0814)	BPR (0.1878)	Item Similarity (Jaccard) (0.1840)	BPR (0.0769)
MovieLens 1M	NRMSE	ItemItem (0.2117)	SVD (0.2232)	Factorization (0.2285)	SVD (0.2276)
	NDCG@10	ALS (0.0707)	BPR (0.4050)		BPR (0.1246)
	Precision@10	ALS (0.0671)	BPR (0.3721)	Item Similarity (Jaccard) (0.2939)	BPR (0.1168)
	Recall@10	ALS (0.0700)	BPR (0.1332)	Item Similarity (Jaccard) (0.1274)	BPR (0.0435)
MovieLens 10M	NRMSE	SVD (0.1742)	SVD (0.1808)	Factorization (0.1864)	SVD (0.1931)
	NDCG@10	ALS (0.0591)	BPR (0.3883)		BPR (0.1192)
	Precision@10	ALS (0.0510)	BPR (0.3527)	Item Similarity (Cosine) (0.2854)	BPR (0.1109)
	Recall@10	ALS (0.0542)	SAR (0.1595)	Item Similarity (Cosine) (0.1614)	BPR (0.0562)
Jester	NRMSE	ItemItem (0.2059)	ALS (0.2131)	Factorization (0.2098)	ItemKNN (0.1995)
	NDCG@10	ItemItem (0.4857)	BPR (0.9465)		SVD (0.2088)
	Precision@10	UserUser (0.5916)	BPR (0.9170)	Item Similarity (Jaccard) (0.8820)	SVD (0.1684)
	Recall@10	UserUser (0.6045)	BPR (0.5640)	Item Similarity (Jaccard) (0.6542)	SVD (0.1891)
Bookcrossing	NRMSE	ALS (0.1669)	SVD (0.3765)	Factorization (0.1860)	SVD (0.1578)
	NDCG@10	ALS (0.0054)	BPR (0.0338)		BPR (0.0218)
	Precision@10	ALS (0.0042) SVD (0.0042)	BPR (0.0300)	Ranking Factorization (0.0152)	BPR (0.0197)
	Recall@10	ALS (0.0042) SVD (0.0042)	BPR (0.0057)	Item Similarity (Jaccard) (0.0029)	BPR (0.0085)
Epinions	NRMSE	SVD (0.2542)	SVD (0.2695)	Factorization (0.3145)	SVD (0.2681)
	NDCG@10	ALS (0.0071)	BPR (0.1073)		BPR (0.0518)
	Precision@10	ALS (0.0073)	BPR (0.0994)	Item Similarity (Jaccard) (0.0552)	BPR (0.0446)
	Recall@10	ALS (0.0073)	BPR (0.0157)	Item Similarity (Jaccard) (0.0107)	BPR (0.0118)

TABLE 15. Top 5 algorithms based on average rank over all datasets across each metric.

Rank	NDCG	F1 Score	NRMSE
1	BPR_Recommenders	BPR_Recommenders	ItemItem_LensKit
2	BPR_Cornac	Item Similarity (Jaccard)_Turi Create	SVD_Cornac
3	SVD_Recommenders	Item Similarity (Cosine)_Turi Create	SVD_LensKit
4	ALS_LensKit	Ranking Factorization_Turi Create	ALS_LensKit
5	SAR_Recommenders	BPR_Cornac	UserKNN_Cornac

is optimal for accuracy-based tasks, with item similarity (Jaccard) excelling in classification, while item similarity (Cosine) also performs well but is slightly behind Jaccard. Item Similarity (Pearson) fares the worst, under-performing in classification. This variability among similarity-based methods emphasizes the importance of selecting the right metric for specific tasks. Jaccard is preferable for classification and Ranking Factorization offers a balanced, general-purpose alternative. Finally, in Cornac, SVD emerges as the most reliable algorithm across metrics, with BPR excelling in relevance and classification tasks, although it struggles with numerical accuracy. ItemKNN provides balanced performance, while UserKNN performs less effectively, confirming the superiority of SVD and BPR in Cornac's evaluation context.

To comprehensively compare the performance of all algorithms, Figure 9 presents a consolidated ranking of the algorithms on all datasets based on each metric. Table 15 summarizes the top 5 performing algorithms for each metric.

The results show that for the NDCG metric, BPR _Recommenders achieved the best average rank, followed by BPR_Cornac, SVD_Recommenders, ALS_LensKit, and SAR_Recommenders. These algorithms demonstrated superior ability in capturing recommendation relevance as assessed by NDCG. For F1 Score, which balances precision and recall, BPR_Recommenders again ranked highest, with Item Similarity (Jaccard)_Turi Create, Item Similarity (Cosine)_Turi Create, Ranking Factorization_Turi Create, and BPR_Cornac completing the top five. These results suggest that BPR-based methods and item similarity algo-

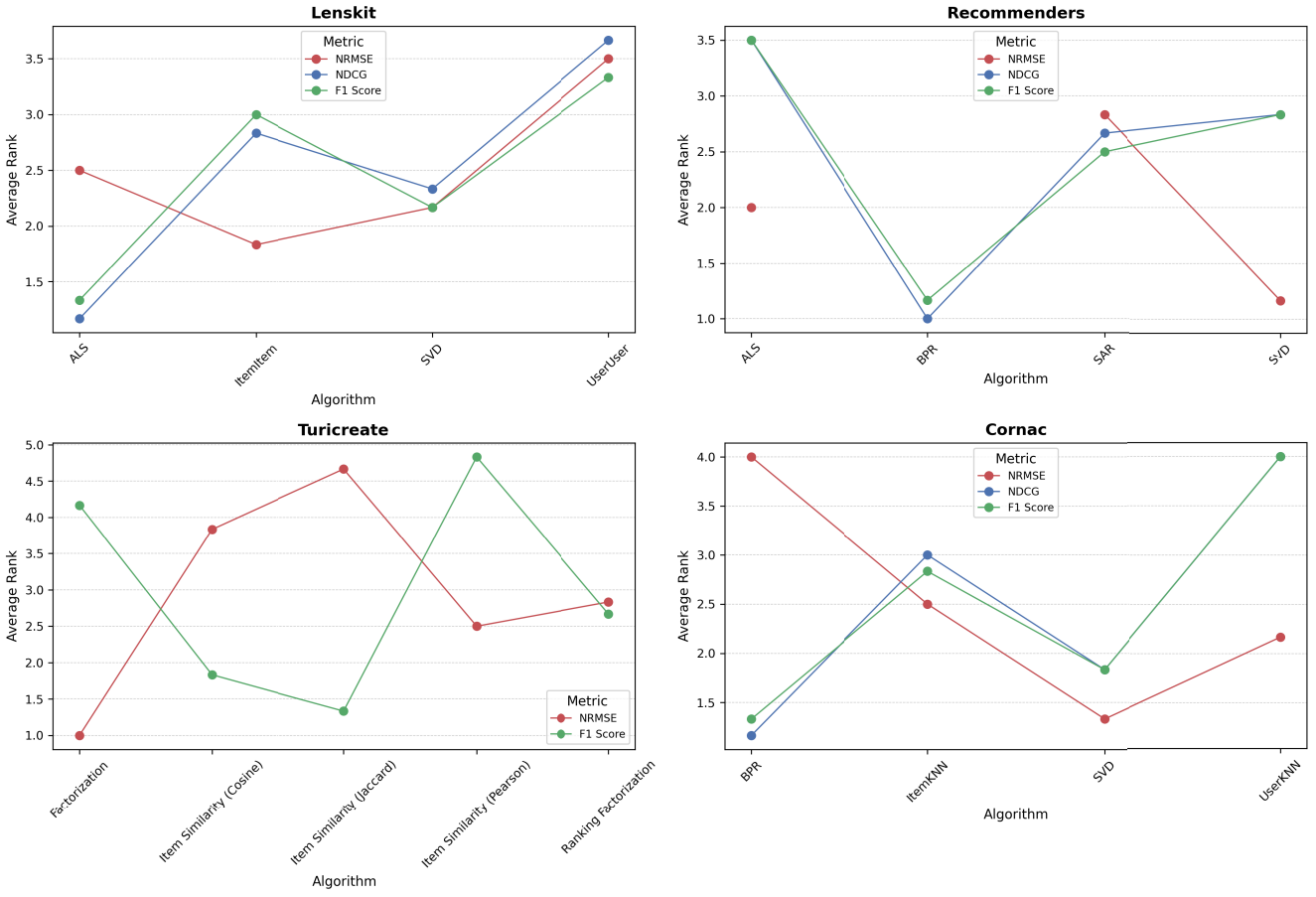

FIGURE 8. The average rank of algorithms over all datasets across each metric for each tool (lower is better).

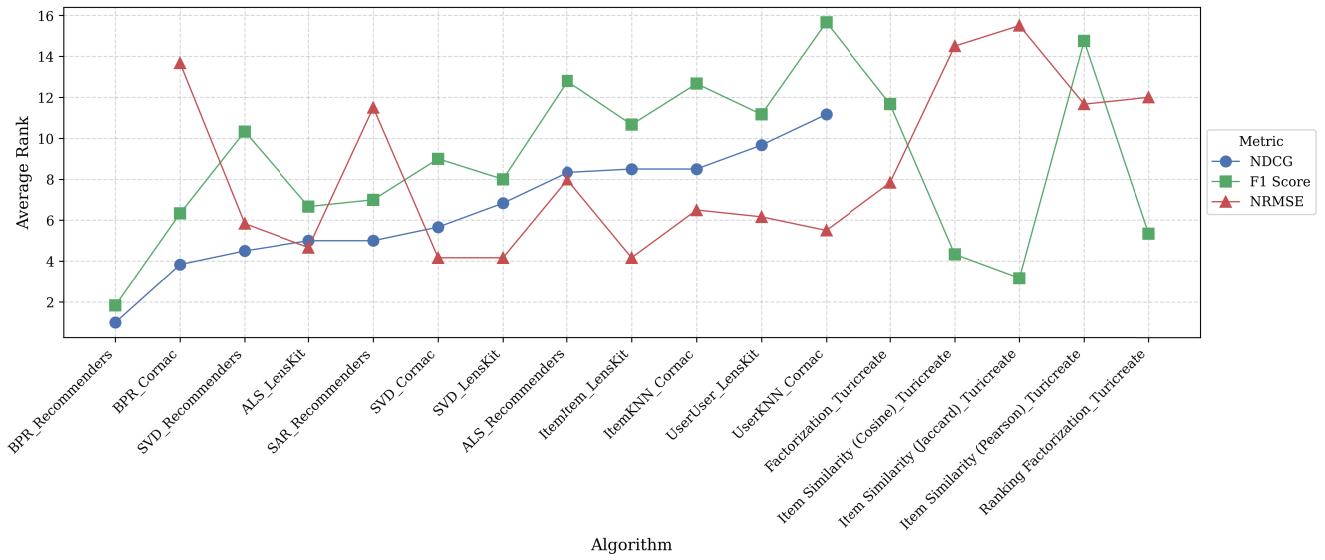

FIGURE 9. The average rank of algorithms across all datasets for each metric (lower is better).

rithms perform well in both relevance and completeness. Lastly, in terms of NRMSE, which measures prediction accuracy, ItemItem_LensKit achieved the best rank, followed by SVD_Cornac, SVD_LensKit, ALS_LensKit, and UserKNN_Cornac. This indicates strong performance in error minimization by neighborhood-based and MF methods.

TABLE 16. Comparison of open source RS software.

Name of software	Github repository	Number of stars	Domain	Active	First date released	Last update date	Techniques	Algorithms	Dataset	Type of feedback	Evaluation metrics	References
Transformers4Rec	https://github.com/NVIDIA-Merlin/Transformers4Rec	1125	E-Commerce	Yes	Apr 14, 2021	Oct 08, 2024	sequential-based recommendation, session-based recommendation, Neural networks-based CF	RNN, UserKNN, ItemKNN, DLRM	Yoochoose	Explicit	Precision, Recall, NDCG, MAP	[64]
Recommenders	https://github.com/Microsoft/Recommenders	19315	Movie, News, Advertising	Yes	Sep 19, 2018	Nov 18, 2024	CF, CBF, Hybrid	DKN, RBM, SASRec, VAE, LightGCN, xDeepFM, A2SVD, SUM,	Criteo, MIND, MovieLens,	Implicit, Explicit	AUC, MAE, MAP, NDCG, Precision, Recall, RMSE	[12]
Lenskit	https://github.com/lenskit/lkpy	270	Movie	Yes	Jun 08, 2018,	Dec 02, 2024	CBF, CF, Item-based CF, User-based CF,	MF, UserKNN, ItemKNN, BiasedMF, MP, Fallback, ImplicitMF, FunkSVD, ALS	MovieLens,	Explicit, Implicit	nDCG, RMSE, MAE, Precision, Recall	[11]
LibRecommender	https://github.com/massquantity/LibRecommender	384	Movie	Yes	Mar 08, 2019	Nov 18, 2024	Hybrid, User-based CF, Item-based CF	SVD++, DeepFM, BPR, SVD,ALS, NCF, BPR, FM, DeepFM, DIN	MovieLens,	Explicit, Implicit	Precision ,Recall, MAP, NDCG, AP	Not mentioned
RecBole	https://github.com/RUCAIBox/RecBole	3462	E-Commerce, Music, Business, Movie, Humor,	Yes	Jun 11, 2020	Sep 05, 2024	KBF, CF, Item-based CF, User-based CF, Neural networks-based CF,	BPR, WARP, AFM, AutoInt, DCN, DeepFM, DSSM, FFM, FM, FNN, FwFM, LR, NFM, PNN, WideDeep, xDeepFM, UserKNN, ItemKNN,	MovieLens, Netflix, LastFM, Douban, Gowalla, Criteo, Jester, Retailrocket, Yelp, Epinions, Amazon	Explicit	AUC, G AUC, HR, MAE, MAP, MRR, NDCG, Precision, RMSE, Recall	[58]
Cornac	https://github.com/PreferredAI/cornac	887	Movie, Research, E-Commerce,	Yes	Jul 17, 2018	Sep 14, 2024	CF, Item-based CF, Used-based CF	GlobalAvg, BPR, MLP, WMF PMF, UserKNN, SoRec, SVD, NMF, MP, MMMF, MF, ItemKNN, CTR, WBPR, HFT, SBPR, EFM, HPF, CDL, VBPR, Skmeans, ConvMF, COE, CDR, VMF, OIBPR, NeuMF, MCF, IBPR, GMF, CVAE, CVAECF, VAECF, PCRL, AMR NARRE, MTER, C2PF, ComparER, CausalRec, BiVAECF	CiteULike, MovieLens, Amazon, Epinions, FilmTrust, Netflix, Tradesy,	Explicit, Implicit	MAE, RMSE, MSE, Recall, Precision, F1-score, NDCG, MRR, AUC	[14]

Continued on next page

TABLE 16. *(Continued.)* **Comparison of open source RS software.**

Name of software	Github repository	Number of stars	Domain	Active	First date released	Last update date	Techniques	Algorithms	Dataset	Type of feedback	Evaluation metrics	References
TorchRec	https://github.com/pytorch/torchrec	1951	Movie, Advertising	Yes	Jul 12, 2021	Dec 01, 2024	Neural networks based CF	DeepFM, DLRM	Criteo, MovieLens	Explicit	NDCG, Precision, Recall, MAP, MRR, AUC	[65]
Implicit	https://github.com/benfred/implicit	3569	Music, Movie	Yes	Apr 17, 2016	Jul 11, 2024	CF, User-based CF, Item-based CF,	UserKNN, ItemKNN, BPRMF, LogMF, ALS	MovieLens, LastFM,	Explicit, Implicit	AUC, NDCG, MAP, Precision	Not Mentioned
Qrec	https://github.com/Coder-Yu/QRec	1590	E-Commerce, Music, Business	Yes	Sep 24, 2016	Dec 26, 2023	User-based CF, Item-based CF	SlopeOne, PMF, SoRec, SVD++, RSTE, SVD, SocialMF, EE, LOCABAL, CUNE-MF, BPR, WRMF, CDAE, DMF, NeuMF, CUNE-BPR, SERec, APR, IF-BPR, CFGAN, NGCF, DiffNet, SimGCL, BUIR, SEPT, SGL, MHCN, ESRF, DHCF, LightGCN, UserKNN, ItemKNN	Ciao, Epinions, LastFM, Yelp, Amazon	Explicit, Implicit	Precision, NDCG, Recall, MAE, RMSE	Not Mentioned
BETA-Rec	https://github.com/beta-team/beta-recsys	163	Movie, E-Commerce, Research, Music, Business, social networking service	Yes	Mar 15, 2020	Aug 28, 2023	CF, Item-based CF, User-based CF	MF, GMF, MLP, CMN, NGCF, LCF LightGCN, VAECF, UserKNN, ItemKNN, MixGCF, UltraGCN, SGL, NARM, SasRec, MARank, Triple2vec, VBCAR, TVBR	MovieLens, LastFM, HetRec, Epinions, Tafeng, Dunnhumby, Instacart, CiteULike, Gowalla, Yelp, Retailrocket, Amazon, Yoochoose	Explicit, Implicit	RMSE, MAE, Recall, NDCG, AUC	[66]
RecList	https://github.com/jacopotagliabue/reclist	459	Movie, Music	Yes	Nov 08, 2021	Aug 09, 2023	CF, USer-based CF, Item-based CF,	UserKNN, ItemKNN, MF	MovieLens, Spotify	Explicit	MRR, Recall, Precision, RMSE	Not Mentioned
CaseRecommender	https://github.com/caserec/CaseRecommender	486	Movie	Yes	Nov 12, 2015	Jan 10, 2024	CBF, CF, Hybrid	BPRMF, ItemKNN, UserKNN, User, Most Popular, Random, MF, SVD, SVD++, GSVD++	MovieLens	Explicit, Implicit	MAE, RMSE, Precision, Recall, NDCG, Precision, Recall, NDCG, MAP	[56]
DeepCTR	https://github.com/shenweichen/deepctr	7583	Movie, Advertising	Yes	Oct 07, 2017	Aug 09, 2024	Neural networks based CF	CCPM, FNN, PNN, WDL, DeepFM, MLR, NFM, AFM, DCN, DCNMix, DIN, DIEN, DSIN, MMOE, BST, AutoInt, ONN, PLE, FGCNN, FiBiNET	MovieLens, Criteo	Explicit, Implicit	AUC, MSE, MAE, RMSE, Precision, Recall,	Not Mentioned

Continued on next page

TABLE 16. *(Continued.)* **Comparison of open source RS software.**

Name of software	Github repository	Number of stars	Domain	Active	First date released	Last update date	Techniques	Algorithms	Dataset	Type of feedback	Evaluation metrics	References
DaisyRec	https://github.com/AmazingDD/daisyRec	549	Movie, Research, E-Commerce, Music	Yes	Dec 02, 2019	Aug 08, 2024	CF	LFM, SLIM, NeuMF, FM, DeepFM, VAE, UserKNN, ItemKNN, WRMF, PureSVD, MP, MF	MovieLens, Netflix, LastFM, Epinions, CiteU-Like, Amazon	Explicit	Precision, Recall, MRR, AP, MAP, DCG, NDCG, AUC, F1,	[67]
Elliot	https://github.com/sisinflab/elliot	275	Movie,	Yes	Oct 23, 2020	Oct 26, 2023	CF, KBF, User-based CF, Item-based CF	SVD++,MF, MostPop, UserKNN,FM, ItemKNN, Random,FFM NFM,PureSVD FunkSVD,SLIM, WRMF,NeuMF, BPRMF,DMF, FISM, NNMF, DeepFM,PMF, AFM, VBPR, ConvMF, GMF,MultiVAE I-AutoR,U-AutoR, CML, SlopeOne,LogMF	MovieLens,	Explicit	NDCG, Precision, AUC, LAUC, GAUC, MAE, MSE, RMSE, ACLT, ARP, MAP, MAR, MRR	[68]
pyRecLab	https://github.com/gasevi/pyreclab	122	Movie	No	Feb 08, 2017	Sep 04, 2020	CF, Item-based CF, User-based CF	UserAverage, ItemAverage, SlopeOne, UserKNN, ItemKNN	MovieLens	Explicit, Implicit	MAP, NDCG, Recall, Precision, AUC, MAE, RMSE	[55]
Polara	https://github.com/evfro/polara	252	Movie, E-Commerce, Music	No	Jul 15, 2016	Jan 16, 2021	Item-based CF, Baselines, Hybrid,	SVD, Random, MP, PMF, HybridSVD, UserKNN, ItemKNN	Movielens, Amazon, Epinions, Netflix, Yahoo! Music	Explicit	Precision, MAP, NDCR, MRR, NDCG	Not Mentioned
LightFM	https://github.com/lyst/lightfm	4777	Movie,	No	Jul 30, 2015	Apr 30, 2023	Hybrid, CBF,CF	BPR,WARP,	MovieLens,	Explicit, Implicit	Precision, AUC, Recall,	[53]
Turi Create	https://github.com/apple/turicreate	11201	Movie	No	Dec 01, 2017	Nov 29, 2023	Item-based CF, CBF	MF, ItemKNN, UserKNN,	Movielens	Explicit	Recall, Precision, MAP, RMSE,	Not Mentioned
Surprise	https://github.com/NicolasHug/Surprise	6419	Movie, Humor	No	Oct 23, 2016	Jun 14, 2024	CBF,CF	SVD, SVD++, NMF, SlopeOne, Centered kNN, Co-Clustering ,Baseline ,Random	Jester, Movie-Lens	Explicit, Implicit	RMSE, MSE, MAE, fcp, kfold	[54]
NeuRec	https://github.com/wubinzzu/NeuRec	1049	Movie, Social net-working service, E-Commerce	No	Feb 25, 2019	Nov 26, 2021	Session-based Recom-mendation, CF	GMF, MLP, NeuMF, NPE, BPRMF, FISM, NAIS, Deep-ICF, ConvNCF, DMF, CDAE, DAE, MultiDAE, MultiVAE, JCA, IRGAN, CFGAN, APR, SpectralCF, NGCF, WRMF, SASRec LightGCN, SBPR, DiffNet, FPMC, HRM, FPM-Cplus, TransRec, GRU4Rec, GRU4RecPlus, SRGNN	MovieLens, Ciao, Gowalla	Explicit, Implicit	Precision, Recall, MAP, NDCG, MRR,	Not Mentioned

Continued on next page

TABLE 16. *(Continued.)* **Comparison of open source RS software.**

Name of software	Github repository	Number of stars	Domain	Active	First date released	Last update date	Techniques	Algorithms	Dataset	Type of feedback	Evaluation metrics	References
DeepRec	https://github.com/cheungdaven/DeepRec	1138	Movie,	No	May 12, 2018	Jun 01, 2022	Matrix Factorization method, Neural networks based CF	biasedSVD, FM, NNMF, NRR, NFM, AFM, BPRMF, CML, CDAE, MLP, GMF, NeuMF, PRME,	MovieLens,	Explicit	RMSE, MAE, NDCG, MRR, Recall, Precision	[36]
OpenRec	https://github.com/ylongqi/openrec	412	Movie, Music, Research, E-Commerce	No	Nov 29, 2017	Feb 19, 2020	CF, CBF, Neural networks based CF	PMF, BPR, WRMF, GMF, DLRM, UCML	CiteULike, Amazon, Criteo, LastFM, Netflix, Tradesy	Explicit, Implicit	NDCG, AUC, Recall, MSE, Precision	[35]
Spotlight	https://github.com/maciejkula/spotlight	2994	Movie, Book rating,	No	Jun 25, 2017	Feb 09, 2020	Session-based recommendations, Matrix factorization method,	CNN, BilinearNet, LSTM,	MovieLens, Goodbooks	Explicit, Implicit	MRR, Recall, Precision, RMSE	not mentioned
Sequeval	https://github.com/D2KLab/sequeval	13	Movie	No	Dec 19, 2017	May 17, 2019	Baseline, CF	Random, MP	MovieLens,	Explicit	Precision, NDCG, Recall, MAE, RMSE,	[69]
TensorRec	https://github.com/jfkirk/tensorrec	1277	Movie,	No	Feb 28, 2017	Feb 14, 2020	CF, CBF,	UserKNN, ItemKNN, SVD, MF	MovieLens,	Explicit, Implicit	RMSE, Precision, Recall, NDCG, DCG, F1-score	Not Mentioned
Rexy	https://github.com/mazdaka/Rexy	61	Not mentioned	No	Nov 12, 2016	May 22, 2021	User-based CF, Item-based CF,	UseKNN, ItemKNN,	Not Mentioned	Explicit	Not Mentioned	Not Mentioned
Devooght	https://github.com/rdevooght/sequence-based-recommendations	370	Movie	No	Oct 23, 2016	Mar 19, 2024	CF, Matrix factorization methods, Neural networks based CF	BPRMF, FISM, FISM, RNN	MovieLens	Explicit	Not Mentioned	Not mentioned
Hermes	https://github.com/lab41/hermes	124	Music, Movie, Humor	No	Sep 01, 2015	Dec 01, 2016	CF, CBF,	ALS, K-means	Jester, MovieLens, LasttFM	Explicit	MAP, MSE, MAE, Precision, NDCG, Recall, RMSE,	Not Mentioned
Crab	https://github.com/muricoca/crab	1180	Movie,	No	Apr 19, 2011	Mar 01, 2012	User-based CF, Item-based CF	UserKNN, ItemKNN, SVD, MF	MovieLens,	Explicit	RMSE, MAE, NMAE, Precision, Recall, F1-score	Not Mentioned

Continued on next page

TABLE 16. *(Continued.)* **Comparison of open source RS software.**

Name of software	Github repository	Number of stars	Domain	Active	First date released	Last update date	Techniques	Algorithms	Dataset	Type of feedback	Evaluation metrics	References
CF4J	https://github.com/ferortega/cf4j	56	Movie, Humor,	Yes	Nov 20, 2017	Oct 19, 2021	KBF, CF, Item-based CF, User-based CF, Neural networks based CF	UserKNN, ItemKNN, MF, GMF, MLP, NCCF, NeuMF, NMF, PMF, BiasedMF, BNMF, DeepMF, DirMF, BeMF,	FilmTrust, Jester, Netflix, Movie-Lens,	Explicit	Recall, Precision, F1-score, MAE, MSE, RMSE, Max, NDCG	[57]
CARSKit	https://github.com/irecsys/CARSKit/	125	Not mentioned	Yes	Jul 09, 2015	Apr 26, 2022	Context-aware recommendation, CF	UserKNN, itemKNN, Slope-One, SVD++, UserAverage, ItemAverage, BiasedMF,	Not Mentioned	Not Mentioned	MAE, RMSE, MPE, Recall, MAP, NDCG, MMR	[70]
EasyRec	https://github.com/alibaba/EasyRec	1795	Not mentioned	Yes	Dec 08, 2020	Nov 15, 2024	CF, Matrix factorization method,	DBMTL, DCN, DeepFM, DLRM, DSSM, ESMM	HDFS files, CSV files	Explicit	MAE, AUC, Precision, Recall	[71]
Mahout	https://github.com/apache/mahout	2143	Movie	Yes	May 23, 2014	Nov 15, 2024	CF, Hybrid, User-based CF, Item-based CF	UserAverage, ItemAverage, UserKNN CF, ItemKNN,	MovieLens,	Not Mentioned	MAE, NMAE, RMSE, Recall, Precision, F1-score	[72]
TagRec	https://github.com/learning-layers/TagRec	124	Movie, Research, Music, Photography, Social bookmarking	No	Apr 02, 2014	Jul 06, 2023	CBF,CF, tag-based recommender, User-based CF	, 3Layers, SimRank, FolkRank LDA, MP, FM, BLL	Flickr, CiteULike, BibSonomy, Delicious, LastFM, MovieLens, Twitter,	Explicit	Recall, Precision, F1-score, MRR, MAP, NDCG	[73]
Oryx	https://github.com/OryxProject/oryx	1787	Movie,	No	Jul 25, 2014	Jul 14, 2021	Matrix factorization methods, Item-based CF, User-based CF,	k-means, ALS, UserKNN, ItemKNN,	MovieLens	Explicit	MAP, Precision, Recall, MAE, RMSE	Not Mentioned
LibRec	https://github.com/guoguibing/librec	3241	Movie, Humor, E-Commerce	No	Jan 09, 2014	Sep 15, 2022	CF, Hybrid,	GlobalAvg, UserAvg, ItemAvg, MostPop, UserKNN, ItemKNN, BiasedMF, BPMF, SocialMF, RegSVD, SVD++, SoRec, BPR, LDA, SlopeOne	FilmTrust, CiaoDVD, MovieLens, Epinions, Flixster, Jester, Douban, Ciao	Explicit, Implicit	MAE, RMSE, MPE, MAP, NDCG, MRR, Precision, Recall, AUC	[52]

Continued on next page

TABLE 16. *(Continued.)* **Comparison of open source RS software.**

Name of software	Github repository	Number of stars	Domain	Active	First date released	Last update date	Techniques	Algorithms	Dataset	Type of feedback	Evaluation metrics	References
RankSys	https://github.com/RankSys/RankSys	273	Movie	No	Jan 17, 2015	Feb 05, 2021	Item-based CF, User-based CF, Matrix factorization methods,	UserKNN, ItemKNN, MF, LDA	MovieLens	Explicit, Implicit	Precision, Recall, NDCG,	Not Mentioned
RiVal	https://github.com/recommenders/rival	151	Movie,	No	Nov 29, 2013	Oct 18, 2017	CF, User-based CF, Item-based CF,	UserKNN, ItemKNN,SVD	MovieLens	Explicit	Recall, Precision, MAE, MAP, NDCG, RMSE,	[74]
PREA	Not found	Not found	Not Mentioned	No	Sep 1, 2012	Sep 1, 2012	CF, Item-based CF, Used-based CF	SVD, NMF, PMF, Bayesian, PMF, Nonlinear PMF, Slope-One, UserAverage, ItemAverage	Not Mentioned	Not Mentioned	RMSE, MAE, NMAE, HLU, NDCG, Kendall's Tau, Spearman	[75]
Duine	Not found	Not found	Movie,	No	Mar 26,2006	Feb 17,2009	Memory-based CF, Item-based CF, User-based CF,	User Average, Item Average, UserKNN, ItemKNN,	MovieLens	Explicit	MAE, NMAE, NDCG, Precision, Recall	[75]
MyMediaLite	https://github.com/zenogantner/MyMediaLite	503	Movie, Humor	No	Aug 03, 2011	Apr 30, 2020	Baselines, Memory-based CF, Matrix Factorization method, CBF, Hybrid	UserAverage, ItemAverage, UserKNN, ItemKNN, Random, SVD++, SLIM, MF, WRMF, BPRMF, LogMF, SoMF, SlopeOne, MostPop	MovieLens, Jester, Netflix	Explicit, Implicit	NMAE, MAE, RMSE, NDCG, HLU, Precision, Recall, F1-score, Precision	[51]

End of table

We can also compare the computation time to train and test the algorithms. Using Lenskit, the UserUser model took less training time, followed by ALS in MovieLens 100k, MovieLens 1M, and MovieLens 10M, while UserUser and ItemItem are the best and second-best, respectively, in Jester, Bookcrossing, and Epinions. SVD and ALS have a shorter time to test data in all datasets. For Turi Create, the Item Similarity models using Jaccard and Cosine similarity metrics are the top for all datasets except Bookcrossing and Epinions; this is due to the large number of unique items in these two datasets as the Item Similarity should compute the similarity between each item. It can also be observed that SVD using Cornac took less training and testing time in most datasets. One can also compare the computation time overall with the recommender tools. Models implemented by Lenskit took less time. The UserUser algorithm ranks first based on training time in all datasets except Epinions. SVD ranks first based on testing time in MovieLens 1M, MovieLens 10M, and Bookcrossing, while SAR ranks first in MovieLens 100K and Jester datasets. Notably, some models have expensive computations to train and test the algorithm. For example, while UserUser implemented by LensKit took only 130.2260 seconds to test the algorithm on MovieLens 10M, the UserKNN implemented in Cornac took 33, 504.9721 seconds to test the algorithm on the same dataset.

IX. CONCLUSION AND FUTURE WORKS

This research highlights the significance of open-source tools in developing recommender systems (RSs). Given the growing need for personalized recommendations, addressing the challenges associated with selecting appropriate software frameworks and machine learning (ML) methods for RSs is crucial. This paper contributes to the field by conducting a comprehensive comparison of 42 open-source RS tools. Based on defined criteria, four of the most suitable tools: Lenskit, Recommenders, Turi Create, and Cornac, were identified for further analysis. Additionally, the paper provides a concise overview of foundational ML algorithms and the most frequently used performance metrics in RS evaluations. Extensive simulations were carried out using diverse algorithms from the top-selected tools, and evaluated across six widely used datasets: MovieLens 100k, 1M, 10M, Jester, BookCrossing, and Epinions. The algorithms' performance was assessed based on NRMSE, NDCG@10, Precision@10, and Recall@10. The results emphasize that algorithm performance is highly dependent on the tool and varies significantly across evaluation metrics. No single algorithm consistently outperformed others across all evaluation metrics and datasets. However, Matrix factorization-based algorithms, such as SVD, demonstrated strong and versatile performance, making them reliable choices for diverse recommendation scenarios. Neighborhood-based methods, such as UserKNN and ItemKNN, exhibited mixed results, excelling in tools like Lenskit but under-performing in others like Cornac. Similarly, Bayesian Personalized Ranking (BPR) showed superior relevance and classification performance in tools like Recommenders and Cornac. Among similarity-based models in Turi Create, Jaccard similarity outperformed others in classification tasks, though variability across metrics underscores the need for metric-specific selection. This study offers valuable insights into the interplay between algorithms, tools, and metrics, providing a practical framework for selecting software and algorithms tailored to specific task requirements.

Further work should focus on developing a structured framework that can provide a practical guidance for selecting the most suitable open-source RS tools based on specific project requirements. This framework needs to examine key factors such as dataset size, computational efficiency, scalability, and deployment constraints more closely to support informed decision-making. By expanding the scope of analysis, the proposed framework could usefully explore a more systematic and comprehensive approach to evaluating and selecting RS tools. Such an approach would greatly help researchers and practitioners make well-informed choices when implementing RS solutions tailored to their specific needs and constraints.

APPENDIX
COMPARISON OF OPEN-SOURCE RECOMMENDER SYSTEM SOFTWARE

See Table 16.

REFERENCES

[1] L. Mekouar, Y. Iraqi, and I. Damaj, "A global user profile framework for effective recommender systems," *Multimedia Tools Appl.*, vol. 83, no. 17, pp. 50711–50731, Nov. 2023.

[2] L. Mekouar, Y. Iraqi, and R. Boutaba, "Personalized recommendations in peer-to-peer systems," in *Proc. IEEE 17th Workshop Enabling Technol., Infrastruct. Collaborative Enterprises*, Los Alamitos, CA, USA, Jun. 2008, pp. 99–104. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/WETICE.2008.45

[3] L. Mekouar, Y. Iraqi, and R. Boutaba, "A recommender scheme for peer-to-peer systems," in *Proc. Int. Symp. Appl. Internet*, Jul. 2008, pp. 197–200.

[4] L. Mekouar, Y. Iraqi, I. Damaj, and T. Naous, "A survey on blockchain-based recommender systems: Integration architecture and taxonomy," *Comput. Commun.*, vol. 187, pp. 1–19, Apr. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422000342

[5] S. Choukry, Y. Iraqi, and L. Mekouar, "An efficient rating system using blockchain for recommender systems," in *Proc. IEEE Int. Conf. Artif. Intell., Blockchain, Internet Things (AIBThings)*, Sep. 2023, pp. 1–5.

[6] M. Jangid and R. Kumar, "Deep learning approaches to address cold start and long tail challenges in recommendation systems: A systematic review," *Multimedia Tools Appl.*, vol. 2024, pp. 1–33, Oct. 2024.

[7] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang, and Q. Li, "Recommender systems in the era of large language models (LLMs)," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 11, pp. 6889–6907, Nov. 2024.

[8] L. Wu, P. Cui, J. Pei, L. Zhao, and X. Guo, "Graph neural networks: Foundation, frontiers and applications," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 4840–4841.

[9] W. Yuan, H. Wang, X. Yu, N. Liu, and Z. Li, "Attention-based context-aware sequential recommendation model," *Inf. Sci.*, vol. 510, pp. 122–134, Feb. 2020.

[10] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, H. Wang, and J. Zhang, "Chat-REC: Towards interactive and explainable LLMs-augmented recommender system," 2023, *arXiv:2303.14524*.

[11] M. D. Ekstrand, "LensKit for Python: Next-generation software for recommender systems experiments," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Oct. 2020, pp. 2999–3006, doi: 10.1145/3340531.3412778.

[12] A. Argyriou, M. González-Fierro, and L. Zhang, "Microsoft recommenders: Best practices for production-ready recommendation systems," in *Companion Proc. Web Conf.*, New York, NY, USA, Apr. 2020, pp. 50–51, doi: 10.1145/3366424.3382692.

[13] Apple. (2022). *Apple/turicreate: Turi Create Simplifies the Development of Custom Machine Learning Models*. Accessed: Mar. 10, 2022. [Online]. Available: https://github.com/apple/turicreate

[14] A. Salah, Q. Truong, and H. W. Lauw, "Cornac: A comparative framework for multimodal recommender systems," *J. Mach. Learn. Res.*, vol. 21, no. 95, pp. 1–5, Jan. 2020.

[15] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015, doi: 10.1145/2827872.

[16] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, pp. 133–151, Jul. 2001.

[17] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proc. 14th Int. Conf. World Wide Web*, 2005, pp. 22–32.

[18] R. Hamedani, I. Ali, J. Hong, and S.-W. Kim, "TrustRec: An effective approach to exploit implicit trust and distrust relationships along with explicitones for accurate recommendations," *Comput. Sci. Inf. Syst.*, vol. 18, no. 1, pp. 93–114, 2021.

[19] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi, "Evaluating collaborative filtering recommender algorithms: A survey," *IEEE Access*, vol. 6, pp. 74003–74024, 2018.

[20] H. Ko, S. Lee, Y. Park, and A. Choi, "A survey of recommendation systems: Recommendation models, techniques, and application fields," *Electronics*, vol. 11, no. 1, p. 141, Jan. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/1/141

[21] B. B. Sinha and R. Dhanalakshmi, "Evolution of recommender system over the time," *Soft Comput.*, vol. 23, no. 23, pp. 12169–12188, Jun. 2019, doi: 10.1007/s00500-019-04143-8.

[22] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Syst. Appl.*, vol. 97, pp. 205–227, May 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417308333

[23] M. T. Alam, S. Ubaid, Shakil, S. S. Sohail, M. Nadeem, S. Hussain, and J. Siddiqui, "Comparative analysis of machine learning based filtering techniques using MovieLens dataset," *Proc. Comput. Sci.*, vol. 194, pp. 210–217, Jan. 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921021165

[24] MS. Dataset. (2022). *Last.fm Dataset*. Accessed: Mar. 10, 2022. [Online]. Available: http://millionsongdataset.com/lastfm/

[25] M. Nilashi, O. Ibrahim, and K. Bagherifard, "A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques," *Expert Syst. Appl.*, vol. 92, pp. 507–520, Feb. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417306577

[26] (2022). *IMDB*. Accessed: Mar. 10, 2022. [Online]. Available: https://www.imdb.com/interfaces/

[27] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, Geneva, Switzerland, Apr. 2017, pp. 173–182, doi: 10.1145/3038912.3052569.

[28] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," in *Recommender Systems Handbook*. Cham, Switzerland: Springer, 2015, pp. 1–34, doi: 10.1007/978-1-4899-7637-6_1.

[29] M. Jahrer, A. Töscher, and R. Legenstein, "Combining predictions for accurate recommender systems," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Jul. 2010, pp. 693–702, doi: 10.1145/1835804.1835893.

[30] A. S. Lampropoulos and G. A. Tsihrintzis, *Machine Learning Paradigms*. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-19135-5.

[31] Z. Xia, Y. Dong, and G. Xing, "Support vector machines for collaborative filtering," in *Proc. 44th Annu. Southeast Regional Conf.*, New York, NY, USA, Mar. 2006, pp. 169–174, doi: 10.1145/1185448.1185487.

[32] P. Valdiviezo-Diaz, F. Ortega, E. Cobos, and R. Lara-Cabrera, "A collaborative filtering approach based on Naïve Bayes classifier," *IEEE Access*, vol. 7, pp. 108581–108592, 2019.

[33] C. C. Aggarwal, *Recommender Systems*. Cham, Switzerland: Springer, 2016, doi: 10.1007/978-3-319-29659-3.

[34] D. Lemire and A. Maclachlan, "Slope one predictors online rating-based collaborative filtering," in *Proc. SIAM Int. Conf. Data Mining*, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics, Jan. 2005, pp. 471–475. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972757.43

[35] L. Yang, E. Bagdasaryan, J. Gruenstein, C.-K. Hsieh, and D. Estrin, "OpenRec: A modular framework for extensible and adaptable recommendation algorithms," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, New York, NY, USA, Feb. 2018, pp. 664–672, doi: 10.1145/3159652.3159681.

[36] S. Zhang, Y. Tay, L. Yao, B. Wu, and A. Sun, "DeepRec: An open-source toolkit for deep learning based recommendation," 2019, *arXiv:1905.10536*.

[37] C. Panagiotakis, H. Papadakis, A. Papagrigoriou, and P. Fragopoulou, "Improving recommender systems via a dual training error based correction approach," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115386.

[38] L. Huang, M. Fu, F. Li, H. Qu, Y. Liu, and W. Chen, "A deep reinforcement learning based long-term recommender system," *Knowledge-Based Syst.*, vol. 213, Feb. 2021, Art. no. 106706.

[39] Z. Zamanzadeh Darban and M. H. Valipour, "GHRS: Graph-based hybrid recommendation system with application to movie recommendation," *Expert Syst. Appl.*, vol. 200, Aug. 2022, Art. no. 116850.

[40] Y. Li, K. Liu, R. Satapathy, S. Wang, and E. Cambria, "Recent developments in recommender systems: A survey," *IEEE Comput. Intell. Mag.*, vol. 19, no. 2, pp. 78–95, Jan. 2023.

[41] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A core of semantic knowledge," in *Proc. 16th Int. Conf. World Wide Web*, May 2007, pp. 697–706.

[42] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[43] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2008, pp. 1247–1250.

[44] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, Dec. 2013, pp. 1–9.

[45] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proc. AAAI Conf. Artif. Intell.*, vol. 28, Jun. 2014, pp. 1–8.

[46] Y. Deng, "Recommender systems based on graph embedding techniques: A review," *IEEE Access*, vol. 10, pp. 51587–51633, 2022.

[47] C. Huang, H. Xu, Y. Xu, P. Dai, L. Xia, M. Lu, L. Bo, H. Xing, X. Lai, and Y. Ye, "Knowledge-aware coupled graph neural network for social recommendation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, May 2021, pp. 4115–4122.

[48] X. Wang, T. Huang, D. Wang, Y. Yuan, Z. Liu, X. He, and T.-S. Chua, "Learning intents behind interactions with knowledge graph for recommendation," in *Proc. Web Conf.*, Apr. 2021, pp. 878–887.

[49] Z. Chu, H. Hao, X. Ouyang, S. Wang, Y. Wang, Y. Shen, J. Gu, Q. Cui, L. Li, S. Xue, J. Y. Zhang, and S. Li, "Leveraging large language models for pre-trained recommender systems," 2023, *arXiv:2308.10837*.

[50] P. M. Alamdari, N. J. Navimipour, M. Hosseinzadeh, A. A. Safaei, and A. Darwesh, "A systematic study on the recommender systems in the e-commerce," *IEEE Access*, vol. 8, pp. 115694–115716, 2020.

[51] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "MyMediaLite: A free recommender system library," in *Proc. 5th ACM Conf. Recommender Syst.*, Oct. 2011, pp. 305–308.

[52] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith, "Librec: A Java library for recommender systems," in *Proc. Umap Workshops*, vol. 1388, Jan. 2015, pp. 1–4.

[53] M. Kula, "Metadata embeddings for user and item cold-start recommendations," 2015, *arXiv:1507.08439*.

[54] N. Hug, "Surprise: A Python library for recommender systems," *J. Open Source Softw.*, vol. 5, no. 52, p. 2174, Aug. 2020.

[55] G. Sepulveda, V. Dominguez, and D. Parra, "PyRecLab: A software library for quick prototyping of recommender systems," 2017, *arXiv:1706.06291*.

[56] A. da Costa, E. Fressato, F. Neto, M. Manzato, and R. Campello, "Case recommender: A flexible and extensible Python framework for recommender systems," in *Proc. 12th ACM Conf. Recommender Syst.*, Sep. 2018, pp. 494–495.

[57] F. Ortega, B. Zhu, J. Bobadilla, and A. Hernando, "CF4J: Collaborative filtering for Java," *Knowl.-Based Syst.*, vol. 152, pp. 94–99, Jul. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705118301734

[58] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, and J.-R. Wen, "RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2021, pp. 4653–4664.

[59] A. Nawrocka, A. Kot, and M. Nawrocki, "Application of machine learning in recommendation systems," in *Proc. 19th Int. Carpathian Control Conf. (ICCC)*, May 2018, pp. 328–331.

[60] S. Graham, J.-K. Min, and T. Wu, "Microsoft recommenders: Tools to accelerate developing recommender systems," in *Proc. 13th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2019, pp. 542–543, doi: 10.1145/3298689.3346967.

[61] Lenskit. (2022). *Lenskit/lkpy: Python Recommendation Toolkit*. Accessed: Mar. 10, 2022. [Online]. Available: https://github.com/lenskit/lkpy

[62] Microsoft. (2022). *Microsoft/recommenders: Best Practices on Recommendation Systems*. Accessed: Mar. 10, 2022. [Online]. Available: https://github.com/microsoft/recommenders

[63] PreferredAI. (2022). *Preferredai/cornac: A Comparative Framework for Multimodal Recommender Systems*. Accessed: Mar. 10, 2022. [Online]. Available: https://github.com/PreferredAI/cornac

[64] G. de Souza Pereira Moreira, S. Rabhi, J. M. Lee, R. Ak, and E. Oldridge, "Transformers4Rec: Bridging the gap between NLP and sequential / session-based recommendation," in *Proc. 15th ACM Conf. Recommender Syst.*, New York, NY, USA, 2021, pp. 143–153, doi: 10.1145/3460231.3474255.

[65] D. Ivchenko, D. Van Der Staay, C. Taylor, X. Liu, W. Feng, R. Kindi, A. Sudarshan, and S. Sefati, "TorchRec: A PyTorch domain library for recommendation systems," in *Proc. 16th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2022, pp. 482–483, doi: 10.1145/3523227.3547387.

[66] Z. Meng, R. McCreadie, C. Macdonald, I. Ounis, S. Liu, Y. Wu, X. Wang, S. Liang, Y. Liang, G. Zeng, J. Liang, and Q. Zhang, "BETA-Rec: Build, evaluate and tune automated recommender systems," in *Proc. 14th ACM Conf. Recommender Syst.*, New York, NY, USA, 2020, pp. 588–590, doi: 10.1145/3383313.3411524.

[67] Z. Sun, H. Fang, J. Yang, X. Qu, H. Liu, D. Yu, Y.-S. Ong, and J. Zhang, "DaisyRec 2.0: Benchmarking recommendation for rigorous evaluation," 2022, *arXiv:2206.10848*.

[68] V. W. Anelli, A. Bellogin, A. Ferrara, D. Malitesta, F. A. Merra, C. Pomo, F. M. Donini, and T. Di Noia, "Elliot: A comprehensive and rigorous framework for reproducible recommender systems evaluation," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Jul. 2021, pp. 2405–2414.

[69] D. Monti, E. Palumbo, G. Rizzo, and M. Morisio, "Sequeval: A framework to assess and benchmark sequence-based recommender systems," 2018, *arXiv:1810.04956*.

[70] Y. Zheng, B. Mobasher, and R. Burke, "CARSKit: A java-based context-aware recommendation engine," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Nov. 2015, pp. 1668–1671.

[71] M. Cheng, Y. Gao, G. Liu, H. Jin, and X. Zhang, "EasyRec: An easy-to-use, extendable and efficient framework for building industrial recommendation systems," 2022, *arXiv:2209.12766*.

[72] J. P. Verma, B. Patel, and A. Patel, "Big data analysis: Recommendation system with Hadoop framework," in *Proc. IEEE Int. Conf. Comput. Intell. Commun. Technol.*, Feb. 2015, pp. 92–97.

[73] D. Kowald, S. Kopeinik, and E. Lex, "The TagRec framework as a toolkit for the development of tag-based recommender systems," in *Proc. Adjunct Publication 25th Conf. User Model., Adaptation Personalization*, New York, NY, USA, Jul. 2017, pp. 23–28, doi: 10.1145/3099023.3099069.

[74] A. Said and A. Bellogín, "Rival: A toolkit to foster reproducibility in recommender system evaluation," in *Proc. 8th ACM Conf. Recommender Syst.*, New York, NY, USA, Oct. 2014, pp. 371–372, doi: 10.1145/2645710.2645712.

[75] J. Lee, M. Sun, and G. Lebanon, "PREA: Personalized recommendation algorithms toolkit," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2699–2703, Jan. 2012.

AYOUB AKHADAM received the B.Sc. degree in computer science and the M.Sc. degree in data science from the Faculty of Science and Techniques, Cadi Ayyad University, Marrakech, Morocco, in 2020 and 2022, respectively. He is currently pursuing the Ph.D. degree with the TICLab, International University of Rabat (UIR), Morocco. His research interests include data science, knowledge graphs, and recommender systems.

OUMAYMA KBIBCHI received the Engineering Diploma degree in computer science from the National Higher School of Computer Science and Systems Analysis (ENSIAS), Rabat, Morocco, in 2022. She is currently a Business Intelligence Consultant in Morocco.

LOUBNA MEKOUAR (Senior Member, IEEE) received the M.Sc. degree in computer science from the University of Montreal, Canada, and the Ph.D. degree in computer science from the University of Waterloo, Canada. She is currently an Assistant Professor with the School of Computer Science, Mohammed VI Polytechnic University (UM6P), Morocco. Before joining UM6P, she was an Assistant Professor with the College of Technological Innovation, Zayed University, United Arab Emirates. Her research interests include trust and reputation management, recommender systems, and computing education. She is a fellow of the Higher Education Academy (AdvanceHE, U.K.).

YOUSSEF IRAQI (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Montreal, Canada, in 2000 and 2003, respectively. He is currently an Associate Professor with the School of Computer Science, Mohammed VI Polytechnic University, Morocco. Before that, he was with the Department of Electrical Engineering and Computer Science, Khalifa University (KU), United Arab Emirates, for 12 years. Before joining KU, he was the Chair of the Department of Computer Science, Dhofar University, Oman, for four years. From 2004 to 2005, he was a Research Assistant Professor with the David R. Cheriton School of Computer Science, University of Waterloo, Canada. He has published over 130 research papers in international journals and refereed conference proceedings. His research interests include resource management in wireless networks, blockchain, trust and reputation management, cloud computing, and stylometry. In 2008, he received the IEEE Communications Society Fred W. Ellersick Paper Award in the field of communications systems. He is on many technical program committees of international conferences and is frequently approached for his expertise by international journals in his field.