Algorithmes basés sur les données pour le comportement individuel et collectif des utilisateurs

Nassim Bouarour

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Informatique
Unité de recherche : Laboratoire d'Informatique de Grenoble

Algorithmes basés sur les données pour le comportement individuel et collectif des utilisateurs

data-driven algorithms for individual and collective user behavior

Présentée par :

Nassim BOUAROUR

Direction de thèse :

Sihem AMER-YAHIA DIRECTRICE DE RECHERCHE, CNRS DELEGATION ALPES	Directrice de thèse
Idir BENOUARET FLORALIS	Co-encadrant de thèse
Paolo FRASCA Chargé de Recherches, CNRS	Co-encadrant de thèse

Rapporteurs :

ALEXANDRE TERMIER
PROFESSEUR DES UNIVERSITES, UNIVERSITE DE RENNES
MELANIE HERSCHEL
PROFESSEURE, UNIVERSITÄT STUTTGART

Thèse soutenue publiquement le **13 décembre 2023**, devant le jury composé de :

CLAUDIA RONCANCIO, PROFESSEURE DES UNIVERSITES, GRENOBLE INP	Présidente
SIHEM AMER-YAHIA, DIRECTRICE DE RECHERCHE, CNRS DELEGATION ALPES	Directrice de thèse
ALEXANDRE TERMIER, PROFESSEUR DES UNIVERSITES, UNIVERSITE DE RENNES	Rapporteur
MELANIE HERSCHEL, PROFESSEURE, UNIVERSITÄT STUTTGART	Rapporteure
AMEL BOUZEGHOUB, PROFESSEURE DES UNIVERSITES, TELECOM SUDPARIS	Examinatrice
REYNOLD CK CHENG, PROFESSEUR, THE UNIVERSITY OF HONG KONG	Examinateur

Invités :

PAOLO FRASCA
DIRECTEUR DE RECHERCHE, CNRS DELEGATION ALPES
IDIR BENOUARET
DOCTEUR EN SCIENCES,

Abstract

User data is becoming increasingly available in multiple domains ranging from e-commerce platforms to social media networks. It includes demographics (e.g., age, gender, location, etc.) and user activities (e.g., browsing habits, purchase history, rating records, etc.). The analysis of this data is appealing as it helps companies to enhance their business, understand users' behavior, reduce their churn, and attract new customers. With the advancement of technology, many data-driven and data-informed tools were developed to understand the preferences of users and extract valuable insights from the collected data.

In this thesis, we propose to study distinctly both the individual and collective behavior of users. We first aim to examine users individually as each one is unique, and their actions and interactions may vary significantly from one to another. We leverage recommender systems that analyze the behavior of users at a low level of granularity which allows for personalized experiences that can better meet users' expectations. More precisely, we leverage dynamic recommenders to infer the states where the users might be and capture their constant evolution over time. Following this, we first extend the standard recommenders by incorporating users' states and profiles within a static environment based on a meta-learning methodology. Then, we explore more realistic contexts where the environment is dynamic. We explore three real-world applications: Educational test recommendation, SQL query recommendation, and diverse session recommendation. Within each application, we define the behavior of users with many dimensions to avoid overspecialization and filter-bubble and propose several solutions based on Multi-armed bandits, and Reinforcement Learning.

In addition to their unique behavior, users with the same characteristics (e.g., demographics) may exhibit the same global trends. Hence, we aim to extract these insights, seek to analyze users' collective behavior and discover the relationships between the different user groups and the subset of items. For the purpose of reducing false discoveries, we rely on hypothesis testing to produce significant and statistically sound insights. We also optimize for coverage to explore all users' groups and avoid analyzing a small subset of them. We design novel solutions based on standard multiples hypothesis testing corrections as well as α-investing.

In this thesis, we evaluate our solutions using an extensive set of experiments both for quality and performance. We also conduct a comparative analysis with existing approaches or state-of-the-art approaches to demonstrate the effectiveness of our solutions.

Résumé

Les données des utilisateurs deviennent de plus en plus disponibles dans plusieurs domaines, allant des plateformes de commerce électronique aux réseaux sociaux. Elles incluent des informations démographiques (par exemple, l'âge, le sexe, la localisation, etc.) et des activités des utilisateurs (par exemple, les habitudes de navigation, l'historique des achats et des notations, etc.). L'analyse de ces données est attrayante car elle aide les entreprises à améliorer leurs activités, comprendre le comportement des utilisateurs, réduire le taux de désabonnement et attirer de nouveaux clients. Avec l'avancement de la technologie, de nombreux outils axés sur les données ont été développés pour comprendre les préférences des utilisateurs et extraire des informations précieuses à partir des données collectées.

Dans cette thèse, nous proposons d'étudier distinctement le comportement individuel et collectif des utilisateurs. Nous examinons d'abord individuellement les utilisateurs. La raison est que chaque utilisateur est unique, et leurs actions et interactions peuvent varier considérablement d'un individu à un autre. Nous nous appuyons sur des systèmes de recommandation qui analysent le comportement des utilisateurs à un niveau de granularité fin. Ceci permet des expériences personnalisées répondant au mieux aux attentes des utilisateurs. Plus précisément, nous utilisons des systèmes de recommandation dynamiques pour induire les états dans lesquels les utilisateurs pourraient se trouver et capturer leur évolution constante dans le temps. Ainsi, nous étendons les systèmes de recommandation classiques en incorporant les états et profils des utilisateurs dans un environnement statique basé sur une méthodologie de méta-apprentissage. Ensuite, nous explorons des contextes plus réalistes où les environnements sont dynamiquse. Nous explorons trois applications réelles et concrètes : la recommandation de tests éducatifs, la recommandation de requêtes SQL et la recommandation de sessions diversifiées. Pour chaque application, les utilisateurs sont définis selon plusieurs dimensions afin d'éviter la spécialisation excessive et la formation de filtres de bulles. Nous proposons plusieurs solutions basées sur les bandits manchots et l'apprentissage par renforcement.

En plus de leur comportement unique, les utilisateurs ayant des caractéristiques similaires (par exemple, démographiques) peuvent avoir le même comportement global. Ainsi, notre deuxième partie vise à analyser le comportement collectif des utilisateurs et à découvrir les relations entre les différents groupes d'utilisateurs et les ensembles d'éléments (par exemple, les produits). Dans le but de réduire les fausses découvertes, nous nous appuyons sur des tests d'hypothèses pour produire des informations significatives et statistiquement fiables. Nous optimisons également la couverture pour explorer tous les groupes d'utilisateurs et éviter d'analyser qu'un petit sous-ensemble. Nous concevons des solutions novatrices basées sur des méthodes classique de tests d'hypothèses multiples ainsi que sur le α-investing.

Dans cette thèse, nous évaluons nos solutions à l'aide d'un ensemble étendu d'expériences, tant en termes de qualité que de performances. Nous effectuons également une analyse comparative avec des approches existantes pour démontrer l'efficacité de nos solutions.

Contents

List of Figures

List of Tables

List of Algorithms

Dedicated to

My dad who passed away at the beginning of my thesis

My mom and my sisters:

Sarra, Linda, Kahina

Roumaissa who was always there for me

A Special Thank to

My supervisors: Sihem and Idir

Mohamed, Salim, Vera, Abdeldjalil, Lisa, Rosa, Mehdi, Amine and Abdelmoumene.

Chapter 1

Introduction

The current economic landscape is characterized by intensified competition. In such a context, companies are frequently faced with new challenges. This makes customer satisfaction a primary goal, while also seeking to attract new customers, reduce customer churn, and increase sales. Companies are relying on user behavior analysis [92] to achieve their goals and enhance their business value [1]. Moreover, user analysis has been expanding for two major reasons. First, the growth of digital technologies has made it easier to obtain users' purchases, clicks, or search histories. This large amount of data represents the essential starting point for investigating users' behaviors. Second, the technological revolution has brought about considerable changes in the daily behavior of users. In fact, with the expansion of e-commerce websites and the emergence of entertainment and streaming platforms, the quantity of products and services offered to users is massive and overwhelming. As a result, users find difficulties in analyzing the different choices and choosing the appropriate products to purchase or the suitable services to choose from.

Recommender systems are one of the tools used to model users' interactions, analyze their behaviors, and predict their future actions. Marketing studies [124] highlighted the business benefit of providing recommendations to users. Netflix asserts that 75% of the stream time registered in their platform is achieved by recommendations [2]. They also report that their system has helped to save more than 1B$ per year [100]. According to a McKinsey study [3], 35% of Amazon's sales come from recommendations. YouTube also reports that 60% of the clicks on their home screen are the direct results of their recommendations [72].

Recommendation systems find their roots in the mid-70s at Duke University [203]. The first known system that is the closest to modern recommenders is the Grundy library-based system [206]. The system was developed after interviewing users about their book preferences to recommend books to users. Since then, and initially based on information retrieval [212], various approaches were proposed. In 1992, the Tapestry system [99] was developed based on collaborative filtering through human evaluation with the goal of controlling information flows and filtering spam emails. In the following years, many thematic recommender systems were proposed, e.g., GroupLens [203] for Usenet articles, Ringo [223] for music recommenda-

[1] https://sell.amazon.com/tools/amazon-brand-analytics
[2] https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429
[3] https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

tion, and Net Perceptions [82] for offering the marketing recommender engine. The research community developed and continues to develop approaches to improve the quality of recommendations based on works from distinct communities such as information retrieval [78], machine learning [193], data mining [12], and deep learning [269].

Recommendation approaches aim to learn the relationships between users and the different items (e.g., products, services) by analyzing their past interactions. The recommendations are generated by building the users' profiles and predicting the utility of all items. Usually, these models assume that users follow similar past patterns in the future [165]. They then consider that users' interests and items' utility are consistent over time. In other words, they consider that all users' interactions are equivalent over time e.g., an interaction that happened two weeks ago is similar to the one that occurred two days ago. This conception is therefore naive as it assumes a static environment for users. In real-world applications, users and items do not remain constant but are continuously evolving over time [136]. Users may change their interests while items may discarded from the catalog. For example, users may stop consuming some products when starting a diet. On the other hand, certain products may be stopped from being manufactured. Moreover, users' behavior usually follows a sequence in time where previous and next interactions are not independent. Therefore, there is a need to capture these constant changes and encounter dynamic recommendation approaches to better investigate users' behaviors.

In addition to their dynamic attitude, users who share similar demographics may express the same behavior towards one or a subset of items (e.g., Teenagers watch regularly romantic movies) or experience the same behavioral shift over time (e.g., The French increase their cinema attendance in the four weeks following the Cannes Festival). Hence there is a need to investigate the user behavior at a higher level and discover the patterns that the different user groups exhibit when interacting with the potentially recommended items.

In this introductory chapter, we first introduce in more detail the individual user behavior along with its challenges in both static and dynamic environments in Section 1.1. We then present the collective user behavior in Section 1.2. Finally, we present the organization of the thesis and an overview of contributions in Section 1.3.

1.1 Individual User Behavior

From the recommender system perspective, user behavior can be static or dynamic. In the first case, the assumption is that past and future users' interactions are generated from the same distribution. So, all historical interactions are equally considered and recommendations are generated by assuming an unchanged environment. This case is also global and observable to the use of all predictive models [165]. However, this assumption of a static environment does not hold in real-world applications as users and their preferences are constantly evolving in time. Hence a necessity of modelling individual user behavior within a dynamic environment. In this thesis, we contributed to both static and dynamic settings of users' individual behavior by focusing more on the second one.

1.1.1 Static Individual Behavior

Recommender systems learn connections between users and items from their previously recorded interactions. Several recommendation approaches have been proposed in the last decades [**?**]. Despite this proliferation, one important question is still wide open: Which is the best approach to choose to generate recommendations for a given user in a real-world system? The standard procedure for choosing the best approach is to repeatedly try a pool of existing approaches and choose the one yielding the best overall results according to some evaluation measures. However, one may agree that this procedure is expensive in terms of time, human and hardware resources. Hence the necessity for a methodology that selects the best recommendation approach.

Challenges. The main challenge is that the performance of recommendation approaches is data-dependant. It is related to the characteristics of the data as well as the experimental protocol. More precisely, depending on the available users' and items' features and the type of their interactions (e.g., ratings, clicks), the best recommendation approach is different. For example, the same user has different behaviors when she interacts with a music streaming system or a retail e-commerce platform. Thus, considering users' profiles and their features as well as items' characteristics is crucial when selecting the best recommender. In addition to that, an approach that was established to be the best on a particular dataset does not imply that it is the best for every user within it. In other words, the best recommender does not exist. For example, in a previous collaboration with an industrial partner, Collaborative Filtering Item-based was the best overall approach but an important subset of users had Matrix Factorization as the best performer. Hence the necessity of best recommender selection for individual user behavior.

Many solutions were proposed to connect the performances of different recommendation approaches. One of them relies on hybrid recommendation approaches and more specifically ensemble-learning [227] to combine predictions of all single approaches to obtain a better one. The main intuition is that errors made by every single approach are compensated by others [211]. One may agree that these methods may improve the quality of recommendations but they do not select the best recommendation approach.

A better solution, called meta-learning [67], has received attention in the recommendation community over the past few years. Its goal is to leverage users and item features to learn which recommendation approach is preferable. Meta-learners for algorithm selection can be classified into 3 categories [61]: Global-level which selects the overall best approach for entire datasets, e.g., [94] builds a meta-model using 30 datasets; Mid-level which selects the overall best approach for subsets of data, e.g., [84] builds a meta-learner that chooses between User-based and Item-based collaborative approaches for each user, Micro-level which selects the best approach for each instance in the data [60, 16].

Contributions. We solve the best recommendation approach problem by exploring a meta-solution that selects the best approach based on users' demographics and items' characteristics. In our methodology, we assimilate the features as states that describe users' profiles. We then develop a micro-level meta-learning methodology that leverages in addition to the demographics, statistical meta-features of items and users to select the best approach for different types of (user, item) pairs. We show the effectiveness of the methodology against single recommendation approaches and Ensemble Learning models. This work was published

in IEEE BigData 2021 [42].

1.1.2 Dynamic Individual Behavior

In real-world contexts, users may change their behavior for many reasons. For example, the emergence of new items, the quality drop of the habitually purchased items, or simply the influence of social and temporal factors. In addition, each user's behavior can evolve differently from the others. Thus, static recommendation approaches can not handle this temporal dimension and cannot capture the behavioral shift of users.

Motivating Example. Consider Sammy, a cinema lover who has had a premium subscription to a streaming platform for several years. Originally, a Western movie fan, Sammy gave recently his premium account to his brother who loves Romance and Thriller movies. Imagine now that the streaming platform has an integrated standard recommendation system. Based on past behavior of Sammy, the system would still recommend continually Western movies even after the change of type of movies recorded by the platform. It may also ignore Thriller and Romance movies. On the contrary, a dynamic recommendation system would have learned the change in Sammy's account interactions and would have been able to combine long-term preferences (Western) and short-term ones (Thriller and Romance) to perform balanced recommendations.

Challenges. This example shows the importance of modeling temporal changes. The main challenge is to know how to integrate time into the recommendation approaches and capture the exact behavioral situations in which users might be. These situations capture both users' long-term and short-term preferences. This is challenging as dependencies over the multiple interactions become more complex and less evident especially when a user has a high number of interactions [252]. It is also challenging because of the large space of situations that the system should capture and cover. In addition to the combination of the different preferences, the sequential behavior of users raises the problem of transiting between two different situations. For example, in the first one, users may favor their short-term interactions but rely on the long-term in the next one.

Moreover, recommender systems usually optimize a notion of relevance to capture the preferences of users. Relying only on that dimension to generate recommendations may provoke the phenomenon of filter bubbles [183, 170] where users are isolated from a majority of items and narrowed into bubbles where the same items are always shown and recommended. This may also cause a polarization phenomenon, echo chambers [96], and accentuate the long-tail problem [264]. The long tail problem describes a situation where a small part of items generates a large part of the interactions. For example, in retail, a small proportion of products generate a large proportion of sales.

In order to integrate time into recommendations, earlier works proposed solutions based on a time-decay function [81]. Recent interactions are assigned higher weights than older ones. This gives more importance to the recent feedback while the past ones are underestimated. Based on our motivating example, this is not convenient as the user still wants a few Western recommendations. A second limitation of this method is that it does not capture the seasonal and cyclical characteristics of some items. Another method was proposed as a solution to these drawbacks and is based on time-binning [136]. Users' interactions are split into different time bins where shorter bins represent the users' short-term needs and interests and longer bins

characterize the long-term ones. One main limitation of these methods is that they consider each bin isolated from the rest of users' behavior as each bin is treated separately [198]. This does not reflect the real environment of users. In fact, users generally interact with items sequentially where previous interactions influence the next ones. For this reason and despite the time incorporation in the system to leverage dynamic recommendations, these methods still consider users' interactions as static.

To reflect the real world and better model the sequential behavior of users, a new family of recommender systems [251] was proposed. These approaches are usually based on Markov methods [7]. These models learn the decision-making of users and their preferences changes by modeling their interaction process into states. A state represents a specific situation where a user might be at a given point in time. The goal of the models is to learn how and when users transition between these states. One main challenge in using these models is related to applications where the space of states is either large or continuous. The advent of deep learning methods, and especially Recurrent Deep Learning Networks [251] alongside the success of Reinforcement Learning [7] helped solve this limitation. In fact, Reinforcement Learning is based on a Markov Decision Process. It learns a policy function that consists of a mapping between users' states and which mimics the performances of the recommendation system when interacting with users over discrete time steps. One limitation of these models is that they require more data for training. Moreover, they can also cost in terms of time and hardware resources. Finally, they may rely on complex architectures which makes it hard to explain their output.

Contributions. We study, in this thesis, realistic scenarios where user behavior is constantly changing. We explore three interactive real-world applications where the environment is dynamic: Test recommendation in educational environments, Query recommendation for data exploration, and Multi-sessions recommendation in a shopping environment for example. We leverage in all applications, Markov models [7]: Multi-armed bandits in tests and query recommendation and Reinforcement Learning in the multi-session recommendation. One may argue about the link between these models and contextual recommendation systems [5]. The difference is that most context-aware recommendation approaches incorporate explicit contextual temporal information to define users' situations. Moreover, several dimensions may be used, e.g., user location, mood, social situation, or several temporal dimensions (Holidays, weekends). On the other hand, sequential recommenders leverage temporal information but do not explicitly observe it. As they study the sequence of interactions, temporal context is implicitly considered by design. We then constitute a sub-case of context-aware recommendation where context is dynamic and not observable [5].

In addition to that, within each application, we used different dimensions to describe users' behaviors and their decision-making. In the test recommendation, we defined three dimensions: Expected performance, aptitude, and gap. In the query recommendation, we rely on entropy and variance in addition to a query distance function to cover and explore the whole space of queries. Finally, in the multi-session recommendation, we combine relevance and diversity to avoid overspecialization. We detail each application and its contribution in the following:

1. **Recommendation for Test Assignment.** We study the problem of test assignments and recommendations for students. With technological growth, new learning systems [237, 133] and opportunities have emerged, e.g., tutorials, MOOCs [224]. They

aim to provide an efficient and personalized learning experience for students by capturing their competencies and interactions with various learning activities and dynamically adapting learning content to suit their abilities or preferences [243]. Many tools were developed to either propose a whole adaptive system for skill improvement, e.g., Desire2Learn [4], RiPPLE [132], or to help teachers with assessment generation [127]. Another direction studies the detection of students' mastery [188] and develops criteria that determine whether a student mastered a skill or not. These are based either on simple statistics, e.g., *NCC* (*N Consecutive Correct*) [131], *Moving Average* [190], or on sophisticated techniques that model learners [207, 1], e.g., *Bayesian Knowledge Tracing (BKT)* [64] and *Latent Factor* models [54, 184, 190].

In this first application, we propose an algorithmic work that focuses on crafting dedicated strategies that help students improve their skills. We assume that the state of students is defined by their skill level as well as the difficulties of previous correct and incorrect tests. Based on that, we formulate three dimensions that leverage the state of students and capture their learning: expected performance, aptitude, and gap. Expected performance represents the likelihood that students answer correctly a test based on previous successes, while aptitude characterizes the progression ability that the test offers based on the knowledge level of students. Gap captures the learning disability of students based on previous failures and pushes them to work on their weaknesses. We frame the assumption that optimizing these three dimensions lead students to a better learning experience and verify its veracity. Optimizing several dimensions also tends to offer a more diverse set of items (tests) and avoid targeting students with the same tests. We propose two solutions to recommend a batch of tests to students in an iterative way with respect to the dimensions. The first solution combines all of them and solves a multi-objective problem by finding the Pareto [24] candidates. It relies on the assumption that optimizing all dimensions during the learning process yields the best gain in knowledge. In contrast to this first proposition where the dimensions are fixed, we assume that the needs of students are dynamically changing from one iteration to the next. For example, at the beginning of the process, optimizing for gap is not necessary. We hence propose a solution based on Multi-Armed Bandits [236]. Our solution chooses, at each iteration of the process, the combination of dimensions to optimize. We show the efficacy of these adaptive solutions in offering a better learning process by comparing them against a state-of-the-art solution [158] that does not consider the states of students and assigns tests in alternating difficulty levels, e.g., assigning easy tests before medium tests and finally hard tests. Part of this work was published as a Workshop paper in the 2^{nd} International Workshop on Data Systems Education [46]. Its extension is under review.

2. **Recommendation for SQL Groypby Queries.** The emergence of new technologies led to an increase in data generation and collection. To help users understand and explore this data, one may rely on different analytics, e.g., data summarization [265]. SQL group-by queries represent a form of these analytics that offer interpretable summaries of subsets of the data. Such queries provide the ability to group by some attributes and aggregate by others. In order to convey the results, we propose to couple the queries with visualization. Visualization is important as it offers an easy distribution of

[4] https://www.d2l.com/

information. This increases the possibility of better understanding the data. Through visualizations, users may also interact with the results. In this application, we study the recommendation of SQL group-by queries presented into visualizations called panels.

Many tools were proposed to explore these queries and offer users meaningful panels that display analytic results, e.g., Qualtrics [5]. The drawback of these tools is that they assume high expertise of the user with a solid background on queries and prior knowledge of the database. Other solutions were proposed to infer data queries and recommend visualizations to users [66, 80, 106]. For example, SeeDB [246] relies on a measure of interestingness to recommend panels. This function of utility favors the panels that deviate from some references. Another solution, QAGView [256], assumes a sorting of panels based on a predefined attribute. The drawback of these solutions is that they assume predefined interesting queries or attributes.

The goal is to develop a solution that targets users with interesting panels by assuming a low background knowledge about queries and data and also by minimizing their effort. To the best of our knowledge, this work is the first that learns query recommendations based on user feedback. In this application, we propose a solution that aims to recommend the panel by leveraging the sequence of users' interactions. This solution is also using Multi-Armed Bandit models [236]. Our solution first generates the most effective SQL query. The results of the query are converted into a visualization panel which is recommended to the user. Our solution aims to recommend at each iteration a panel that has never been seen before by the user. To illustrate the functioning of our solution, we present the following example: Consider an analyst who interacts with our solution in order to examine movie data. Our analyst is targeted first with a panel that displays the results of "Select avg(rating), avg(interaction) from movies groupby gender". The analyst rejects this panel. She gives the following reason: "Bad group-by attribute". This means that the panel shows analytics that are not relevant to her. In the second iteration, she is targeted with a panel of this query "Select avg(rating), avg(interaction) from movies groupby location". The analyst accepts this new panel. In the third iteration, our solution recommends "Select max(episodes), min(episodes), count(seasons) from tv groupby name". The analyst rejects this panel without mentioning any reason. Our solution refines then this last panel and suggests "Select avg(episodes), count(seasons) from tv groupby name" which was rejected again by the analyst. This procedure will continue until the analyst stops it.

From this example, one may see that our solution generates panels sequentially. The next recommendation relies on both the answer of the users and the previous panel. If the user gives positive feedback (acceptance), the next panel is generated using new attributes that are distinct from the ones previously accepted (Third iteration). This aims to push the user to explore all the space of possible panels and avoid narrowing her in a small data region. If the user rejects a panel, our model recommends a new panel by refining the last one. Furthermore, when rejecting a panel, the user may provide a reason why. If a rejection reason is given (First iteration), our method then modifies the last proposed panel accordingly and recommends it (Second iteration). If a reason is not given, our solution infers it following one of two distinct semantics. The first semantics uses a distance function between panels. It recommends either a similar panel to the

[5] https://www.qualtrics.com/support/vocalize/widgets/creating-cx-dashboard-pages/

previous one by applying minor changes or a completely different panel by exploring the space. The second semantic relies on the likelihood of application of each reason. In this work, we show the importance of our model to reduce the effort of users in creating dashboards. We also show the effectiveness of the second semantic compared to the first one. This work was published as a demo paper in CIKM 2021 [69].

3. **Recommendation for Diverse Sessions.** We study the problem of diverse multiple-session recommendations. It is known that recommendation approaches tend to recommend items that are too similar to the ones the users interacted with which may provoke the phenomenon of *filter bubble* [183, 170] and overspecialization [204]. As user's interests may be complex, highly dynamic, and heterogeneous [53]. Relying only on similarity and relevance may decrease the quality of recommendations as they do not offer new content. To avoid this problem, the recommender system has to consider other dimensions that characterize users' preferences of users like diversity and novelty [53]. Many solutions were proposed [140, 260, 53] to optimize the diversity in recommendations. One type of diversity incorporation is re-ranking the original list of recommendations [266, 52]. The re-raking procedure stops when the diversity of the list reaches an upper bound. Another approach is Maximal Marginal Relevance (MMR) [52, 90] selects items that maximize a linear combination of relevance and diversity. New methods based on deep learning and Reinforcement Learning were also proposed [269, 200]. For example, Hansen et al. [110] proposed deep learning models and a simple Reinforcement Learning ranker that samples items to produce a ranked list of diverse items. Another diversity-promoting RL model, called SMORL, was developed by [233] that extends the work of [262] optimizing, in addition to relevance, two more objectives: diversity and novelty to produce recommendations. One drawback of these solutions is that they optimize for diversity in a single session and do not capture its evolution across time. The second limitation is that they assume a fixed and predefined notion of diversity that does not allow the best item set that gives the highest diversity score to be obtained.

   In this application, our goal is to develop solutions that capture the variation of diversity between different sessions. More precisely, we aim to select in each session the attribute that yields the highest diversity. Based on this selected attribute, our solution generates a list of recommended items. To illustrate that, we consider a user listening to music through different playlists. Our solution first targets her with a playlist of Eric Clapton's songs from different eras (60's, 70's, etc) and different genres (Blues, Rock, etc). After some time, she receives a less diversified playlist composed mainly of Rock music from the 70s but interpreted by diverse artists. In the end, she listens to a playlist of Blues songs from a variety of eras. One may observe that our solution selects at each time an attribute that yields the highest diversity and which differs across sessions.

   There exists one work that leverages multi-attribute recommendation diversity [76, 77] but it is only applicable to a single session while our solution leverages multiple sessions. By selecting a different attribute each time, we assume a dynamic notion of diversity that is changing from one session to the next one which assures that diversity is always maximized. To permit that, we propose two solutions. The first one is a generalization of standard diversity-based recommender systems. In fact, we extend the MMR [52] and the re-ranking [266] algorithms to the multi-session context. Our generalization consists

of these algorithms iterating over all attributes to find the one yielding the highest diversity. One may agree that this solution becomes inefficient with a large number of attributes. Our second solution overcomes this limit by leveraging Reinforcement Learning. We adapted and extended the method provided by SMORL [233]. Our solution generates, at each iteration, an (attribute, list of items) pair where the diversity of the list is maximized by assuming the last session seen by the users as their state. We show in our work that methods that assume a dynamic diversity are better than the ones with a fixed one. We also show that the RL-based solution is better at finding the best trade-off between diversity and relevance. This work was published in IEEE BigData 2022 [43].

1.2 Collective User Behavior

In addition to their unique and individual behavior, users exhibit global trends when they interact with the potentially recommended items. Examining these trends at an individual level has one main limitation: Sparsity. In fact, individual data is generally sparse [6] and may result in poor pattern discovery. The alternative is to leverage collective behaviors by grouping users. In fact, grouping users allows sparsity reduction and analysis improvement [175]. One solution would be to use Group-based recommendation systems [71] but it has one main limitation: confidence of the results produced by the recommender system [104]. In fact, predictive models may mislead the users as they can extract patterns that are considered significant but are not in reality. As a consequence, without a significance test, the models might extract false insights. Hence the great necessity of relying on user groups and leveraging hypothesis testing to extract statistically sound patterns and offer significant insights and discoveries about users' collective behavior.

Motivating Example. Consider an expert who wants to know how different categories of users interact with newly released products in an e-commerce platform. In the beginning, the analyst wants to discover the age groups for which the number of clicks on these products is greater than a threshold. After selecting the 5 most accurate groups, the analyst explores the residence states of these groups based on the overall distribution of the ratings they gave to the products. The analyst wants to select the geographic groups for which the ratings are not uniform. After selecting the most diverse geographic groups, the analyst desires to compare the average rating of product types two by two and select the ones that are extremely dissimilar. At the end of the process, the analyst obtains pairs of new products that have a minimal number of clicks and for which the rating distribution within each geographic group is diverse.

Challenges. To realize this example, the system has to apply, at each step, the users' demographic and product types filters, aggregate over the specified dimension (e.g., clicks, ratings), and return the best geographic groups. The main challenge is the risk of misleading the exploratory process. In fact, data groups, that are not significant, may seem interesting. The system will then return insignificant and meaningless groups to the analyst. To avoid this problem, we leverage hypothesis testing to verify the relevance of the information. Hypothesis testing is one of the first statistical inference methods [86] and one of the most used [146]. Widely used in many scientific fields like biology [240], medicine [167], or psychology [218], hypothesis testing allows us to make probabilistic statements and assure that the results are

meaningful with a certain probability of making a mistake (p-value). This p-value should be smaller than a predefined and fixed value (α), which is usually set to 0.05, to assert the significance.

Applying hypothesis testing in the motivating example means that many hypotheses are tested separately as in each step, each hypothesis characterizes one user group. For example, generating m groups is equivalent to testing m hypotheses. This is referred to as multiple hypothesis testing [221]. Performing multiple hypotheses has one main challenge. When multiple user groups are tested, the chance of observing a rare event increases, and hence, the likelihood of incorrectly returning a significant result increases too. In other words, when the number of hypotheses increases, the expectation of returning insignificant results also increases.

The second challenge of realizing multiple testing is the risk of returning results that are not representative of the input data. In our example, we may explore a small subset of the user groups and ignore the remaining ones especially when the number of groups is too high. This may narrow the scope of the returned results and lead the analyst to misleading conclusions. For instance, in the second step of the previous example, the system may exclusively return groups from the south of France and ignore the remaining regions.

We address the first challenge by verifying the confidence and significance of the information returned by each resulting group. Many procedures were proposed to control the error of returning insignificant hypotheses: Bonferroni correction [30] that controls the probability of making a false discovery, Benjamini-Hochberg [33] that controls the False Discovery Rate (FDR), and α-investing procedures [89] that computes the tests on the fly, each with a specific level of α, by controlling the marginal FDR. We discuss these procedures in more detail in Section 2.2.

Our work is also related to others that already used multiple hypothesis testing in machine learning, data mining, data exploration, and visualization. Existing work on customer segment discovery [189, 107] combines the computational power of pattern mining with multiple hypothesis testing to find genuine and meaningful patterns in the data. Another idea was considered in [254] where the authors introduced Subfamily-wise Multiple Testing, a multiple testing correction subjects to a risk budget. A similar idea is considered in the context of interactive data exploration in [273] where different α-investing heuristics were used [89]. Existing work has also combined machine learning and statistical testing to verify associations in genome data (e.g. [162]), or compares the performances of different machine learning classifiers using different datasets [74].

The second challenge is particularly important for large datasets where the number of groups that pass the test increases. We address it by optimizing for coverage in addition to the significance and revisiting the conditions under which a hypothesis is tested. By doing that, we ensure that the returned groups are representative of the whole data. For example, by maximizing coverage, in the second step of the previous motivating example, the system will return user groups from all regions of France and does not focus only on a specific region.

Contributions. To the best of our knowledge, no prior work optimizes statistical significance and coverage to find interesting patterns and insights in data. We formulate ValMin and CovMax, two generic top-n problems that seek n significant user groups that are representative of the input data. ValMin optimizes significance while setting a constraint on data

coverage. We develop a greedy algorithm to solve the former problem based on Bonferroni [30] and Benjamini-Yekutieli [34] corrections. We compare this greedy solution to another proposed one that is based on the Subfamily-wise [254] solution. Solving this problem extends the multiple hypothesis testing by enforcing data representativity but it does not assure the maximization of coverage. We then optimize the second top-n problem CovMax that aims to maximize data coverage while controlling significance. We propose two algorithms where the first one is a greedy algorithm with a provable approximation guarantee which is also based on Bonferroni [30] and Benjamini-Yekutieli [34]. The second solution is a heuristic-based algorithm based on α-investing [89]. In these two problems, we encounter different types of hypotheses as we handle One-sample, two-sample, and multiple-sample tests, and different dimension aggregation functions (mean, variance, and distribution). Our experiments demonstrate the necessity to optimize coverage for sound discoveries on large datasets, and the efficiency of our algorithms. This work was originally published in WWW 2022 [45] and extended to a journal paper published in the Transactions on Large-Scale Data- and Knowledge-Centered Systems [44].

1.3 Thesis Organisation

This thesis is organized around three main domains of investigation that are demonstrated in the framework. We pursue the following outline:

1. In Chapter 2, we introduce the concepts related to recommender systems. We cover all recommendation approaches focusing more on the ones used in this thesis. We present the different methodologies to quantify the quality of recommendations and the different metrics used to evaluate them. Finally, we discuss the challenges and limitations and how existing works propose to tackle and solve them.

2. In Chapter 3, we first investigate the best recommendation approach problem in a static context. We rely on an existing methodology, meta-learning [67], and propose an approach that chooses for every instance in the data, the most suitable recommendation approach to consider based on user profiles. Our approach generalizes existing works to return Top-N recommendations and leverages implicit and explicit data. We conducted extensive experiments on four real-world data and compared our approach against single recommendation approaches and hybrid ones. Our results show the need to leverage the results of a variety of approaches as the meta-learning methodology provides more accurate recommendations. This work was published in IEEE BigData 2021 [42].

   In this same Chapter, we explore the different real-world applications of dynamic recommendations:

   (a) We develop a greedy solution that solves a multi-objective problem for test assignments in an educational context (Section 3.2.1). We propose a Pareto solution [24] that relies on Hill Climbing [176] optimizing three dimensions related to the learning of students. We extend this solution using Multi-Armed Bandits [236]. This second solution learns at each iteration the dimensions to optimize and generates the batch of tests according to that. We conducted simulated experiments on semi-synthetic data and showed that our first approach outperforms a state-of-the-art solution, Alternate which recommends tests by alternating over the difficulty

levels [158]. We also showed that our dynamic extension outperforms the first solution. This work was initially published as a Workshop paper in the 2^{nd} International Workshop on Data Systems Education [46]. Its extension is under review.

(b) We develop a solution based on Multi-Armed Bandits [236] for the query recommendation problem (Section 3.2.2). We developed a tool that guides users in generating visualization panels based on groupby analytical queries with the objective of minimizing their effort and time. We conducted simulated experiments using real-world data and showed the effectiveness of our solution. This work was published as a demo paper in CIKM 2021 [69].

(c) We first proposed a generalization of existing diversity-based recommendation solutions: re-ranking [266] and MMR [90] to solve the dynamic diversity problem in session recommendation (Section 3.2.3). We then leveraged Reinforcement Learning [236] and extended an existing architecture, SMORL [233], to take into account multi-session diversity and compared it against standard diversity-based approaches that do not assume a dynamic notion of diversity. Our extensive experiments on semi-synthetic data show that our solution offers recommendations that have the best diversity and accuracy trade-off in a quicker time. This work was published in IEEE BigData 2022 [43].

3. In Chapter 4, we explore hypothesis testing and coverage for discovering Collective Behavior. We propose a solution that supports a variety of statistical tests to verify users' collective behavior. We formulate two generic problems for significant and covering discoveries and propose one solution to solve the former and two to solve the latter. We extend multiple hypothesis testing corrections to optimize and maximize coverage. Our extensive experiments on real data show the effectiveness of our proposals in terms of significance and data coverage. This work was originally published in WWW 2022 [45] and extended to a journal paper published in the Transactions on Large-Scale Data- and Knowledge-Centered Systems [44].

4. We conclude in Chapter 5 and propose future directions of this thesis.

Chapter 2

Related Work

2.1 Recommendation Systems

With the expansion of the web and the evolution of technology, the amount of used and analyzed data has become very large. So much so that it has become difficult for users to know which products or services to choose. For example, with the spread of streaming platforms such as Netflix [1], users are asking themselves which movies or series they should watch. With the emergence and expansion of e-commerce websites, such as Amazon [2], users struggle to find the appropriate choices from the huge variety of products offered by these websites. The variety of content and services available on the web and the introduction of new ones have led to customers and users making poor decisions. The huge availability of choice, contrary to what one might think, has the side effect of diminishing the well-being of users [122]. For this main reason, recommender systems were proposed as a solution to solve the problem of information overload.

Recommender systems have large real-world applicability. They are used to recommend music on Spotify [2, 159], movies on Netflix [100], products on Amazon [229], friends and people on Facebook [20], or Twitter [108], and other daily life fields like restaurants [18], points of interest (POI) [248], or healthcare services [88]. Their popularity is explained by their use for increasing user satisfaction [134] and user retention [17], and by user loyalty and overall business revenue.

This section provides an overview of recommender systems, the different methodologies and metrics that evaluate the quality of recommendations, and the most popular approaches focusing on the ones that are relevant to this thesis. We raise and summarize the limitations of standard recommender systems and discuss the ones that are related to this thesis.

2.1.1 Recommender Systems

Recommender systems gained a huge interest and arguably became popular in academia and in industry after the Netflix Challenge [35] in 2006. The competition was destined for the

[1] https://www.netflix.com/
[2] https://www.amazon.com/

community in order to develop systems to predict movie ratings. The objective was to beat the accuracy of the Netflix model for a prize of 1 Million $.

Since then, many literature works tried to define recommender systems. The most general one is that of [50]: *Any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options.* Another definition is given by [204]: *RS are software tools and techniques that provide suggestions for items that are most likely of interest to a particular user.*

To predict interesting and personalized items, an RS relies on the interaction history of users with these items from which their potential preferences are inferred. The inference depends on the nature of interactions and the type of data the system is using. Indeed, users' actions may have two distinct forms: Explicit and implicit.

Data Types

During the Netflix competition, the company released a large movie dataset, a collection of 100 million ratings given by users to a set of movies. Thus, the interaction history of users was in the form of records *explicitly* provided by them. These records represent their real preference judgment of the different movies. This type of feedback is called *explicit* feedback.

Explicit Feedback. It often takes the form of numerical ratings provided by users to express their explicit preferences for different items, e.g., a 5-star rating scale. Other forms of explicit data may exist such as the binary scale [216] (Like, or dislike), or in the form of textual comments and tags [149].

One main drawback of this type: it is not always available. Collecting this data requires users to perform an action every time, which might be problematic given the huge digital volume that users are interacting with on a daily basis. It would be difficult for users to provide each item with a preference score. As a result, systems can rely on the second type of feedback: *implicit* feedback.

Implicit Feedback [174]. It is collected without any intervention or additional actions of users. Implicit feedback includes search and browsing history, as well as clicks, and purchase history.

Comparing these two types, it is clear that implicit feedback is more used than explicit one due to its ease of acquisition. However, it is less accurate as it can be biased and noisy. Also, there is a trade-off between data quality and quantity. The former has, obviously, better quality but lower quantity, while the latter has more data but lower quality. They are, in effect, complementary.

Type of Recommendations

Recommender systems can have two different functions: predicting missing ratings, or recommending a list of items [4].

Rating Prediction. During the Netflix Prize, the company formalized the problem as "Rating Prediction". It consists of the prediction of the rating that a user might give to an item, which they have not previously rated. It learns a user utility function by assuming

that the suitability of an item for that user is related to the rating given by this latter. The highest-rated items which maximize the utility can be recommended to the user. In other words, it is the prediction of users' potential favorite items.

Top-N Recommendation. In general, users are interested in the subsets of items that are likely relevant to them. For this reason, the top-N recommendation has been developed. It consists of the prediction of the usefulness of an item to a user, e.g., Will the user watch that movie or not? It learns a user utility function by assuming the suitability of an item for that user is related to the relevance of the former. The items that maximize relevance can be recommended to the user.

2.1.2 Recommendation Approaches

In order to identify the list of items, which may be of interest to users, recommender systems calculate a utility for each pair "user-item". Items with the highest utility are recommended to the targeted user. Several algorithms have been proposed, using machine learning, statistics, graph theory, and information extraction methods, to have a better recommendation quality. Generally, recommendation approaches are classified according to the formalization and the estimate of utility as well as the used type of data. This section discusses the most common classifications in the literature [4]. The reader may also refer to other classifications that were proposed [51, 205]. Recommendation approaches are classified into these categories:

- **Content-Based Filtering:** It is based on the user's own history, user, and item features to predict similar items to those they were previously interested in.
- **Collaborative Filtering:** It is based on the user's interaction history and the interactions of all similar users to predict items that may be relevant.
- **Hybrid:** It combines both previous approaches using the user's history and item features.

Without loss of generality, in this section, we focus mainly on collaborative filtering approaches as these are the ones exploited in the thesis.

Content-Based Filtering Approaches

Content-based recommender approaches use content information as features, text about items, and users to generate recommendations. In effect, these systems rely on the description of items and the interaction history of users to create their profiles respectively. The idea behind this approach is to recommend items that have similar profiles compared to the targeted user. The goal is to users and items profiles. As a user profile is built based on its history, the approach recommends items that are similar to the ones they previously interacted in.

The general architecture of the approach is presented in [156], and in which the recommendation is based on three main axes:

- **Content Analyzer:** It is pre-processing of the description of items that can be unstructured. It also represents the extraction of the relevant information on the items and represents it in an appropriate format.

- **Profile Learner:** It allows the computation of users' profiles from their interaction histories. Given the continuous nature of users' activity and their potential change of opinion, interactions are collected periodically. So, the profiles of users are updated with the same frequency. The creation of these profiles is performed through machine learning techniques, such as Bayes [220], neural networks [26], or decision trees [185]. However, a huge number of works use the VSM (*Vector Space Model*) vector formalization and the TF-IDF (*Term Frequency - Inverse Document Frequency*) measure [186]. The reader may refer to [156] for further information about profile learning methods.
- **Filtering Component:** It selects items that are relevant to the user by matching the profile of these latter to the ones of all potential items. The match is done using a similarity function.

Content-based approaches have many benefits. They may offer explainable recommendations as they can provide, at the same time, the set of relevant content information that led to the choice of these recommended items. These approaches may be a solution to the Item Cold-start problem. Newly added items in the system can be recommended without being rated or purchased by any user. However, User Cold-start represents one of its drawbacks. A newly added user has to interact with a number of items so that their profile is properly learned. Another drawback is item content availability. Finally, content-based approaches tend to offer overspecialized recommendations. Users are overly narrowed into a small set of recommended items which results in the phenomenon of *Filter bubble* [183].

Collaborative Filtering Approaches

Collaborative recommender approaches recommend items to users based on their interactions without taking into account exogenous information about these items or users. This term, first used in the Tapesty document recommendation [99], means the collaboration between similar users using their respective interactions to better recommend items. After the Netflix Prize [35], many advances were made in the field of collaborative approaches [137].

The general idea lies in sharing experiences and opinions between different users. Indeed, these approaches assume that relevant recommendations to an active user, which may contain items that the latter has not interacted with, can be made based on the opinions and actions of a group of similar users. Collaborative recommendations are, therefore, based on the interaction history and behavior of similar users.

There are two main approaches to collaborative filtering: *Neighbourhood-based methods*, also called *Memory-based methods*, or *Heuristic-based methods*, and *Latent factor models*, also called *Pattern-based methods* [4]. In addition to the interaction history, neighborhood methods generally focus on the relationship between items or alternatively between users while model-based methods, such as matrix factorization, rely on transforming item and user information into a single latent factor space. The latent space attempts to explain users' interactions based on factors automatically learned and inferred from their history. In this section, we are going to discuss these two types of methods in more detail.

Neighborhood-based Approaches. The best-known and most common approaches to collaborative filtering are based on the neighbors [173]. They are based on *k nearest neighbors - KNN*. *GroupLens* system [203] is an example that applies the user-based version. This

version assumes that similar users are interested in the same items. It predicts, for each user, the items for which they didn't interact and that are popular for similar users. An analogous version, based on items and called item-based, was also developed a few years after the user-based [214]. This one assumes that users are interested in similar items to the ones they interacted with.

Neighborhood approaches are simple and can offer intuitively explainable recommendations as they are based on a clear similarity function.

• **Similarity Measure.** The calculation of similarity has a considerable role in neighborhood methods. These measures allow the selection of neighbors to be used in the prediction. They are a very critical aspect as they have a significant impact on their quality performance. There are several similarity measures, we discuss the most commonly used one, *cosine* similarity.

Cosine Similarity: Given two objects, users or items, a and b, the cosine similarity between these objects is represented by the cosine of the angle formed between their two vectors $\vec{x_a}$, $\vec{x_b}$:

$$sim(a,b) = cos(\vec{x_a}, \vec{x_b}) = \frac{\vec{x_a} \cdot \vec{x_b}}{||\vec{x_a}||||\vec{x_b}||}$$

For example, in the case of user-based KNN, each individual u is represented by a vector $\vec{x_u}$, where $x_{ui} = r_{ui}$ such that r_{ui} is the rating that user u gave to the item i. Therefore, the similarity between two users u and v is given by :

$$sim(u,v) = cos(\vec{x_u}, \vec{x_v}) = \frac{\sum_{i \in I_{uv}} r_{ui} \times r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

where I_{uv} is the set of items that both u and v interacted with. The maximum value of similarity indicates that both users have identical preferences, and the minimum value indicates that they have nothing in common. The reader can refer to [173] for other similarity measures, e.g., Pearson correlation, Jaccard similarity.

• **Neighborhood Selection.** The number of neighbors that are selected also has a considerable impact on the results and the performance of neighborhood-based approaches. Techniques are applied to pre-filter the set of neighbors. We discuss some of them. The remaining ones are mentioned in [173].

Top-N Filtering: The N best similar neighbors are only considered ones. The choice of N must be very judicious to avoid a large memory consumption if N is large, or the *Long Tail* problem if N is very small.

Threshold filtering: Neighbors with a similarity above a threshold S_{min} are considered. Despite the challenge of determining the value of S_{min}, it is more global and flexible than the previous method.

• **User-Based and Item-Based Approaches.**

User-Based: It represents the original version of neighbour-based filtering [99]. For each user u, the inferred rating $\hat{r_{ui}}$ of an item i is determined by the aggregation of all ratings

given by the neighbors to that item:

$$\hat{r_{ui}} = \frac{\sum_{v \in V_i(u)} r_{vi}}{|V_i(u)|}$$

where $V_i(u)$ is the set of neighbors of user u that rated the item. This aggregation is simple but has many drawbacks, e.g., it considers that all neighbors are equal. For this reason, it has many extended versions, e.g., Weighted aggregation by their similarity. The reader may explore [173] for all other extensions.

Item-Based: Introduced by [214], this approach infers the rating $\hat{r_{ui}}$ of a potentially recommended item i by calculating the aggregation of ratings given by user u to the neighbors of i. The neighbors are determined for a fixed user.

$$\hat{r_{ui}} = \frac{\sum_{j \in V_u(i)} r_{uj}}{|V_u(i)|}$$

where $V_u(i)$ is the set of neighbors of item i that have been rated by u. This may be extended to a weighted aggregation as explained in [173].

It has been shown that the Item-Based approach gives better recommendations [214] as the similarity between items tends to be more stable than users' similarity which might change over time. Despite the simplicity and explainability of neighborhood-based recommendations, they have several limitations. In fact, these approaches are sensitive to the data that are sparse. They also lack in covering the entire items in the dataset. Several approaches have been proposed to overcome these problems, such as dimension reduction or graph-based approaches [173]. However, despite these solutions, it has been shown that model-based approaches handle much better data sparsity and provide more accurate recommendations. Therefore, they are preferred in recommendation applications.

Model-based Approaches. The aim of these approaches [137] is to create intermediate latent factors to explain users' interactions. The principle is to train a model based on user history and determine values of different parameters, which are used in the prediction of the utility of each item. These methods tend to find more patterns in the data. Thus, they provide better recommendations than neighborhood-based methods.

Several methods have been developed, such as neural networks [113, 150], Markov process [197], random walks [148], or bandits [239]. The most used methods are matrix factorization [143, 138]. The latter methods are more popular due to the accuracy of the recommendation provided as well as their scalability. In this section, we discuss different methods of matrix factorization, association rules, deep learning, and reinforcement learning recommendation algorithms.

• **Matrix Factorization.** The principle of matrix factorization [138] is the representation of users and items in a common latent space, of dimension d. If all interactions of users and items are represented as a matrix, the product of the new factors of users and items results in that matrix. These factors, learned from the data, are used to infer the utility score of each item for each user.

Given an interaction matrix R, the main objective is to determine the latent vectors $p_u \in \mathbb{R}^d$, $q_i \in \mathbb{R}^d$, which represent the information inferred from R that characterize the preferences

of user u and the characteristics of item i respectively. The deduction of the utility score $\hat{r_{ui}}$ of item i with respect to user u is calculated as the inner product of their embedded vectors:

$$\hat{r_{ui}} = p_u q_i^\top$$

From this equation, the deduction of R is performed as follows:

$$R = PQ^\top$$

where $P \in \mathbb{R}^{n \times d}$ and $Q \in \mathbb{R}^{m \times d}$, and n and m represent the number of users and items respectively.

The learning goal is to determine the latent matrices P and Q from the interaction matrix R which are used to reproduce that input matrix by predicting all its unobserved interactions. Several methods were proposed. They differ in the way to learn the embedded matrices by either minimizing a point-wise [137] or a pair-wise [202] objective function. We introduce the ones that were leveraged in this thesis.

Singular Value Decomposition (SVD): SVD [65] is a technique from linear algebra, initially used for dimensionality reduction. It decomposes the matrix R as follows:

$$R = P\Sigma Q$$

where $P \in \mathbb{R}^{n \times d}$ is the orthogonal left singular matrix. It represents the relationship between users and learned factors. $Q \in \mathbb{R}^{m \times d}$ is the orthogonal right singular matrix. It represents the relationship between items and learned factors. $\Sigma \in \mathbb{R}^{d \times d}$ is a diagonal matrix containing the d singular values on its diagonal.

Using SVD for recommendations has one main drawback: it requires a full matrix to be decomposed and is undefined in the presence of unknown values. This may be problematic as interaction matrices are sparse and many user-item values are missing (The user did not interact with that item). Several solutions were proposed to address this issue by either replacing the missing values with zero [213] or with an aggregation per user or per item [141]. These solutions lead to a dense user interaction matrix which can become computationally expensive. It may also lead to inaccurate solutions since filling the matrix might falsify the represented concepts [11]. Another solution proposes [137] to learn the factors only on known interaction through a point-wise objective function that minimizes prediction error. The predicted rating $\hat{r_{ui}}$ of a user u to an item i is set as:

$$\hat{r_{ui}} = \mu + b_u + b_i + p_u q_i^\top$$

where b_u, b_i, and μ represent user bias, item bias, and overall bias respectively. If the user u is unknown, the bias b_u and the factors p_u are assumed to be zero. The same applies to item i with b_i and q_i. In order to learn the model parameters (b_u, b_i, p_u and q_i), the following regularized squared error is minimized using stochastic gradient descent [41]:

$$\sum_{(u,i) \in R_t} (r_{ui} - \hat{r_{ui}}) + \lambda(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2)$$

where R_t is the set of all known interactions and λ the regularization parameter.

Another version of SVD was proposed in the literature: *Non-Negative Matrix Factorization (NMF)* [268]. This approach is similar to the original SVD but it restricts the values of user and item factors to only non-negative values. It has been shown that it gives good performances [98].

Alternating Least Squares (ALS): It has been shown in some works that using only observed interactions present in matrix R to factorize it leads to more efficient results [238]. This is done by minimizing a point-wise objective function which is the squared error of all observed interactions:

$$\sum_{(u,i)\in R_t} (r_{ui} - p_u q_i^\top)^2 + \lambda\Omega(P) + \lambda\Omega(Q)$$

where R_t is the set of all known interactions, λ the parameter of regularization, $\Omega(P)$, and $\Omega(Q)$ the regularization terms. Regularization is applied to avoid overfitting.

Minimizing this objective function can be done using stochastic gradient descent [41] or alternating least squares [31]. We leverage the latter as it has been used in the past in the context of implicit data [120].

The principle of *ALS* is to fix one of the two unknowns, p_u or q_i which makes the function convex for the second one. The calculation is then done by alternating the choice of the parameter to be fixed at each iteration. This method [31, 11] relies on the execution of the following two steps, until convergence:

1. To optimize p_u, the vectors of the matrix Q are fixed. The optimal value of P, obtained after solving the least squares, is:

$$P = (QR^\top)(QQ^\top + \lambda I)^{-1}$$

2. The same method is used to determine Q. The formula for calculating this matrix is:

$$Q = (PR)(PP^\top + \lambda I)^{-1}$$

   where I represents the identity matrix.

Bayesian Personalized Ranking (BPR): It is an optimization principle designed to deal with implicit feedback [202]. BPR falls into the category of "learning-to-rank" algorithms as a general framework for pairwise learning. Unlike other matrix factorization approaches, BPR uses pairwise item preferences as training data and optimizes for correctly ranking item pairs instead of estimating scores for single items.

This assumes that if a user u expressed an implicit preference, by interacting with an item i, then u prefers i over all other items that they did not interact with. No preference can be inferred for two items that the user interacted with or two items the user did not interact with. Hence, a training instance is a triple (u, i, j) which reflects that customer u prefers item i over item j, denoted $i >_u j$.

The set of all inferred preferences D_S, i.e., the training data used for optimization, is defined as follows:

$$D_S = \{(u, i, j) \mid i \in I_u \wedge j \in \mathcal{I} \setminus I_u\}$$

where I_u is the set of all items that the user interacted with while $\mathcal{I}$ is the set of all items. The generic optimization criterion of BPR is given as:

$$OPT(\mathcal{D}_S) = argmax_\Theta \sum_{(u,i,j)\in D_S} ln\, \sigma(\hat{x}_{u,i,j}) - \lambda_\Theta ||\Theta||^2$$

Where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the logistic sigmoid function, $\hat{x}_{u,i,j}$ is the pairwise prediction for user u and items i, j, Θ is a parameter vector of an arbitrary model and λ_Θ, is a model-specific regularization parameter to prevent over-fitting.

$\hat{x}_{u,i,j}$ is a real-valued function of Θ which captures the relationship between user u and items i and j. The estimation of $\hat{x}_{u,i,j}$ is performed through matrix factorization but since it can only predict single scores, the estimator is decomposed into single prediction tasks: $\hat{x}_{u,i,j} = \hat{x}_{u,i} - \hat{x}_{u,j}$. The optimization uses Stochastic Gradient Descent with a bootstrap sampling of training triples with the following update rule [202] (α is the learning rate):

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{u,i,j}}}{1 + e^{-\hat{x}_{u,i,j}}} . \frac{\partial}{\partial \Theta} \hat{x}_{u,i,j} + \lambda_\Theta . \Theta \right) \tag{2.1}$$

• **Association Rules.** The association rule-based model [10] is a frequently used technique to analyze the purchasing pattern of users. The goal is to find all items that are purchased together in the same transactions. It then predicts the purchase of an item by a user based on that co-occurrence relationship between all items.

The association rules take the following form: $x \implies y$ which means that whenever users purchase a set of items x, it is likely that they will also purchase the set y. Generally, the main objective is to find all relevant association rules for a given user u. The most intuitive approach is to identify all possible rules before filtering the ones that are irrelevant. Given its considerable cost, several optimization techniques have been developed that help minimize the number of rules to be created, e.g., Apriori method [114].

The bi-gram association rules are a simple example of this model. The rules are constructed so that x and y contain only one item ($i \implies j$). This method builds an association matrix A, from the interaction one R, where each entry $a_{i,j}$ corresponds to the confidence of the association rule $i \Rightarrow j$, estimated as follows:

$$conf(i \Rightarrow j) = \frac{R_{\bullet i}^T R_{\bullet j}}{||R_{\bullet i}||_1}, ||R_{\bullet i}||_1 = \sum_{l=1}^{n} |R_i|$$

The confidence represents the frequency of occurrence of j in transactions where i appears. To generate the recommendations for a user u, the bi-gram approach first identifies a set of association rules of the form $i \Rightarrow j$, where $i \in I_u$, the set of all items that the user interacted with. It then recommends the items with the highest score values:

$$score(u, j) = \sum_{i \in I_u} conf(i \Rightarrow j)$$

Despite the obvious connection between this method and the recommender systems, its use remains limited [13]. However, few works used it for performing item recommendations, especially in e-commerce platforms [215]. Another work [194] conducted an experimental study comparing the performance of various collaborative filtering approaches, including association rules, on a purchase dataset coming from a French building supplies chain.

• **Deep Learning Approaches.** Deep learning has become the method of choice in several domains [166, 163, 139, 247]. The field of recommender systems does not make an exception as it is widely used in recent years [70]. Several works on algorithmic aspects of recommender systems are based on leveraging Neural Networks as a core technique. A classification of these approaches was proposed by [269] based on the different neural networks:

Multilayer Perceptron: These approaches extend existing recommender methods, e.g., Matrix Factorization by adding nonlinear transformation. Neural collaborative filtering (NCF) [113], and Neural Network Matrix Factorization (NNMF) [83] generalize Matrix Factorization by replacing the inner product with a neural architecture that can learn an arbitrary function from the data. They assume as input the one-hot identifier of the users and items and predict either the ratings, in the case of explicit data, or the relevance in the case of implicit data. Other work extended pairwise ranking methods [231] and factorization machines [105].

Autoencoder: It exists two ways to apply autoencoders to recommendations: learn lower-dimensional latent vectors at the bottleneck layer [152] or fill missing values of the R in the reconstruction layer [217]. Collaborative Variational Autoencoder (CVAE) [150] considers both content and interaction information. It learns latent vectors from content data and relations between users and items from both content and interactions. Variational Autoencoders for Collaborative Filtering (Mult-VAE) [152] is exclusively designed for implicit data. It learns the latent space by approximating it with a probabilistic distribution.

Convolutional Neural Networks: These models are either used to extract features, e.g., image sources [253], audio sources [242], or video sources [144]. CNNs were also used to extend NCF [113] resulting in a new method called ConvNCF [112]. The CNN was used to capture the correlation between the user and item embeddings.

Recurrent Neural Networks: They are suitable for session-based and sequential based recommendations [251]. For example, GRU4Rec [115] is a session-based model that was proposed to model the order and dependencies between items in each session and infer the items that compose the next session. Recurrent Recommender Network (RRN) [259] is a sequence-based method that uses LSTM to capture the seasonal evolution of items and changes in user preferences over time.

The reader can refer to [269] for other deep learning recommendation approaches. These methods offer many advantages as nonlinear transformations which makes them capable to capture complex user-item interaction patterns. They also can provide a better learning representation as they can consider, simultaneously, heterogeneous content information, e.g., images, and text. However, a recent piece of work at ACM RecSys'19 [70] performed an experimental study and found that most deep learning approaches do not perform better than "simpler" algorithms such as Item-Based collaborative filtering. They also lack interpretability and explainability as they behave like black boxes.

Collaborative filtering approaches have many benefits. Unlike content-based approaches, they only need the interaction history of users and do not require additional content about users and items. They can also recommend relevant items that the user would not have found on their own which reduces the overspecialization of recommendations. However, item and user cold-start represents one of its drawbacks. They also need a significant amount of data in order to fully train the models.

Hybrid Approaches

Collaborative and content-based filtering approaches are complementary, each with its own challenges. In order to avoid the disadvantages of both approaches and to take advantage of their benefits, the hybridization of both techniques has become popular in the research community of recommender systems [51]. Intuitively, an item is recommended to a user only if its description is similar to the user's profile and if the neighbors of the user interacted with it (respectively, the user interacted with the neighbors of items). From the several classifications of hybrid approaches that were proposed in the literature [50, 4], we leverage the one proposed by [4]:

- **Combining separate recommendations:** The recommendations, made separately, by the two types of approaches are combined either by using a linear function [58] or by choosing in each case the best of the two approaches according to a predefined quality metric [38].
- **Adding content characteristics in collaborative approach:** An intuitive method is the incorporation of the user's profile, inferred from the content-based approach, into the designation of the neighborhood in the collaborative approach [22].
- **Adding collaborative characteristics in the content-based approach:** A method is to apply collaborative filtering on a group of profiles inferred from the content approach [230].
- **Unified recommendation approach:** Several methods have been proposed as [192, 63].

Apart from this classification, there is another one in the literature such as [50] which assigns seven different categories. In [161], it is shown that hybrid approaches offer more accurate results than collaborative or content-based filtering.

2.1.3 Recommendation Evaluation

The main objective of the evaluation is to measure the ability of recommender systems to achieve their objectives. For this, the choice of the appropriate methodology, criteria, and metrics is crucial.

Methodology of Evaluation

It defines the evaluation protocol used to test the approaches. Three methodologies were defined in [104]:

Offline Methodology. This method is the most popular and used one in the literature as it allows for the comparison of several approaches at a low cost and without any interaction

with real users. It exploits users' interaction history to simulate their behaviors by assuming that they remain the same from data collection to system deployment.

Generally, this methodology splits the data into three subsets: **Training subset**, mainly used to reproduce the user's behavior towards the items and train the recommender approach, **Validation subset**, mainly used to fine-tune the parameters, and **Test subset**, with which the approaches' performance will be inferred. The choice of the split is crucial as it has a considerable impact on the quality of recommendations. It has to be determined according to the type of the problem and the data available from the several proposed procedures. We report two of them. The reader can refer to [104] for more.

Random split: The data is randomly split, after shuffling, according to fixed percentages for each subset. An interaction can only be found in one and only one subset.

K-fold cross-validation : This method is more sophisticated than the previous one. It consists of separating the data into k subsets. A cross-validation is applied by choosing one of the subsets as the test set and the union of the remaining ones as the training set. This operation is repeated k times by changing the test set each time.

The main drawback of this methodology is that it does not capture the influence of the recommender systems on users. In addition, it is only suitable for a limited number of metrics. This is why it is not sufficient to determine the quality of an approach of recommendation. Other methodologies that consider the real involvement of users during the test phase are proposed.

User Study. In this methodology, the attitude of the users is recorded as they perform controlled experimental tasks. Quantitative results, such as the accuracy of recommendations and time reaction, can be deduced. The other advantage of this method is that it provides answers to qualitative questions (Explicit feedback from users), which gives the possibility of results interpretation.

Its main drawback is the bias of collected data as the subjects (users) know they are recorded. Second, the number of users, participating in the study, should be small to avoid time-consuming. In addition, the sample of users must be representative of the population targeted by the system to allow for good generalization.

Online Methodology. This methodology provides the most concrete results, due to its use by real users in a real environment. Therefore, it requires the deployment of all approaches. In the online evaluation, the comparison is made with respect to the change in user behavior when interacting with different approaches. For example, one approach outperforms another if a user selects its recommendations more often than the other.

An example of this methodology is the A/B test. Having two approaches, usually the current and new versions of the same system, user traffic is randomly distributed between them. The choice to adopt the new approach is made after calculating and analyzing the recorded metrics.

In some cases, these experiments may have a negative influence on the system if the recommendations of the new approach are not relevant. To avoid this risk, it is best to conduct an online evaluation after an offline one, to ensure an average quality of candidate approaches.

Criteria of Evaluation

Given the development of recommender systems in different fields, their evaluation varies according to the nature of the problem. The choice of criteria must be judicious, as each criterion responds to a specific property. The overall performance of the approach is the trade-off of the multiple criteria. We discuss in this section three criteria. The remaining ones can be found in [104].

Accuracy. It is the most discussed criterion in the literature. It represents the utility of an item for a user. Intuitively, the approach preferred by users is the one that has better accuracy. Several metrics are used to calculate it and are divided into two classes:

Prediction accuracy metrics: These metrics measure rating prediction approaches. They measure the error between the real and the predicted ratings [257]. It includes:

- Mean Absolute Error (MAE): measures the average absolute deviation of ratings' errors contained in the test set $\mathcal{T}$:

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} |r_{ui} - \hat{r}_{ui}|$$

  where $r_{u,i}$ is the real rating given by user u to item i and $\hat{r}_{u,i}$ is the predicted rating.

- Root Mean Squared Error (RMSE): it relies on the Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (r_{ui} - \hat{r}_{ui})^2}$$

Extension of these metrics, e.g., Normalized MAE/RMSE, can be also used in specific situations.

Top-N metrics: Initially borrowed from information retrieval, these metrics measure the quality of top-N recommendation lists. Precision and recall are the most used [65]. For a recommended item, four outcomes, shown in the table 2.1, are possible.

Table 2.1: Confusion Matrix

	Recommended	Not Recommended
Relevant	True Positive (TP)	False Negative (FN)
Irrelevant	False Positive (FP)	True Negative (TN)

- Precision: it measures the proportion of recommended items that are relevant. Given a user u and a list of N recommended items:

$$Precision_u@N = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- Recall: it measures the proportion of relevant items that are recommended:

$$Recall_u@N = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

As N increases, recall increases as a larger recommended list can contain more relevant items, but precision decreases. The F1 Score measures the balance between them:

$$F1_u@N = 2 \times \frac{Precision_u@N \times Rappel_u@N}{Precision_u@N + Rappel_u@N}$$

Additional metrics can be used for ranked recommended lists. These metrics measure, in addition to the relevance, the ranking quality of items. We report Discounted Cumulative Gain and Mean Average Precision.

- Discounted Cumulative Gain (DCG) [125]: it considers that better-ranked items give better satisfaction.

$$DCG_u@N = \sum_{i=1}^{N} \frac{rel_{ui}}{\log_2(i+1)}$$

where

$$Pertinence_{ui} = \begin{cases} 1, & \text{if item of rank } i \text{ is relevant for } u \\ 0, & \text{otherwise} \end{cases}$$

- Mean Average Precision (mAP) [263]: It measures the average of all precisions in the positions where an item is relevant.

$$mAP_u@N = \frac{1}{N} \sum_{k=1}^{N} Precision_u@k \times rel_{uk}$$

Other measures of accuracy like MRR, ROC curve, and hit rate can be used [104].

Diversity. It relates to the internal differences within the recommended list. If a list is diverse, the items within it are different from each other. Several metrics are used to calculate it [53, 140], and we report the most used one.

- Intra-list distance (ILD): It is defined as the average pairwise distance of the items in the recommended list $\mathcal{S}$.

$$ILD(\mathcal{S}) = \frac{\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S} \setminus \{i\}} d(i,j)}{|\mathcal{S}|(|\mathcal{S}| - 1)}$$

where d is a distance function.

Novelty. It consists of the recommendation of items that the user is not aware of [135]. In other words, the recommendation does not take into account items that the users interacted with.

2.1.4 Limitations

Although research and development in the field of recommender systems has made considerable progress in terms of improving user satisfaction, due to the improved quality of recommendations, it has also highlighted several limitations that need to be specifically addressed [204]. These limitations are apparent when recommender systems are actually deployed in real-world and industrial applications. For example, the winner of the Netflix Prize [32] was never used

in a real-world system as it was time and hardware expensive to deploy [3]. We enumerate briefly some classical limitation before discussing others in more detail. **Data sparsity** is caused by the interaction of users with a small subset of items. Another limitation is **cold start**. It is caused by the evolution of the system and the constant adding of new items and the arrival of new users. Solutions were proposed to address it [93] but it is still a ubiquitous problem. There are more limitations [204] like **popularity**, or **privacy**. We will focus on the ones we tackle in this thesis.

- Best Recommender Selection: Of all open problems in recommender systems, the challenge of selecting the best recommender is the most important and less studied. Several solutions were proposed to choose the best performer for either a dataset [94], a group of users [84], or each instance of the data [16].
- Static Recommendations: User preferences are changing across time as new items emerge and the temporal temporal dynamics of certain items, e.g., seasonal items. Standard recommendation approaches tend to use all historical user-item interactions to learn each user's short and long-term preferences. This produces static recommendations as it assumes that all users' interactions are equally important which may not be the case in real-world cases. Many solutions were proposed to either incorporate temporal dynamics into standard recommenders [136], use sequence-based models [251], or rely on Reinforcement Learning [272, 178, 271].
- Overspecialization: One of the effects of recommendations on users is to provoke the phenomenon of *filter bubble* [183, 170] which may also cause a polarization phenomenon called *echo chambers* [96]. In fact, recommendation approaches tend to recommend items too similar to the ones the users interacted with. Relying on other optimization objectives constitutes a solution to overspecialization, e.g., diversity [90, 233], novelty [53].
- New recommendation tasks: As discussed in [204], recommender systems are mainly focused on simple and inexpensive items recommendations such as music, movies, or books, while other systems, e.g., educational systems and databses, with atypical items are not studied enough. Designing recommendation approaches for these systems raises different challenges like understanding the domain knowledge and the user decision-making process. Addressing new types of recommendations in different domains may be beneficial in opening many new and interesting research lines.
- Scalability: Many works studied the scalability of recommenders [228]. In fact, these approaches should have the possibility to scale well with the increase in the number of users and items. The second reason is that, when interacting with a real-world system, users expect reactivity when receiving recommendations.

In this section, we proposed an overview of recommender systems, the different approaches, and measures of evaluation. We also highlighted a few limitations and challenges that need to be addressed. In the scope of this thesis, we propose solutions to overcome these limitations. We first propose a meta-learning methodology [67] to solve the best recommender selection problem. Then, to be consistent with real-world environments, we rely mostly on dynamic recommenders. We designed new approaches for different recommendation tasks:

[3] https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429

Test recommendations, SQL query recommendations, and session recommendations relying on Multi-armed bandits, or Reinforcement Learning [236]. Within each task, we leveraged different dimensions that characterize users' behavior in order to avoid overspecialization and filter bubble [183].

2.2 Multiple Hypothesis Testing

The rapid growth of collected data generates the need for advanced analysis methods to extract different insights and patterns between users and the items. One of these methods is Hypothesis testing. It is one of the first statistical inference methods [86] and one of the most used [146]. In this thesis, we used hypothesis testing to verify the existence of patterns of different users' groups. We rely on multiple hypothesis testing as we encounter multiple groups. In this section, we briefly introduce the concept of hypothesis testing. Then we present the multiple hypothesis problem and the different solutions to solve it.

2.2.1 Hypothesis Testing

The main idea of hypothesis testing is to test whether a given data supports a certain quantitative statement, referred to as "hypothesis" [86]. This decision-making involves comparing two different hypotheses: *null hypothesis* (H_0) and *alternative hypothesis* (H_a). Both hypotheses frame distinct observations that are exclusive. For example, in one of the scenarios we test in our work, the hypotheses are:

- H_0: The average ratings of young male users for Hip-Hop music is equal to 3.5 on a scale of 5.
- H_0: The average ratings of young male users for Hip-Hop music is greater than 3.5

The goal of the test is to decide whether the null hypothesis is accepted or rejected. In the case where it is rejected, the alternative hypothesis is accepted. To make that decision, a test is applied on the null hypothesis H_0 with always the assumption that it is true (accepted). The test outputs a p-value that represents the probability of observing more extreme values than the observed ones. It also represents the probability of rejecting a true null hypothesis. The p-value is compared to a threshold value, $\alpha \in [0, 1]$, and the null hypothesis is rejected if and only if p-value $< \alpha$. The value of α is usually set to 0.05 or 0.1 [86]. In the previous example, the p-value of the null hypothesis test is $5.2e^{-7}$ which means that the young male users that are part of the data give an average rating higher than 3.5 for Hip-Hop songs (Accepting the alternative hypothesis). This result represents the type of trends and patterns our solutions presented in Chapter 4 extract from users' groups.

2.2.2 Multiple Hypothesis Testing

In real-world applications, usually, more than one hypothesis is tested. In fact, in our application, as multiple groups are encountered, and each group is linked to one hypothesis, multiple hypotheses may be tested. This may raise the multiple hypothesis testing problem.

The multiple testing problem is an issue in statistical inference that arises when one assesses the significance of multiple hypotheses simultaneously. This challenge is particularly relevant

in scientific experiments [240], medical studies [167], and data-driven investigations [273] where numerous hypotheses are tested concurrently.

The essence of the problem lies in the increased likelihood of obtaining false hypothesis rejections (or false positive errors) when multiple tests are conducted without appropriate adjustments. As the number of hypotheses tested rises, the probability of erroneously rejecting at least one true null hypothesis also escalates, potentially leading to increased rates of false discoveries. For example, in genomics testing [162], researchers may examine the expression levels of thousands of genes simultaneously. Each gene represents a distinct hypothesis, and testing them without appropriate corrections can lead to an increased likelihood of reporting false positives.

The fundamental aim is to draw meaningful findings from data by evaluating whether observed patterns are statistically significant. For example, we want to find all the groups that have an overall rating higher than 4 for a given type of product. The pattern is related to the average of ratings. As multiple groups are involved, we aim to select the real meaningful ones. However, when these multiple comparisons are applied, the likelihood of returning at least one group by chance alone becomes higher, necessitating more attention to the interpretation of individual p-values.

Several methods have been proposed to address this problem, reflecting the importance of mitigating the increased risk of false positives: Family-wise error rate [30], False discovery rate [33], and α-investing [89] procedures.

Family-wise Error Rate Procedures

The Family-wise error rate (FWER) procedures are methods that control the probability of making a false discovery (false positive) when testing multiple hypotheses. It consists in adjusting the significance level α for each individual test. From the several solutions that were proposed to control the FWER, we report the ones that are relevant to our work:

Bonferroni Correction [30]. This method adjusts the threshold α by dividing it over the overall number of comparison tests. If we consider m the number of tested hypotheses, a null hypothesis H_0 is rejected if and only if the p-value of its test is lower than $\frac{\alpha}{m}$. This method is too conservative and potentially leads to an increased likelihood of the rate of false negatives (The accepted hypotheses that are false).

Šidák Correction [225]. This method has the assumption that all the tested hypotheses are independent. A null hypothesis H_0 is rejected if the p-value of the test is lower than $1-(1-\alpha)^{\frac{1}{m}}$. This method is less conservative and stringent than Bonferroni but the difference is slight. For example, by assuming an $\alpha = 0.05$ and $m = 10$, Bonferroni-adjusted level is 0.005 and the Šidák-adjusted level is approximately 0.005116.

SubFamily Correction [254]. This method also controls the FWER where hypotheses are organized into families. One null hypothesis H_0 is to be rejected in each family. The procedure has two steps:

- Step 1: finds r^* as

$$r^* = \underset{r}{\operatorname{argmax}}\left(\sum_{i=1}^{r} p_i^{min} \cdot |F_i| \leq \alpha\right) \tag{2.2}$$

where r is the number of families, F_i is the i^{th} family and p_i^{min} is the smallest p-value in that family. r^* is the perfect number of families.

- Step 2: Reject $h_1^{min}, ..., h_{r^*}^{min}$, where h_i^{min} is the rejected hypothesis of family F_i.

This method is less conservative than the previous ones but has many limitations. The main one is that many significant hypotheses may be part of the same family while other families may contain only insignificant ones.

Other FWER corrections were proposed in the literature like Tukey [241], Holm [117]. The FWER procedure is not without its challenges. Critics argue that stringent control of the family-wise error rate may lead to an increased likelihood of the rate of false negatives, where true effects are missed due to overly conservative adjustments. Moreover, this procedure is hardly related to the overall number of tests. One can not use it when computing an infinite or previously unknown set of tests.

False Discovery Rate Procedures

In recent years, more sophisticated methods have been developed to address the limitations of traditional FWER procedures. The concept of false discovery rate (FDR) has gained prominence, offering an alternative perspective by controlling the expected proportion of false positives among the declared significant results. More precisely, the FDR is the expected ratio of the number of false positives to the total number of rejections of the null. Unlike FWER procedures, FDR procedures allow for a more liberal approach which results in an increase of the test power [221], at the cost of increased numbers of false positives.

Several solutions were proposed to control the FDR. In all of them, the hypotheses are listed and ranked in the ascending order of their p-values. By assuming m hypotheses: H_0, H_1 ... H_m, $p_0 \leq p_1 \leq ... \leq p_m$ (p_i if the p-value of the i^{th} hypothesis). We report the most known procedures:

Benajmini-Hochberg Correction [33]. This correction works in two steps:

- For a given significance level α, find the highest k such that: $p_k \leq \alpha.\frac{k}{m}$
- Reject all the hypotheses that have a rank $i \leq k$: H_0, H_1 ... H_k.

One needs to note that this procedure is valid if and only if all the m hypotheses are independent. Because of this assumption, the following correction was proposed.

Benajmini-Yekutieli Correction [34]. This method is similar to the Benajmini-Hochberg Correction except for the violation of the independence rule. It follows the same two steps where in the first one, the highest k is chosen such that: $p_k \leq \alpha.\frac{k}{m.c}$ where $c = \sum_{i=1}^{m} \frac{1}{i}$

Other methods were proposed to control FDR like Storey-Tibshirani correction [234]. Despite their advantages, FDR procedures are most effective in situations where the identification of potential discoveries outweighs the risk of occasional errors [107]. In the cases, where the control of false positive probabilities has to be more stringent, FWER procedures are better. In addition, similarly to the previous procedure, FDR is hardly related to the overall number of tests. One can not use it when computing an infinite or previously unknown set of tests.

α-investing Procedures

α-investing procedures represent an innovative approach to multiple testing that aims to improve the power of hypothesis testing while controlling the overall error rate [89]. This procedure builds upon the principles of the false discovery rate one and introduces dynamic elements to it.

Instead of fixing a pre-determined level of significance, α-investing adapts the threshold dynamically during the whole testing process. The core idea is to invest statistical significance in promising hypotheses. This adaptability distinguishes α-investing procedure from the conventional approaches discussed before and allows it to capitalize on the variability inherent in large-scale data exploration.

The procedure begins by assigning an initial budget α. As hypotheses are tested, the procedure dynamically allocates parts of the budget to each test based on the evidence accumulated. If the test fails to reject the current null hypothesis, the allocated budget to that test is lost and the remaining unspent alpha is reallocated proportionally to all remaining tests. On the other hand, if a test reaches statistical significance, the invested alpha is reclaimed and can be reinvested in subsequent tests, ensuring continuous adaptability to the evolving evidence.

One of the advantages of this procedure is its adaptability and its independence from a prefixed number of tests. The procedure is well-suited for scenarios where the sizes and patterns are not known in advance. Moreover, it has the potential for a better use of the significance budget α as it prioritizes and invests more in hypotheses that show promise, adapting to the data patterns as they emerge.

Many α-investing policies for investing the budget were proposed in the past [273]. We report the ones that are relevant to our work:

- β-`Farsighted`: it ensures that at each step a fraction β of the available α-wealth is preserved for future tests.
- γ-`Fixed`: it assigns a fixed budget defined as a function of γ for each performed test.
- δ-`Hopeful`: it assigns the whole current available α-wealth to each hypothesis with the *hope* that at least one of the next δ null hypotheses is rejected.
- ϵ-`Hybrid`: it adjusts the budget assigned to the tests based on estimated data randomness and chooses between γ-`Fixed` and δ-`Hopeful` using a threshold ϵ.
- ψ-`Support`: it adjusts the budget of each hypothesis based on its support population, i.e., the number of data points that are used to perform the test.

This methodology is particularly pertinent in the context of large-scale studies as it permits testing hypotheses on the fly. However, this procedure has a few limitations. Its success depends on the policy of budget allocation and the rate of reallocation. In addition to that, some policies rely on different parameters. One should appropriately tune these parameters to ensure a better reliability of the results. Mismanagement of these parameters can lead to an increased risk of false discoveries or, conversely, an overly conservative approach that sacrifices the power of the tests.

In this section, we proposed an overview of the multiple hypothesis testing procedures that adjust the significance level α to minimize the rate of false discoveries. We also highlighted a few limitations and challenges of each procedure. In the scope of this thesis, we relied on these procedures to study the collective behavior of users. We formulated two generic problems that optimize for tests significance and the coverage of all users' groups. We proposed two greedy algorithms by extending FWER and FDR procedures and a heuristic solution by proposing an α-investing policy that maximize coverage. This work was originally published in WWW 2022 [45] and extended to a journal paper published in the Transactions on Large-Scale Data- and Knowledge-Centered Systems [44].

Chapter 3

Individual User Behavior

In this chapter, we explore the analysis of individual user behavior with recommenders. In fact, recommendation systems have permeated our lives and are used by a variety of applications to serve the best content to users. In practice, recommendation systems should capture the change in users' preferences over time to generate relevant recommendations. These systems need to correctly model the interactions of users in states in which they may find themselves. In this chapter, we first investigate the use of users' states to extend standard recommenders and propose a meta-learning solution to the best recommender selection problem in Section 3.1. We then propose different solutions of recommendations in a dynamic context. We explore several real-world applications: Test Recommendation (Section 3.2.1), SQL Groupby Queries Recommendation (Section 3.2.2), and Diverse Session Recommendation (Section 3.2.3).

3.1 Static Recommendations

In this section, we investigate how leveraging users' profiles and users' behavioral states impacts the output recommendations. To do so, we propose to study a less popular recommendation problem: Best recommender selection. In fact, despite the proliferation of recommendation approaches, the question of which recommender works best remains widely open. We propose a methodology based on meta-learning [67] that leverages users' profiles and states and chooses among several recommendation approaches, which one is best suited for predicting the preference of a user for an item. This work was published in IEEE BigData 2021 [42].

3.1.1 Motivation: Best Recommender Selection

Consider an industrial company that wants to integrate a recommender system into its web application. The established approach to determine which recommender to implement to serve users with relevant recommendations is to repeatedly try a pool of existing recommendation approaches and to choose the one yielding the best results according to some predefined evaluation measures. This relies on comparing recommendation approaches by reporting their average mean performance across all test users. One may agree that this brute-force solution is expensive in terms of time, hardware, and human resources. In addition, this only

gives an aggregated measure of which recommendation approach performs best. It does not consider the users and items individually. As an example, assume a pool of 3 recommendation approaches: a deep learning approach `Multi-VAE` [152], `Item-Based Collaborative Filtering` [214], and `Content-Based Filtering` [156]. The company aims to identify which approach to deploy on their data. An empirical study was made and the engineers observed that the `Item-Based CF` approach performs best overall. This study was done by averaging the performances of all users on a chosen evaluation measure. However, the team in charge observed that `Content-Based Filtering` is best for users who leave a lot of textual comments. They also observed that `Multi-VAE` approach is best for users who generate many interactions with items. They concluded that despite `Item-Based CF` being the best overall approach, there exist, users, for which `Multi-VAE` or `Content-Based` perform better. In addition, they made the following observations: (1) recommendations to the same user may differ between approaches; (2) recommendations by the same approach may differ between two similar users; (3) recommendation performance of an approach, measured at the level of all user-item instances, may differ between two comparable users or comparable items.

3.1.2 Best Recommender Selection Challenges

The performance of recommenders largely depends on the chosen experimental protocol, on how data is split and how results are aggregated, and also on the evaluation metrics. It also depends on data characteristics, and on whether user feedback is implicit or explicit. A recent work took a deeper dive into the data and showed that recommendation accuracy highly depends on user and item meta-features [36]. It may also depend on the density of the user-item matrix [84], or the demographics of users [85]. Many years of empirical testing taught us a few lessons. For instance, Item-Based Collaborative Filtering (`IBCF`) generally performs better than User-Based CF when the number of users is much greater than the number of items [214]. Matrix factorization approcaches (such as `ALS` for implicit data) have shown better accuracy results than neighborhood-based models [137]. These intuitions are hard to generalize, and practice has shown that no single approach is best in all scenarios. This complex dependence on data characteristics [6], data splitting and result aggregation, user activity, and item popularity, naturally calls for developing an approach that learns from data to choose the right recommendation strategy for each user-item instance.

A recent research direction for the "algorithm selection problem" in recommendations is *meta-learning* [28, 29, 67]. Meta-learning approaches seek to improve prediction through the use of "side information", typically user and item meta-features. Such systems can be trained and used at different levels. *Collins et al.* [61] identify three different levels: (1) Global-level meta-learners use dataset characteristics to choose the best recommender for each dataset, (2) Mid-level meta-learners select the best recommender for subsets of the data, (3) Micro-level meta-learners select the best recommender for every user-item instance.

The micro-level methodology that was proposed by [61] builds a per-instance meta-learner that predicts errors of each recommender and learns to choose the one that makes the lowest error. However, it has the following limitations: (1) it was applied only to explicit feedback data. For many retailers, rating data is not available and personalization relies on the customers' purchase history or other type of implicit feedback [121]; (2) it was developed solely for predicting ratings and is not suitable for top-N recommendations. This requires redesigning meta-learning to perform the training on ranked lists and use appropriate ranking measures;

(3) Meta-learning was tested only on the Movielens dataset. These observations lead us to formulate three research questions.

- **RQ1**: How does meta-learning generalize to the top-N recommendation task? In practice, users are recommended more than one item at a time.
- **RQ2**: Can we adapt meta-learning to work with implicit feedback datasets and how does it perform? Indeed, the performance of a meta-learner may depend on the type of user feedback.
- **RQ3**: How does meta-learning perform on datasets other than Movielens? This will address the question of whether the performance of a meta-learning approach depends on the dataset.

3.1.3 Our Contributions

To address our questions, we develop a meta-learning approach that relies on training a binary classification model in the case of implicit data and a regression model in the case of explicit data. The application of a binary classification model predicts whether a given recommender is "good" or not for the considered user-item instance. The intuition behind that is dictated by the nature of implicit feedback datasets. When no explicit ratings are available, we handle binary datasets, i.e., for a given user, an item is considered relevant or not. In the case of explicit datasets, we use regression models to predict which recommendation approach is best for each user-item instance. In that case, ratings are used to predict the ranking error between true rankings and predicted rankings by each recommendation approach. The regression models learn to choose the recommendation approach that has the lowest ranking error for each user-item instance.

Our methodology relies on a 3-way data split: the training set is used to train single recommendation approaches; the meta-training set is used to train meta-learners to predict the best performer on each user-item instance; the test set is used to compare the performance of single recommenders against the meta-learned model. For implicit feedback data, the meta-learner is trained with four classification models where for each user-item pair, we label the data in the meta-training set according to the ability of each recommender to predict that item in the user's top-N list. In the case of explicit data, the availability of ratings lets us consider the ranking of items in the meta-training set. We implement four regression models that learn which approach achieves the lowest rank error between the true rank of the item and the predicted rank by each recommendation approach.

3.1.4 Data Model

Let $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ be the set of users and $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ the set of items. We note $R = |\mathcal{U}| \times |\mathcal{I}|$, the rating matrix where each element r_{ui} is the explicit rating which reflects u's preference for item i.

Most research exploits explicit feedback to perform item recommendations. However, explicit ratings are not always available, in particular in retail and some online platforms where user interactions with items are implicit, e.g., purchase, view, like, etc. In this case, the user-item matrix becomes binary and is usually referred to as implicit feedback. We note $P = |\mathcal{U}| \times |\mathcal{I}|$, the implicit interaction matrix where each element p_{ui} is set to 1 if the user interacted with

item i, and to 0 otherwise. In both cases, we note I_u the set of items for which user u expressed positive feedback: a high rating value in the case of explicit feedback, or the event of a purchase/view/click/etc by the user, in the case of implicit feedback.

3.1.5 Meta-learning Methodology

A number of recommendation approaches were proposed in the last years, with a strong focus on Collaborative Filtering approaches, such as neighborhood models, matrix factorization, association rules, and more recently, deep learning. In this section, we describe the pool of used recommenders as well as the meta-learning methodology.

Selected Recommendation Approaches

In this section, we describe representative recommenders which are summarized in Table 3.1. The reader is referred to Section 2.1.2 for more details.

Table 3.1: Overview of selected recommendation approaches

Category	Recommenders	Description and reference
Association rules	`ARM`	Recommends items using association rules [194]
Neighberhood-based	`IBCF`	Item-based k-nearest neighbors [214]
Matrix Factorization	`PureSVD`	SDV-based matrix factorization [65]
	`NMF`	Non negative matrix factorization [268]
	`Implicit-ALS`	Matrix factorization for implicit feedback datasets[120]
Learning-to-rank Matrix Factorization	`BPR`	learns a personalized ranking for every user [202]
Deep Learning	`Mult-VAE`	Variational autoencoders for collaborative filtering [152]
	`SpectralCF`	Spectral collaborative filtering [274]

Our Methodology

Figure 3.1 shows the main components of our meta-learning methodology. Following [61], we split the data into three parts: a training set to train the pool of recommenders and tune their parameters; a meta-training set to train the meta-learner to predict if an item of an instance set is relevant or not; a test set to evaluate the meta-learner against recommendations produced by every single recommender. For implicit data, relevance is estimated as binary values of appearance or not of each item of the meta-training set in the top-N list of each recommender. For explicit data, relevance is estimated using the error rank between the real rank of an item in the meta-training set and the ranking that each recommender gives to that item. These predictions represent the performance feature of the approaches on user-item instances and are used along with data meta-features, to train the meta-learner. Meta-features used in the training and meta-training sets are users' profiles which represent their states. They describe the activity of users as well as their demographics (age, gender). Meta-features also describe items by their popularity.

Meta-learning for Implicit Data: The pool of recommenders is trained with the training set and tuned to find their optimal parameters. This results in one model for each recommender. The model is used to produce a list of top-N recommendations. Each instance in

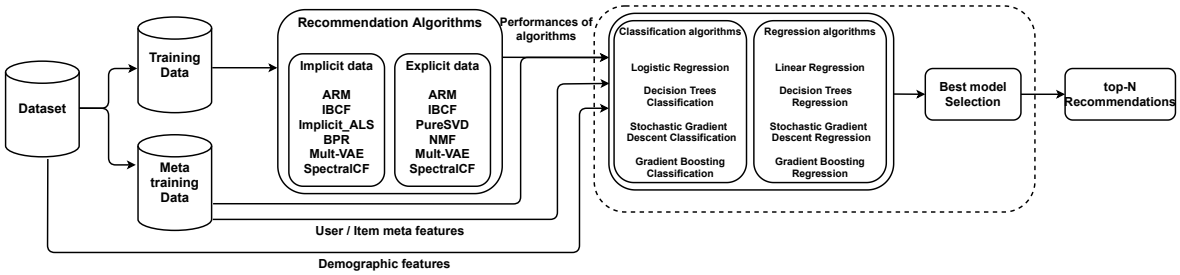


Figure 3.1: Our Meta-learner Architecture

the meta-training set represents a user u, item i, and a recommender, and is assigned 1 if item i is predicted to be in the top-N generated for u by the recommender, 0 otherwise. This data is used as input to a classifier that is trained on user-item instances augmented with meta-features. The training targets are the Boolean values assigned to each user, item, and recommender. We implemented four classifiers: logistic regression [267], decision trees [155], stochastic gradient descent (SGD) [270], and gradient boosting [91]. Finally, using the test set, for each instance we apply the trained classification models. The recommender related to the classification model that classifies the instance as 1 is considered the best. If several models do so, we choose one at random.

Meta-learning for Explicit Data: The methodology is similar for explicit data. We only highlight the differences. Each instance in the meta-training set is assigned a ranking error $|r_i - \hat{r}_i|$, where r_i is the real rank of item i in the meta-training set and $\hat{r}_i$ is the rank which is assigned to item i by the recommender. This data is used as input to regression to learn to predict rank errors. We implemented four regression models and trained them on meta-features. As a result, each meta-learner allows us to predict for each recommender its rank error for a user-item instance. Finally, using the test set, for each instance, we apply the trained regression models for each recommender. The recommender related to the regression model that predicts the lowest rank error is considered the best for that instance. If several regression models predict the same error we choose one at random.

Meta-learning Illustration: Table 3.2 and 3.3 show examples of best-performing recommenders for some instances. Each row represents an instance of the meta-training set characterized by user and item features. These features define the profile (state) of the user. For explicit data, *Rating* is the explicit rating given by the *user* to the *item*. *Rank* is the position of the item in the meta-training set, i.e., for each user we estimate a ranking for items that are in the meta-training set. Table 3.2 contains instances from the meta-training set of implicit data. We can observe that popular items are more likely to be recommended by several recommenders. For instance item *i8* with popularity 1666 is recommended by all recommenders while *i9* with popularity 28 is not recommended by any. Table 3.3 contains instances of the training set for explicit data. Here again, we can see that some algorithms are able to predict ranks that are close to the actual ranking of an item in the top-N of a user. For instance, `ARM` is a perfect rank predictor for *u3-i3* and for *u4-i5* while `PureSVD` and `NMF` are the closest rank predictors for *u2-i2*. Here again, multiple recommenders can be the best/closest rank predictors for the same user-item instance.

Table 3.4 shows examples for which the classification meta-learner outputs the best recom-

Table 3.2: Example of meta-training instances for implicit data. The recommender with a predicted value 1 per instance is highlighted.

	User meta-features				Item meta-features		Recommenders' predictions					
Row	User	Age	Gender	User activity	Item Id	Item popularity	ARM	IBCF-50	Implicit-ALS	BPR	Mult-VAE	SpectralCF
0	u0	>65	M	646	i5	1863	**1**	0	0	**1**	**1**	**1**
1	u1	35-49	M	467	i6	307	0	**1**	**1**	0	0	0
2	u2	<35	M	74	i7	524	0	**1**	**1**	0	**1**	0
3	u3	35-49	M	225	i8	1666	**1**	**1**	**1**	**1**	**1**	**1**
4	u4	35-49	F	268	i9	28	0	0	0	0	0	0

Table 3.3: Example meta-training instances for explicit data. The recommender with the lowest predicted rank error is highlighted.

	User meta-features		Item meta-features				Recommenders' rankings					
Row	User	User activity	Item	Item popularity	Rating	Rank	ARM	PureSVD	NMF	IBCF	Mult-VAE	SpectralCF
0	u0	5	i0	3	3.0	3	**3**	1	**3**	**3**	**3**	451
1	u1	5	i1	11	1.0	4	**4**	2	1	7	**4**	93
2	u2	22	i2	2	4.0	20	11	**19**	**19**	10	33	198
3	u3	15	i3	3	4.0	14	**14**	8	4	11	9	67
4	u0	5	i4	3	5.0	0	13	13	**2**	11	3	44
5	u4	9	i5	9	5.0	0	**0**	1	5	4	2	39

mender per instance. The same user might receive recommendations from two different recommenders for two different items. One can see that for user *u0*, IBCF is best for recommending item *i0* and Implicit-ALS is best for *i3*. Moreover, the same item can be recommended to different users by different recommenders. For example, Mult-VAE is best for *u2-i0* while IBCF is best for *u0-i0*. Table 3.5 shows examples for which the regression meta-learner learned the best recommender per instance. One can see that the regression assigned to Mult-VAE predicts the smallest error for the first row. No other regression model has a better prediction. Mult-VAE is hence the best for the first instance (*u0-i10*).

For both implicit and explicit data, when no recommender predicts an item in the top-N recommendations of a user (e.g., Row Id 3, instance *u3-i3* in Table 3.4), the item is not chosen for that user.

Table 3.4: An illustration of recommendations for implicit data. The recommender with the best-predicted value 1 per instance is highlighted.

	User meta-features		Item meta-features		Meta-learner predictions (classification)					
Row Id	User Id	User activity	Item	Item popularity	ARM	IBCF	Implicit-ALS	BPR	Mult-VAE	SpectralCF
0	u0	646	i0	3445	0	**1**	0	0	0	0
1	u1	467	i1	4756	**1**	0	0	**1**	0	0
2	u2	74	i0	3445	0	0	0	0	**1**	0
3	u3	225	i3	4922	0	0	0	0	0	0
4	u4	268	i4	20	0	0	0	0	**1**	0
5	u0	646	i3	4922	0	0	**1**	0	0	0

Table 3.5: An illustration of recommendations for explicit data. The recommender with the lowest predicted rank error per instance is highlighted.

	User features	Item features	Meta-learner predicted rankings					
Row	User Id	Item Id	ARM	PureSVD	NMF	IBCF	Mult-VAE	SpectralCF
0	u0	i10	2059.9	2808.1	2044.8	3478.1	**1324.1**	4920
1	u1	i11	2975.5	2937.5	**2126.3**	3341.9	2459.2	4710.6
2	u2	i12	**2212.6**	3253.6	2746.1	3171.2	2453.9	4690.4
3	u5	i13	2776.9	2991.4	2392.5	3323.3	**2019.9**	4738.9

3.1.6 Experiments

Setup: We split data in a user-wise fashion, i.e., the split is done for every user, chronologically according to the provided timestamps so that 55% of the data constitutes the training set, 30% the meta-training set and the remaining 15% is assigned to the test set. This choice makes our experiments more realistic and also prevents data leakage [126]. We use the test set to evaluate the meta-learned models against every single recommender. We also compare our meta-models with an ensemble-learning model [227] as well as factorization machines [60]. We note that for single recommenders we use the same proportion of the training set, and tune their parameters using the meta-training set. This ensures that all single recommenders are evaluated fairly against the meta-models with the same size for training and testing.

Data: We used four real-world datasets. Their main characteristics are summarized in Table 3.6. `Retail`, is a proprietary implicit dataset provided by the Marketing department of our industrial partner. The second implicit corpus is a Chinese store transactions data (`TA-FENG`[1]). The two others are explicit data from Amazon[2]: `Amazon_TV` and `Amazon_M` for two product categories *Movies & TV* and *Digital Music*, respectively.

Table 3.6: Characteristics of the datasets

Dataset	Time Span	Number of users	Number of items	Avg #purchases per user	Avg frequency per item	#Unique records	Sparsity
`Retail`	Jan 2017–Dec 2019	11086	3576	234.52	727.04	889062	97.75
`Tafeng`	Nov 2000–Feb 2001	2596	3567	83.02	60.47	175002	98.11
`Amazon_M`	May 1996–Oct 2018	3480	10721	19.24	6.24	66977	99.82
`Amazon_TV`	May 1996–Oct 2018	2257	27479	213.99	17.57	482787	99.22

Metrics: For each user u, we measure **precision** and **recall**. For implicit data, an item is considered relevant to a user if that user purchased it in the test set. For explicit data, relevance is determined by a rating value strictly greater than 3 on a scale of 1 to 5. We also use ranking measures for evaluation: Discounted Cumulative Gain (DCG) and Mean Average Precision (MAP). The reader may refer to Section 2.1.3 for more details about the measures.

Variants: Each recommender requires to finely tune its parameters. Similarly to [70], we

[1] https://www.kaggle.com/chiranjivdas09/ta-feng-grocery-dataset
[2] https://nijianmo.github.io/amazon/index.html

performed a grid search over a set of possible parameters. `ARM`: has no parameters to estimate. For `IBCF`[3], we vary the number of neighbors k in $\{10, 20, 50, 100\}$. For `PureSVD` [3], we set the number of latent factors to 100. For `NMF` [3], we set the number of latent factors to 200 and the number of iterations is set to 2000. The missing values of the user-item matrix are filled with the rating average. For `Implicit-ALS`[4], we set the number of latent factors to 40, the number of iterations to 30, the regularization parameter λ to 0.001 and the confidence constant α to 10. For `BPR`[5]: we set the number of latent factors to 20, the learning rate to 0.001, the regularization parameter to 0.05. For `Mult-VAE` [3], we set the batch size to 500, the number of epochs to 150, and the learning rate to 0.001. For `SpectralCF` [3], we set the batch size to 512, the number of epochs to 400, the embedding dimension to 128, the learning rate to 0.01 and the regularization parameter to 0.001.

We developed a basic stacking model which combines predictions from single recommenders to generate a hybrid prediction. We set the number of estimators to 500, the learning rate to 0.01, the regularization rate to 0.005, and the maximum depth to 3 for the Gradient Boosting Regressor. For the Gradient Boosting classification, the estimators are set to 1000, α to 0.001, regularization to 0.001, and depth to 5. For SGD, we use a *l2* penalty. The number of iterations and α are set to 1000, 0.003 for regression, and 100, 0.001 for classification, respectively. Finally, Decision Trees models have the following parameters: a random splitter, *MSE* criterion (*Gini* criterion for classification), and maximum depth set to 3 (10 for the classification).

We only report results on top-10 recommendations. Similar results were obtained with other values of the recommendation size N, which we also made available in our GitHub[6] repository.

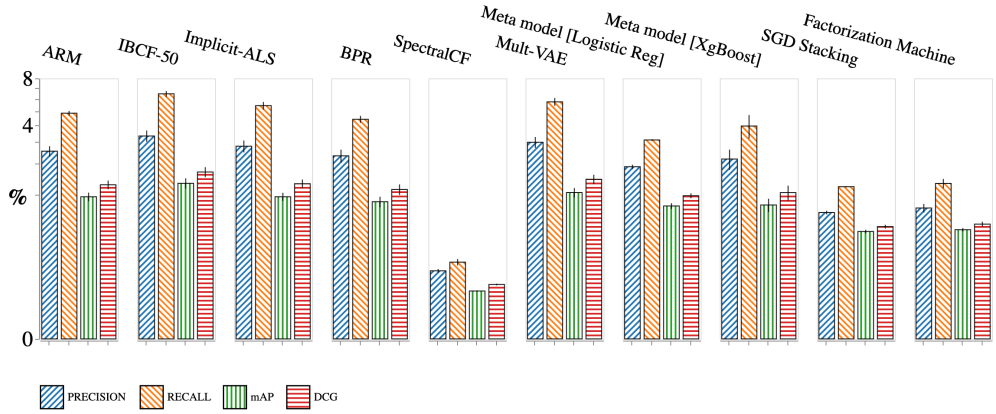


Figure 3.2: Top-10 results of our meta-learner against single recommendation algorithms for `Retail`

[3]https://github.com/MaurizioFD/RecSys2019_DeepLearning_Evaluation
[4]https://spark.apache.org/mllib/
[5]https://implicit.readthedocs.io/en/latest/
[6]https://github.com/meta-model/meta-learner

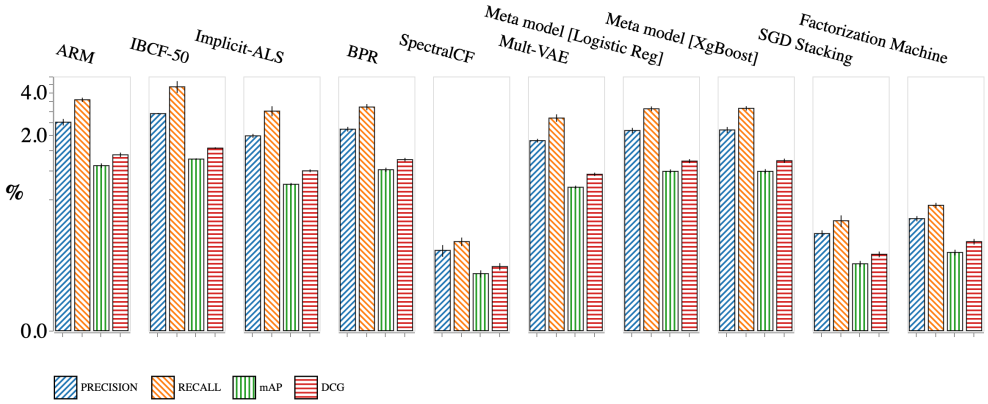


Figure 3.3: Top-10 results of our meta-learner against single recommendation algorithms for `Tafeng`.

Implicit Data

Figure 3.2 shows the results on `Retail` dataset. `IBCF-50` appears to be the best recommender for all metrics. `Mult-VAE` gives good results and performs better than recommenders tailored for implicit data (`Implicit_ALS` and `BPR`). `SpectralCF` is the poorest performer. Similarly to [70], we find that all deep learning methods are outperformed by `IBCF` when it is well-tuned. We note that the ranking-based recommender `BPR` gives poor results compared to `ARM` and the other collaborative filtering recommenders. This rather low performance is surprising, especially since it is specifically designed for implicit feedback datasets. Some recent works [147] and [179] improve over `BPR` to account for the integration of heterogeneous feedback such as clicks and add-to-cart. However, in our dataset `Retail`, the only available feedback is user purchases. One can see from the figure that the two best meta-models outperform the worst baseline (`SpectralCF`). The best meta-model (based on `Gradient Boosting` Classification) outperforms `BPR` and its performances are closely similar to those of `ARM`. We also see that all meta-learners outperform the best stacking model (based on `SGD` Classification) which produces similar results to a factorization machines algorithm.

From Figure 3.3, we see that `Tafeng` results are similar to `Retail` ones except that `Mult-VAE` is outperformed by all "simple" recommenders. One can note that the two best meta-models behave exactly the same. They outperform deep learning recommenders and are close to matrix factorization ones. The best stacking model (based on `Gradient Boosting`) and the factorization machines are still worse than all others (except `SpectralCF`).

The results suggest that for `Retail` and `Tafeng`, the current implementation of meta-learners is unable to accurately classify recommenders according to their prediction power. Another interpretation is that since classifiers are trained on user-item instances, performing top-N recommendations per user is not trivial, and our intuition of ranking items according to the number of recommenders that predict them does not work in practice. Since no explicit ratings are available we cannot evaluate the meta-learning using instance-level evaluation metrics such as root mean squared error ($RMSE$). We tested the approach using regression models on implicit data to see if meta-models perform better than baselines. We generated items' ranks based on their purchase frequencies and consider these frequencies as factors of preference. We trained the same implicit recommenders using the same evaluation protocol.

Results were not as good as the ones reported in Figures 3.2 and 3.3.

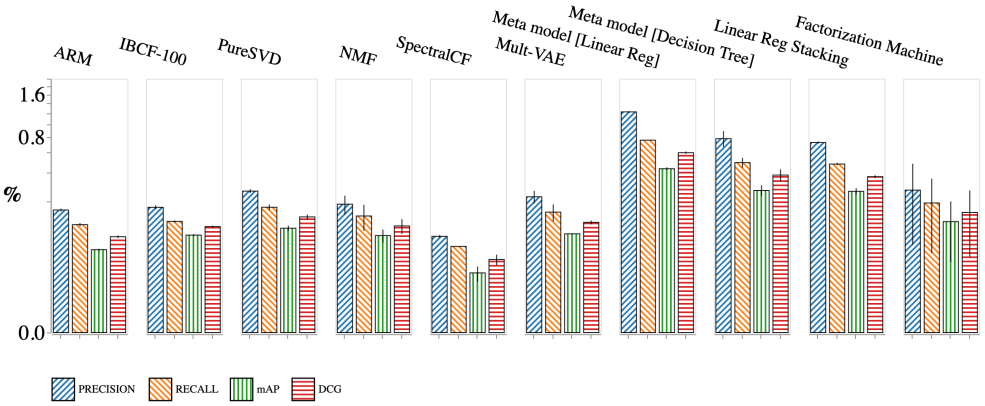


Figure 3.4: Top-10 results of our meta-learner against single recommendation algorithms for `Amazon_TV`

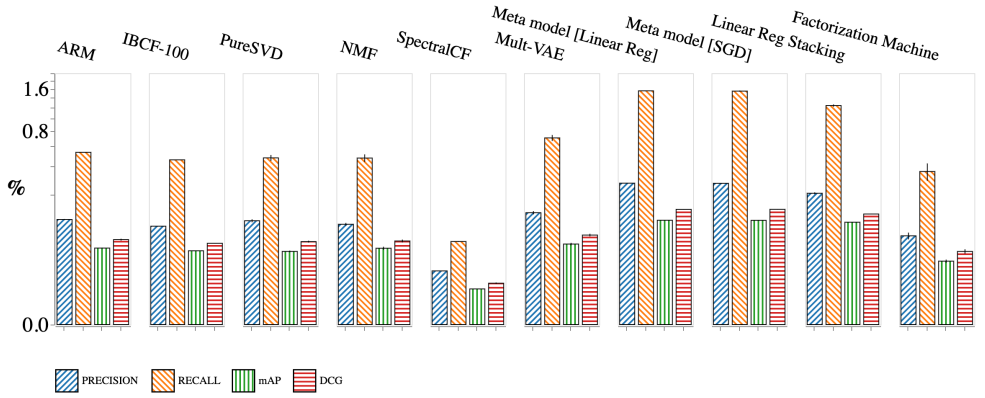


Figure 3.5: Top-10 results of our meta-learner against single recommendation algorithms for `Amazon_M`

Explicit Data

Figure 3.4 shows top-10 performance results on `Amazon_TV`. We note that `Mult-VAE` and matrix-factorization (`PureSVD`) are the best performers. The high performance of `PureSVD` is likely due to the high density of the user-item matrix. The number of interactions per user is 214 items on average. One can note that `NMF` and `IBCF-100` are generally outperformed by `Mult-VAE`. We can also note that `ARM` is more accurate than `SpectralCF` which is still the worst baseline. We also notice that deep learning recommenders are outperformed by "simpler" ones. From the same figure, we note that all meta-models outperform the best baseline for all metrics. We see also that the best stacking model (`Linear Regression`) is better than (`Decision Tree`) meta-model and all baselines. Factorization machines are not as good as the stacking model but still outperform all the baselines.

Figure 3.5 shows top-10 performance results on `Amazon_M`. We note that `Mult-VAE` is the best performer for all metrics, surprisingly followed by association rules (`ARM`). This latter is better than matrix-factorization and neighborhood-based while it was the second worst for

Amazon_TV (see Figure 3.4). This contrasts with the majority of results on explicit data where memory-based and matrix-factorization algorithms are among the best performers [136].

Additionally, association rules are very rarely used in the literature in the context of recommendation systems, with the exception of some works [194] who found that association rules perform better on the purchase data of a French store "La Boîte à Outils". More recently, [36] showed that ARM performs best on users with low activity. This is the case in Amazon_M, where the average number of interactions per user is relatively low. One can note that IBCF-100, PureSVD, and NMF have practically similar performances. SpectralCF is still the poorest performer. Results performed by the best meta-models are better than single recommenders. Results show that the best stacking model (Linear Regression) also outperforms the single recommenders. The stacking model gives performances similar to the best meta-model (Linear Regression). Finally, all baselines, except SpectralCF, are better than factorization machines.

Scalability

Tables 3.7 and 3.8 show training and recommendation times for Tafeng and Amazon_M datasets respectively. Recommendation time is an average of 1080 and 2240 users for Tafeng and Amazon_M respectively. In each case, we report the results of each recommender, the best-performing meta-learner, and the stacking model. Obviously, we can see that training a deep learning recommender is much more expensive than training a standard one. We note that the training times of the stacking model and meta-models are similar and greater than factorization machines. The reason is that both of them have a two-level training policy: the first one consists of training single recommenders and the second one consists of training regression/classification models based on single recommenders' performances. While training a meta-learner takes about the same time as the stacking model, the former is faster than the latter at recommendation time.

Table 3.7: Train and test times of all recommenders for Tafeng.

	Train time (sec)	Test time (10^{-2} sec)
ARM	0.12	0.317
IBCF-50	0.35	1.04
Implicit-ALS	22.8	1.65
BPR	1	2.22
SpectralCF	95	3.44
Mult-VAE	169.04	2.84
Logistic Reg Meta-model	858.83	4.1
SGD Stacking Model	751.75	47.2
Factorization Machines	234.61	18.9

Table 3.8: Test and test times of all recommenders for Amazon_M.

	Train time (sec)	Reco time (10^{-2} sec)
ARM	0.1	1.55
IBCF-100	0.48	5.58
PureSVD	0.82	3.19
NMF	1	4.42
SpectralCF	327.47	7.86
Mult-VAE	175.54	6.77
Linear Reg Meta-model	1020.4	9.21
Linear Reg Stacking Model	1020	27.54
Factorization Machines	48.5	19.39

Summary. In this work, we tackled the question of the usefulness of meta-recommendation by leveraging users' activity statistics and demographics. We assumed that these represent the

profile of users and the states where they might be. We designed a meta-learning strategy that reasons over these profiles based on the most common recommenders. Our proposed strategy based on a meta-learning strategy can be applied to both explicit and implicit data. Our extensive experiments on four real-world datasets: `Retail`, `Tafeng`, `Amazon_M`, and `Amazon_TV` show the high dependency of recommenders on the type of data. The results showed that with explicit data, leveraging multiple recommenders provides more relevant recommendations while the opposite happens with implicit data. This is due to the absence of a clear factor of preferences when data is implicit. This also shows that our implicit meta-model needs further exploration in the future.

3.2 Dynamic Recommendations Applications

After leveraging the meta-learning methodology to investigate the impact of using users' and items' information on standard recommenders and show their importance, we study more realistic problems where users' preferences are dynamic and constantly changing over time. In fact, as discussed in Section 3.2.3, dynamic recommenders capture the shift in users' behavior by modeling their actions into sequences where time is implicitly integrated. As previously introduced, we explore three real-world applications where the environment is dynamic and where the characteristics and dimensions that describe users evolve. In each application, we define different dimensions that capture the attitudes and preferences of users. In this section, and within each application, we explore and propose different dynamic solutions based on the nature of the environment and compare them to ones that may ignore or capture a part of the temporal change. The different applications are:

- **Recommendations for Test Assignment**: Test assignment and upskilling are fast-growing segments of the education economy [180]. Yet, there is little algorithmic work that focuses on crafting dedicated strategies to reach high skills. In this application, we formulate three behavioral dimensions that capture the states of users (students or learners) and their learning. These dimensions are changing based on the correctness of the answers provided by learners. We propose two adaptive solutions that capture the changing in users' dimensions and tend to maximize the learning: the multi-objective Pareto [24] solution and its extension based on Multi-armed bandits [236]. Parts of this work related to the Pareto solution have been published in the 2^{nd} International Workshop on Data Systems Education [46]. The extension related to the Multi-armed bandits solution was recently submitted and is under review.

- **Recommendations for SQL Groupby Queries**: Groupby queries have been the method of choice for Exploratory Data Analysis (EDA) as they provide a birds-eye view of data and return interpretable results. In this application, we propose a solution that guides users in generating and selecting visual analytics based on these SQL queries. Our goal is to encounter the sequence of users' feedback (users' states) as well as previously shown analytics to recommend relevant ones and minimize users' effort. Our solution is based on Multi-armed bandits [236] to balance the exploitation of relevance and exploration of different regions of the data to achieve coverage. A part of this work has been published as a demo paper in CIKM 2021 [69].

- **Recommendations for Diverse Sessions**: Diversity in recommendation has been studied extensively. It has been shown that maximizing diversity subject to constrained

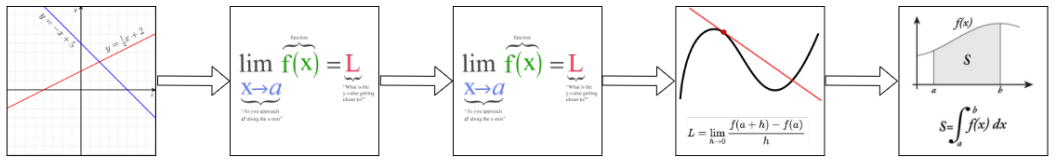
relevance yields high user engagement over time. In contrast to existing work which relies on setting some attributes to define diversity, we propose two solutions that dynamically learn diversity attributes in a multiple-sessions environment. One solution generalizes standard diversity-based recommenders like `MMR` [90] while the second one extends on a Reinforcement Learning architecture, `SMORL` [233]. This work was published in IEEE BigData 2022 [43].

3.2.1 Application 1: Recommendation for Test Assignment

Today, learners engage in self-directed learning, managing many elements of their own study, which, in turn, often requires working on various learning activities independently with less direct guidance from teachers [95]. Consequently, providing guarantees on the quality of learning outcomes is increasingly difficult in these new bite-sized learning structures as they can lead to the so-called illusion of explanatory depth [209] where learners only acquire a superficial understanding of a topic. Ideally, each learner should receive tests chosen in a such way that the learner's skill progresses. Yet, there is little algorithmic work that focuses on crafting dedicated strategies to reach high skills. This should account for the learner's ability to resolve tests based on skill and past performance. That is the topic of mastery learning [188] where the focus of instruction is the time required for different learners to acquire the same competencies and achieve the same level of mastery. This is very much in contrast with classic models of teaching where all learners are given approximately the same amount of time to learn. We illustrate that with an example.

Motivation: Test Assignment

Consider a learner with very basic math knowledge who wants to learn mathematical functions. Figure 3.6 illustrates an example of the learning process. In the beginning, the learner receives tests with a moderate difficulty level of 0.3 for which she provides correct answers. As a result, she incurs no skill gap, and her skill is estimated. This triggers a second step where she is assigned more difficult tests (on limits of functions) for which she fails. In addition to not increasing her skill, she incurs a skill gap. To fill that gap, she is given a second chance with the same type of tests in which she succeeded. Her input is correct and her skill increases. The same process is repeated, and the learner receives more difficult tests on derivatives of functions and then on integrals. She provides correct results and her skill increases.



Difficulty	0.3
Correct	1
No Gap	
Skill Updated	

Difficulty	0.45
Correct	0
Gap on Limits	
Skill Not Updated	

Difficulty	0.45
Correct	1
No Gap	
Skill Updated	

Difficulty	0.7
Correct	1
No Gap	
Skill Updated	

Difficulty	0.9
Correct	1
No Gap	
Skill Updated	

Figure 3.6: Example of the process of learning mathematical functions.

Test Assignment Challenges

Our example identifies several challenges. First, we need to determine which k tests to assign to a learner at each iteration. Existing work on recommending tests optimizes the learner's expected performance either by assuming tests with the same difficulty level [188] or by pre-defining the composition of difficulties beforehand (e.g., by alternating test difficulty levels [158]). Indeed, according to learning theories illustrated in Figure 3.7, simply relying on the learner's expected performance runs the risk of narrowing down the learner into a zone of "boring" and under challenging tests that do not incur upskilling. To address that, we propose to also account for the learner's aptitude, i.e., the difference between the learner's skill and the test difficulty level. This will encourage selecting tests that challenge the learner (the learnable zone in Figure 3.7). Hence, we need to balance expected performance and aptitude. Second, we need to account for the potential skill gap for determining the next k tests. To the best of our knowledge, no existing work encounters all these dimensions. Third, we need to simulate the learners' performance and devise a skill update strategy after they complete a batch of k tests. Based on these challenges, we formulate four research questions.

- **RQ1:** Is the combination of all dimensions well-adapted for attaining mastery and improving skill gain?
- **RQ2.a:** Do different settings of the skill update strategy exhibit different results?
- **RQ2.b:** Does the choice of the learner simulation model impact the skill gain?
- **RQ3.:** Does an application of a meta-strategy that chooses to optimize a subset of dimensions at each iteration, improve mastery achievement?

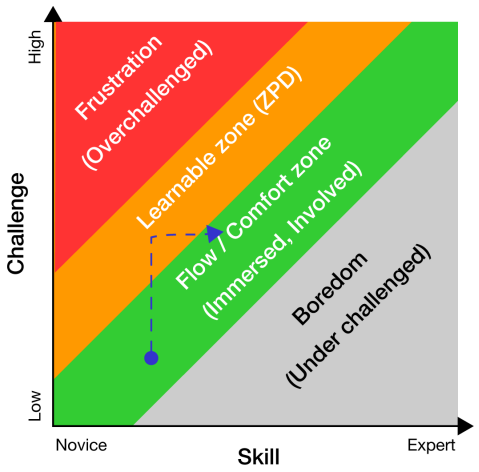


Figure 3.7: Schematic illustration of Zone of Proximal Flow (ZPF) [25], which combines the results of Zone of Proximal Development (ZPD) and Flow Theory. In [25], it is shown that learners improve their skills by completing tests that are more but not too challenging (dotted line).

Our Contributions

We formalize the AdUp Problem, our Adaptive Upskilling Problem as an optimization problem where a learner receives k tests that maximize expected performance and aptitude, and

minimize accumulated skill gap. The combination of these objectives constitutes the novelty of our formalization.

The main challenge in solving AdUp, is its multi-objective nature. We propose to explore two solutions: a Multi-Objective Optimization, referred to as `MOO`, and a Multi-Armed Bandits solution, referred to as `MAB`. `MOO` is addressed by developing a Pareto solution that relies on dominance between k test sets and a *Hill Climbing* [176] heuristic algorithm that finds a subset of the non-dominated solutions [24]. Several variants can be drawn from `MOO` depending on the different compositions between the objectives. A drawback of `MOO` is that all variants optimize exactly the same dimensions over all the assigned batches of tests during the whole learning process. However, it would be desirable to have an approach that learns to find the dimensions to optimize at each iteration. For example, if the learner keeps providing wrong answers to the same tests, favoring the optimization of gap could be more desirable as we need to make sure that the learner successfully completes tests before providing more challenging ones. Therefore, we propose `MAB`, a solution that chooses automatically which of the three optimization dimensions to optimize at each iteration of k tests. We formalize this approach as a multi-armed bandit (`MAB`) problem.

To simulate learners and predict their probability of providing correct answers, we leverage two models for that: an extended version of *Bayesian Knowledge Tracing (BKT)* [64] that leverages test difficulties [182] and *Item Response Theory model (IRT)* [201]. After each iteration, the skill of a learner is updated following existing approaches that aggregate consecutive correct answers [131].

Test Assignment Problem

We consider a learner $l \in \mathcal{L}$ who follows an iterative learning process for a given skill sk. At each step, l completes a set of k tests with different difficulty levels for sk. Each test $t \in \mathcal{T}$ has a skill difficulty d_t that remains unchanged. We associate to each learner l a skill value $l.sk$ that either remains the same or increases as the learner successfully completes tests. The initial value of $l.sk$ can be computed from the information the learner fills in when joining the system (e.g., by completing an initial set of tests or through a pre-assessment questionnaire).

We aim to formalize a problem where at any given iteration, the learner receives a batch of k tests whose difficulty level is greater than $l.sk$. To define our problem, we formalize dimensions that characterize the learning process of a learner l for a skill sk.

Expected performance, aptitude, and gap:

Expected performance. It is the expected performance of learner l for a test t. It is based on the similarity of t with successfully completed tests $l.\mathcal{S} \subseteq \mathcal{T}$ by l and is formalized as follows:

$$exPerf(l, t, sk) = sim(t, l.\mathcal{S}, sk)$$

Aptitude. It quantifies the difference between a learner's skill value ($l.sk$) and the difficulty level of a test t (d_t). It represents the learner's progression ability for the skill when assigned tests that are correctly completed. Aptitude is defined as follows:

$$apt(l, t, sk) = d_t - l.sk$$

Gap. It quantifies the distance between the past failed tests of learner l (set $l.\mathcal{F} \subseteq \mathcal{T}$) and the test t wrt skill sk.

$$gap(l, t, sk) = dist(t, l.\mathcal{F}, sk)$$

Similarity and distance between tests can be computed in several ways. In our implementation, we use the Euclidean distance between the difficulty levels of tests.

The AdUp problem: To achieve skill mastery, we propose an iterative formulation that solves the following problem:

Problem 1 (The AdUp Problem)**.** *Given a learner l, with a skill $l.sk$, find a batch $B \subseteq \mathcal{T}$ of k tests to assign to learner l at iteration i s.t.:*

$$\begin{aligned} &maximize \sum_{t \in B} exPerf(l, t, sk) \\ &maximize \sum_{t \in B} apt(l, t, l.sk) \\ &minimize \sum_{t \in B} gap(l, t, l.sk) \\ &subject\ to \ \ |B| = k \end{aligned} \tag{3.1}$$

Our Proposed Solutions

The main challenge in solving AdUp, is its multi-objective nature. Scalarization is a common approach that transforms the problem into a single objective whereas optimization dimensions are combined via a linear weighted sum. Another approach is the ϵ-Constraint method where a single objective is optimized and the other objectives are constrained with user-specific values [181]. These methods suffer from the need to fix weights or thresholds, leading to sub-optimal solutions. Therefore, we propose to explore two solutions: a Multi-Objective Optimization, referred to as `MOO`, and a Multi-Armed Bandits solution, referred to as `MAB`.

Multi-Objective Optimization (`MOO`): We propose an approach that finds the Pareto solutions by addressing all objectives at once [24]. To do so, we define a dominance relation between two sets of size k.

We represent the set of all test batches as $C_k = \{B | B \subseteq \mathcal{T}, |B| = k\}$. We define batch dominance $B_1 \succ B_2$ between any two sets in C_k:

Batch dominance. We say that B_1 dominates B_2 ($B_1 \succ B_2$) iff:

- B_1 is no worse than B_2 for all three objectives.
- B_1 is strictly better than B_2 for at least one objective.

We design a heuristic Algorithm 1 to avoid an exhaustive exploration of the whole search space. It starts by performing *times* iterations where in each it finds an optimal batch of tests (Lines 3 to 7) to avoid local optimums. At each iteration, it first generates a random candidate. Then it performs *Hill Climbing* to optimize both expected performance and aptitude using Algorithm 2. The returned candidates are added to the set of results. From this set, only non-dominated candidates are kept (Line 8). Finally, the candidate that yields the lowest skill gap is chosen (Line 9) and assigned to the learner (Line 10). The learner's skill is updated

after the completion of the test batch (Line 11). This process is repeated until the learner l achieves skill mastery.

Algorithm 1: Heuristic `MOO`

Input: learner l, set of tests $\mathcal{T}$, size k, number of repetition *times*

```
1  while not mastery do
2      Results ← ∅
3      for n in [1..times] do
4          C ← Random_candidate(k)
5          C* ← HCAE(C)
6          Results.Add(C*)
7      end
8      Keep non-dominated candidates in Results
9      B ← The solution from Results with the lowest skill gap
10     l completes B
11     l.sk ← skill_update(l.sk, B)
12 end
```

Algorithm 2: HCAE - Hill Climbing for Aptitude and Expected Performance

Input: Batch of k tests B
Output: Optimized batch B^*

```
1  while True do
2      Candidates ← ∅
3      for test ∈ B do
4          test_down ← A test with the next lower difficulty
5          B_1 ← B − {test} + {test_down}
6          test_up ← A test with the next higher difficulty
7          B_2 ← B − {test} + {test_up}
8          Candidates.add([B_1, apt(B_1), exPerf(B_1)])
9          Candidates.add([B_2, apt(B_2), exPerf(B_2)])
10     end
11     Keep non-dominated candidates in Candidates
12     if B dominates all candidates in Candidates then
13         B* ← B
14         return B*
15     end
16     else
17         B ← A random candidate from the non-dominates ones in Candidates
18     end
19 end
```

Algorithm 2 searches over all the neighbors of the input batch and selects the one that improves aptitude and expected performance. A neighbor of a batch is computed by replacing one and only one test with another test that has either the next higher or next lower difficulty (Lines 3 to 10). If all neighbors are dominated by the current batch, this latter is chosen as the optimized batch. Otherwise, the algorithm replaces the current batch by randomly selecting one from the non-dominated neighbors.

`MOO` variants. There are multiple solution variants to AdUp: `MOO` as described above; `MOEG`, `MOAG`, and `MOAE` optimize expected performance and gap, aptitude and gap, or aptitude and ex-

pected performance respectively; MOG, MOE, and MOA optimize gap only, expected performance only, or aptitude only respectively. Similarly to Algorithm 1, the bi-objective variants are also based on *Hill Climbing* to explore the space of batches and find potential candidates. In the case of MOAE and MOEG, the *Hill Climbing* optimizes expected performance. The condition in Line 9 (Algorithm 1) relates to aptitude for MOAE and gap for MOEG. On the other hand, for MOAG, the *Hill Climbing* optimizes aptitude, and Line 9 remains unchanged.

Multi-Armed Bandits Algorithm (MAB): A drawback of the previous solution is that all the variants optimize exactly the same dimensions over all the assigned batches of tests during the whole learning process. However, it would be desirable to have an approach that can learn to find the dimensions to optimize at each iteration. For example, if the learner keeps providing wrong answers to the same tests, optimizing gap solely could be more desirable as we need to make sure that the learner successfully completes these tests before providing more challenging ones. On the contrary, if the learner answers correctly in the last batches of tests, it might be better to optimize aptitude so that the learner gets challenged with more difficult tests as she has no gap in her learning process. Therefore, our goal is to design an approach that chooses automatically which of the three dimensions will be optimized at each iteration of k tests. We formalize this approach as a multi-armed bandit (MAB) problem.

The goal of MAB is to verify if a meta approach could be used to address the AdUp problem. The meta approach chooses, at each iteration, an optimization variant of our problem, i.e., bi-objective, or multi-objective optimizations, to generate k tests. We formalize that as a multi-armed bandit problem where each arm corresponds to an optimization variant and the reward r_i, at iteration i, for each variant v is defined as the speed of skill progression:

$$r_{iv} = \frac{\sum_{\forall \text{iterations } j, j<i} \text{skill gain offered by } v \text{ at iteration } j}{\#\text{time the variant } v \text{ was chosen}}$$

At each iteration, the progression speed of each arm is computed and the one with the highest is selected. In the case where an arm has never been selected before, its speed is set to zero. The batch of k tests is then generated based on the variant of the chosen arm.

MAB variants. We implemented different multi-armed bandit strategies [236]: ϵ-GREEDY that chooses randomly a variant with an ϵ probability, THOMPSON Sampling which selects the arm with probability equal to the probability of it being optimal, the upper confidence bound (UCB) which combines the reward and an uncertainty measure with a confidence degree and SOFTMAX which relies on Boltzmann distribution with temperature (τ).

Experiments

The goal of our experiments is to address the following research questions:

- **RQ1:** Is the combination of all optimization dimensions well-adapted for attaining mastery and improving skill gain?
- **RQ2.a:** Do different settings of the skill update strategy exhibit different results?
- **RQ2.b:** Does the choice of the model of simulation impact mastery and skill gain?

- **RQ3:** Does an application of a meta-strategy that chooses to optimize a subset of dimensions at each iteration, improve mastery achievement?

Data: We use real data collected from a Czech educational system[7]. It is an adaptive practice system for elementary arithmetic tests. The data contains more than 1800 tests from which we infer 42 distinct difficulty levels ranging in $]0, 1[$. We assume this order of difficulty level: "divisions" > "multiplications" > "subtractions" > "additions" > "numbers". We consider that all tests for "numbers" have the lowest difficulty (0.13). The difficulty ranges of "additions", "subtractions", "multiplications", and "divisions" are [0.2, 0.4[, [0.4, 0.6[, [0.6, 0.8[, and [0.8, 1[respectively. Within each difficulty range, we assume that multi-digit operations are more difficult than single-digit ones, and tests displayed with visual examples are simpler than directly written tests.

Skill update and mastery achievement: At each iteration and after the completion of a batch B of k tests, we update the skill of learner l as follows:

$$skill\_update(l.sk, B) = max_{sk \in D \cup \{l.sk\}} sk \tag{3.2}$$

where D is the set of difficulty values of correctly completed tests for which all tests with lower difficulties were correctly completed.

To show the intuition of this strategy, we consider a learner with $l.sk = 0.3$ at iteration i. At the next iteration $i + 1$, the learner is targeted with $k = 3$ tests t_4, t_5, and t_6 having 0.35, 0.4, and 0.45 as difficulty levels respectively. We consider that the learner correctly answered t_4 and t_6 and failed t_5. Using our strategy, the skill value $l.sk$ is updated to 0.35 (difficulty of t_4). The correct completion of t_6 is not considered as there exists one test (t_5) with a lower difficulty that was wrongly completed. To account for variability in learners' answers, we used the static mastery detection method NCC [131] that updates the skill if the number of consecutive correct answers, for a given difficulty level is N. For mastery achievement, we consider that learners attain mastery when their skill can not be further improved and equals the highest difficulty level.

Learner simulation: There exist several models to simulate learners and predict the probability of correct completion. In this work, we use two of them.

The first model simulates learners using an extended version of BKT (KT-IDEM) that takes into account the difficulty level of tests. BKT is a cognitively diagnostic form of assessment that has been recognized as beneficial to learners and instructors [182]. It models the learning process given the chronological sequence and correctness of tests. It infers the knowledge of learners by predicting the probability of learning. In addition to this inferred probability, two more probabilities are used to estimate the performance of the learner: Guess and Slip probabilities. Guess is the probability of correctly answering a test when the learner does not master the difficulty while Slip is the probability of incorrectly answering a test even if the learner masters the difficulty. If the test is easy, the probability of Guess is high. If the test is hard, the probability of Slip is high as the learners are likely to make mistakes [182]. We used the implementation of [21] in our experiments.

The second model simulates learners based on latent factors [54]. The probabilities of the next tests are calculated by applying a sigmoid function and learning a logistic regression.

[7] https://github.com/adaptive-learning/matmat-web/blob/master/data/data_description.md

One method, AFM [54], infers the probability by characterizing the learner and the difficulty of tests with two distinct parameters. Another method, PFM [184], extends AFM by integrating the number of successes and failures as parameters in addition to previous ones. Other latent models are based on Item Response Theory (IRT) [201], a traditional cognitive diagnosis model [145]. The simplest version [201] predicts a probability of a binary answer (correct/incorrect) by assuming a unique internal parameter for each learner. In addition, it defines tests with one parameter (difficulty) [199], two parameters (the number of correct answers and difficulty) [39], or three parameters (probability of correct answer) [157]. In our experiments, we used this last method based on the implementation of [37]. The reason is that compared to other latent models, it incorporates the probability of guessing in addition to the difficulty and the number of correct answers.

BKT and IRT are structurally different as BKT captures the learning as a chronological process while latent models do not capture the temporal dimensions. They are trained differently as BKT uses Expectation Maximization (EM) algorithm [21] and IRT uses Adam [37]. Despite these differences, both BKT and latent models infer the probability of correct answers and simulate the learning by capturing similar concepts: the difficulty of tests, the level of learning, and the probability of guessing the correct answers.

Variants and Metrics: We compare `MOO` and its variants as well as `MAB` and its variants described in Section 3.2.1. We also consider `ALTERNATE`, a state-of-the-art approach that assigns a random set of k tests whose difficulty levels alternate in a round-robin fashion: k easy then k medium then k hard tests [158]. We report (1) the average skill gain i.e. the difference between the last and first skill values for all simulated learners, and (2) the average skill progression i.e. the average skill evolution from iteration to iteration. To better understand this first experiment, we examine (3) the percentage of learners who attained mastery and (4) the average number of iterations required to attain mastery. Finally, we compute (5) the average time to generate a batch of k tests.

We set the maximum number of iterations to attain mastery to 500. We vary the value of k in $\{3, 5, 10, 15, 20\}$ and the number of simulations, i.e., learners, in $\{50, 100\}$. Due to limited space, we only report results of 100 simulations with $k = 3$. We refer the reader to our GitHub repository[8] for our complete results and code for reproducibility.

RQ1. Impact of optimizing all dimensions: To verify the impact of optimizing all dimensions, we use BKT (KT-IDEM) [182] and assume $N = 1$ in the skill update strategy. We consider two settings: fixed initial skill value and variable initial skill value.

Fixed initial skill value. We assume the same fixed initial skill value for all learners and consider that learners attain mastery when their skill equals the highest difficulty level. We set the initial value to the lowest difficulty level in our simulated data.

Skill gain and progression. Figure 3.8 reports the average skill gain. We observe that `MOO` and `MOAE` produce the highest average skill gain. Surprisingly, `ALTERNATE` seems to also produce a high skill gain. To elucidate that, we plot Figure 3.9 to examine the average step-wise skill progression. Here again, we observe that `MOO` and `MOAE` results in the fastest upskilling with a clear advantage for the former. `MOAG` is slower but still faster than `MOEG`. This reinforces our initial assumption that optimizing for all three objectives at once yields the best

[8] https://github.com/AdaptiveUpskilling/AdUp.git

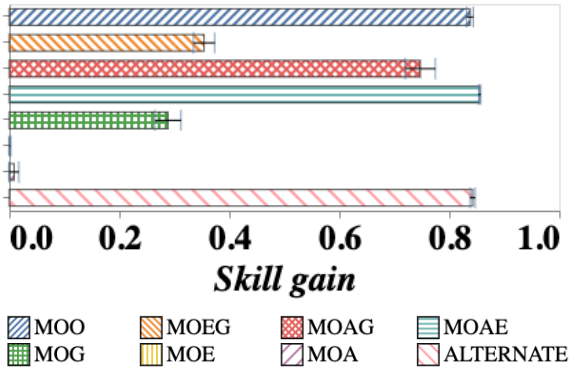


Figure 3.8: Average skill gain for each variant with a fixed initial skill for learners.

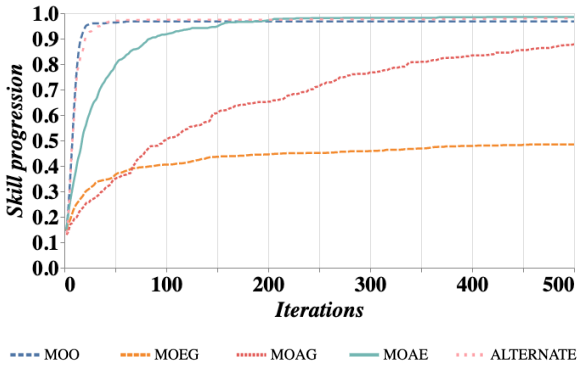


Figure 3.9: Skill progression as a function of # iterations with a fixed initial skill for learners.

results. It also shows that alternating task difficulties yield good skill gain and progression. Therefore, in the next experiment, we examine whether `ALTERNATE` compares favorably to `MOO` and `MOAE` in terms of achieving skill mastery.

Mastery. Figure 3.10 (a) reports the number of times each variant attained mastery. One can see that while `ALTERNATE` reaches a reasonable mastery level ($\approx$ 59%), it is much lower than `MOO`, `MOAG` and `MOAE` ($\approx$ 90%). This clearly confirms that aptitude plays a central role in attaining mastery. Hence, while alternating test difficulty levels in `ALTERNATE` does achieve good skill gain and skill progression performances, it is capped in terms of mastery level since it does not explicitly optimize aptitude. We can also observe that single-objective variants rarely attain mastery. This experiment confirms our initial assumptions: `MOE` assigns tests that are similar to the ones the learner completed correctly, thereby staying within the under-challenging zone [249]. `MOA` assigns tests that are too difficult and that keep the learner in a frustration zone [249].

Figure 3.10 (b) shows the average number of iterations to attain mastery for each variant. One can observe that `ALTERNATE` attains mastery in a similar number of iterations as `MOAG` but has a lower rate of mastery. Nevertheless, it is quicker than all single-objective variants. As explained before, these variants narrow the learners into zones where their skill value does not evolve while `ALTERNATE` offers more challenging batches which allow learners to attain mastery more often. However, simulated learners under `ALTERNATE` are able to correctly complete difficult tests but are unable to do so for the most difficult tests. While `MOAE` attains a slightly higher mastery level than `MOO`, it is outperformed by `MOO` in terms of the number of iterations needed to achieve mastery.

Response time. Time experiments have shown that single-objective variants are obviously the fastest to generate a batch of k tests (Figure 3.11). `MOO` has the worst time average as it has to optimize three objectives ($\approx$10 seconds). `MOAE` would be a good candidate since it runs faster than `MOO`. However, `MOO` does better than `MOAE` on skill progression and on the average number of iterations needed to attain mastery. Therefore, we will need to focus on improving response time for `MOO` in future work.

Variable initial skill value. We study the case where learners have different initial skill values and consider that a skill is mastered when the skill gain attains a fixed value. We set

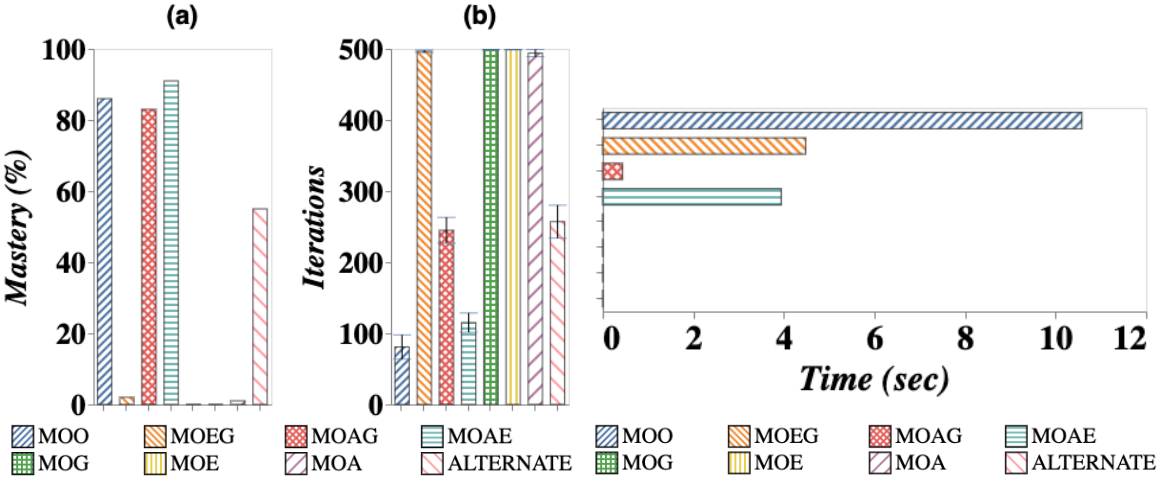


Figure 3.10: **(a)** Percentage of learners who attain mastery - **(b)** Average number of iterations to attain mastery.

Figure 3.11: Average time for generating one batch.

the value to 0.4. We report only bi-objective and `MOO` variants in addition to `ALTERNATE` as we showed already that single-objective solutions are inefficient.

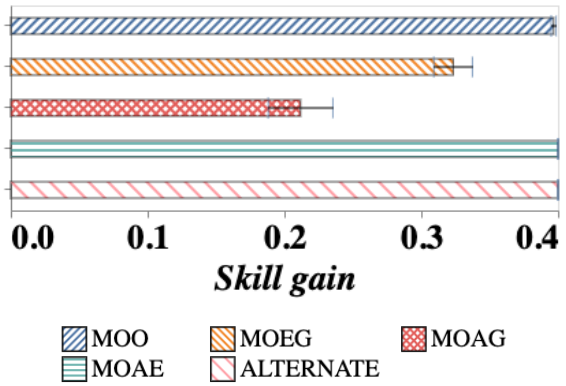


Figure 3.12: Average skill gain with variable initial skills for learners.

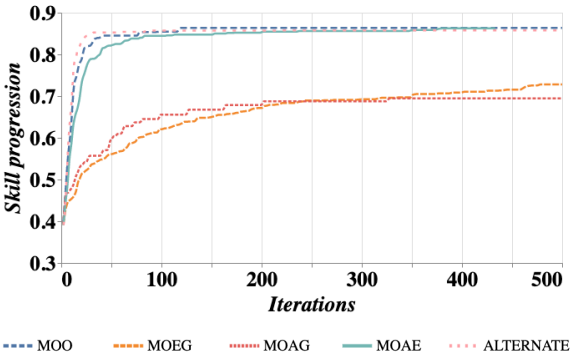


Figure 3.13: Skill progression as a function of # iterations with variable initial skills for learners.

Skill gain and progression. From Figure 3.12, which reports the average skill gain for all variants, we note that similarly to the case of a fixed initial skill value, `MOO`, `MOAE`, and `ALTERNATE` offer the highest skill gain that is equal to the maximum value (0.4). Figure 3.13 also generalizes previous results by showing that `MOO` and `ALTERNATE` skill progressions are the fastest followed by `MOAE`.

Mastery. Figure 3.14 shows the percentage of mastery attained by each variant as well as the number of iterations needed to attain mastery. One can confirm that, despite a smaller gain value to attain mastery, optimizing aptitude is still necessary as `MOEG` is the worst performer for the number of iterations and the second worst for mastery. We also see that `ALTERNATE` has comparable results to `MOO` and `MOAE` which confirm that it is capped in terms of mastery.

Obviously, we can see that all variants attain mastery more often and in fewer iterations than when initial skills are fixed. This is due to the fact that in the latter learners must achieve a much higher skill gain to attain mastery.

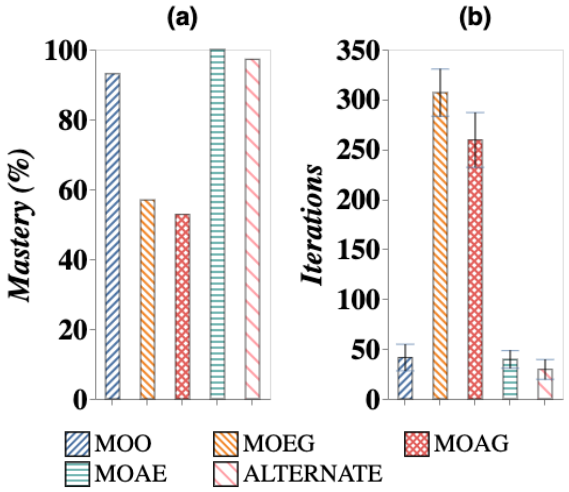


Figure 3.14: **(a)** Percentage of learners who attain mastery - **(b)** Average number of iterations to attain mastery using variable initial skills for learners.

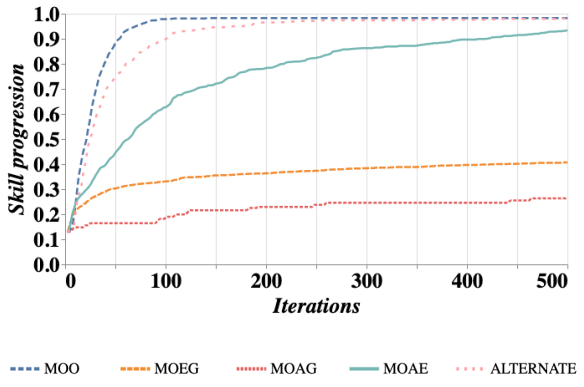


Figure 3.15: Skill progression as a function of # iterations with $N = 3$.

Findings. This experiment shows that combining all objectives yields the highest skill gain which permits a higher mastery in fewer iterations. It also shows challenging learners and optimizing aptitude are beneficial to attain mastery. These results also confirm the ZPD and Flow theories [25] and show the importance of leveraging aptitude and challenging learners.

RQ2.a. Impact of changing the settings of the skill update: We report skill and mastery results by further challenging the learners during the skill update. We increase the value of N, the number of consecutive correct answers, to $N = 3$. We report results in the case where the initial skill is fixed.

Skill gain and progression. The average skill gain is similar to the one reported in Figure 3.8 where `ALTERNATE` is comparable to `MOO` and `MOAE` best and second best variants respectively. Figure 3.15 shows the average progression of the skill. We see that the progression is slower than the one presented in Figure 3.9. This is intuitive as learners have to answer correctly three tests of the same difficulty level to see their skill updated while previously one correct answer was enough. The second observation is that `MOO` is still the best variant with a clear advantage compared to `ALTERNATE` and `MOAE`. This means that `MOO` is less affected by the different values of N than other variants.

Mastery. The results show that more than 90% of learners attain mastery under `MOO` while less than 70% achieve it under `ALTERNATE` and `MOAE`. We also see a small decrease in the mastery rates of `MOAG` and `MOEG`. Results also show that `MOO` is the fastest as it offers learners fewer iterations to reach the highest difficulty level. These results confirm that `MOO` is not

affected by different settings of the skill update strategy.

Findings. This experiment finds that MOO is not sensitive to varying different settings of the skill update strategy.

RQ2.b. Impact of changing the learner simulation model: We report the results of the same metrics using a different learner simulation. We used item response theory (IRT) as explained in the settings. We report results where the initial skill value is similar to all learners. Similar results were observed when the initial value is different from one learner to the other.

Skill gain and progression. Figure 3.16 reports the average skill gain for the variants that performed well previously. We observe that MOO and MOAG along with ALTERNATE produce the highest skill gain. One can also note that, similarly to the case of KT-IDEM, MOEG is the worst bi-objective variant. The main reason is that the test batches of this variant do not challenge the learners as it does not optimize for aptitude.

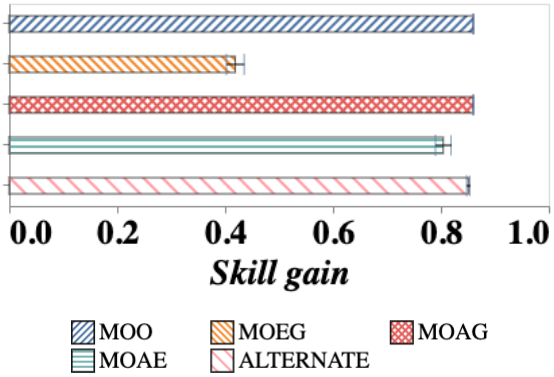


Figure 3.16: Average skill gain using IRT.

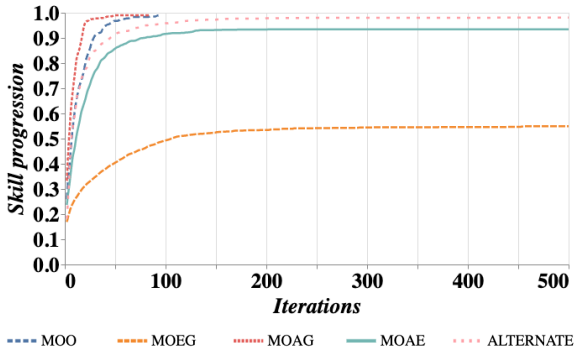


Figure 3.17: Skill progression using IRT.

Figure 3.17 shows the step-wise skill progression. We observe that MOO and MOAG are the fastest in terms of upskilling outperforming ALTERNATE which was equivalent to MOO under BKT. One can also observe that MOEG is the slowest. Next, we compare these variants in terms of mastery.

Mastery. Figure 3.18 shows the average rate of mastery achieved by each variant as well as the average number of iterations to attain it. From figure (a), we observe that the state-of-the-art alternating solution (ALTERNATE) achieves a high mastery level (≈80%) but it is clearly outperformed by MOO and MOAG. This experiment confirms that aptitude is required to attain a high rate of mastery as we see that MOEG is the worst variant (It attains mastery in ≈7% of time). From this figure, we can also observe that MOAE is outperformed by MOAG while it was better under the BKT model. A hypothetical explanation is related to the internal design of both methods. BKT formalizes the learning process as a hidden Markov model where tests' completion is viewed as a chronological sequence and where the different parameters are learned using the correctness of tests. In this case and intuitively, recent learner performances on recently assigned tests appear to be more influential than older tests while in the case of IRT, and because of the absence of time dimension, all performances are equivalent having the same weights. Usually, as the gap is related to earlier tests, IRT seems to give more

attention to it than BKT. Another possible explanation is that BKT tends to overestimate the importance of failure as reported in [184]. In that work, it was observed that BKT tends to predict worse performance after an incorrect answer. One can then make a hypothesis that BKT is negatively biased towards gap in contrast to latent factors models.

Results from Figure 3.18 (b) are inversely proportional to the ones depicted in Figure (a). Variants with the highest mastery percentage are the quickest to attain it. Inversely, the variants that attain a lower rate of mastery are the slowest.

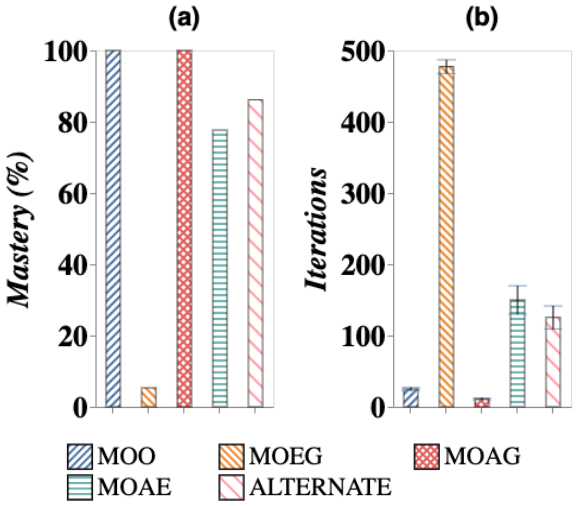


Figure 3.18: **(a)** Percentage of learners who attain mastery - **(b)** Average number of iterations to attain mastery using IRT.

Findings. This experiment finds that IRT generalizes the results of KT-IDEM. In this case, we also observe that `MOO` offers the highest rate of mastery. Optimizing aptitude remains essential as `MOEG` is the worst variant. Despite the differences between BKT and IRT, one can explain their similar results with the fact that they both assume a guessing probability of correct answers and characterize tests by their difficulties. They both infer the correctness probability by approximating the knowledge of the learner based on previous correct answers. In addition, prior work [101, 207] has shown that these models exhibit similar prediction accuracy. However, from these results, we see that the main difference between the two learner simulation models is that IRT tends to favor gap as `MOAG` is comparable to `MOO` while BKT favors expected performance as `MOAE` was the second best. We believe that further research needs to perform a more detailed comparison to understand why these two models offer the same predictions.

RQ3. Impact of the meta-strategy: We seek to verify whether choosing automatically a subset of learning dimensions to optimize at each iteration improves mastery and skill progression compared to optimizing fixed dimensions throughout the process. We implemented the four MAB strategies described in Section 3.2.1 and tested them with a fixed initial skill and $N = 1$.

Skill gain and progression. Figure 3.19 shows the skill gain offered by the different `MAB` strategies. The blue and pink lines represent the skill gain attained by learners under `MOO` and `ALTERNATE` respectively (As shown in Figure 3.8). One can see that `UCB` and `THOMPSON`, and

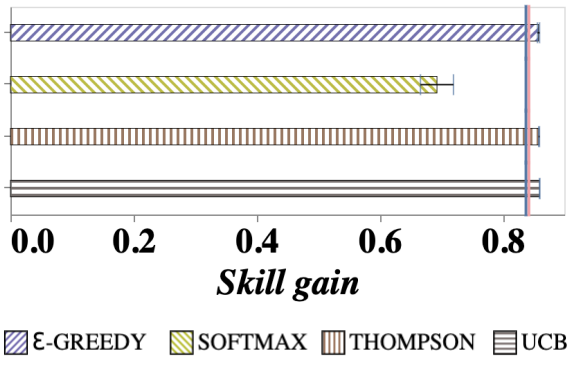


Figure 3.19: Average skill gain using MAB strategies.

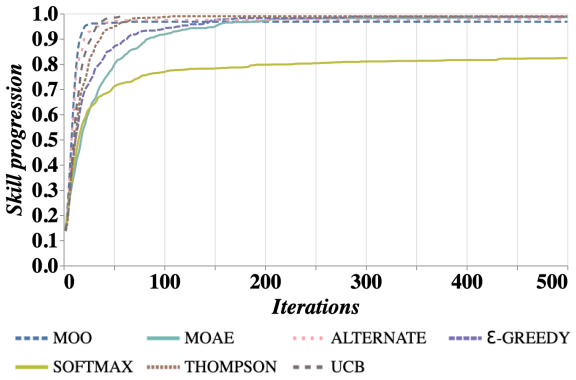


Figure 3.20: Skill progression of learners using MAB strategies.

ϵ-GREEDY strategies slightly improve skill gain compared to MOO and ALTERNATE. We also see that SOFTMAX is the worst strategy showing that probability-based MAB is not adapted to this context. Similar results can be observed in terms of skill progression in Figure 3.20. We can see that UCB and THOMPSON are as faster as MOO while ϵ-GREEDY a slightly than MOAE in terms of progression.

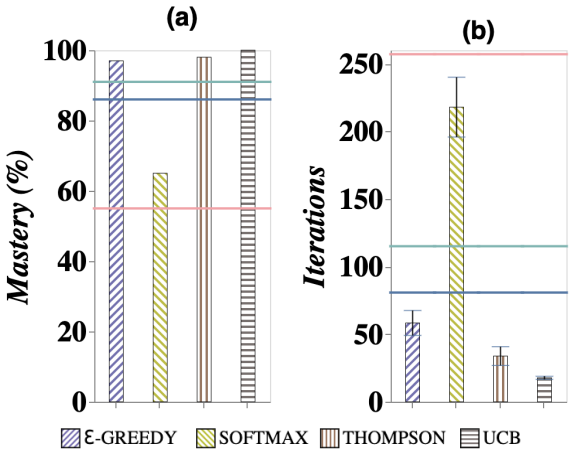


Figure 3.21: **(a)** Percentage of mastery attained - **(b)** Average number of iterations to attain mastery using MAB strategies.

Mastery. Figure 3.21 shows the percentage of learners that attained mastery and the average number of iterations to achieve that. The blue, cyan, and pink lines represent the results of MOO, MOAE, and ALTERNATE respectively. One can see that MAB based on UCB is the best performer and outperforms all other MAB strategies as well as previous variants for mastery. It also achieves that in fewer iterations. We can also see that ϵ-GREEDY and THOMPSON attain more mastery than ALTERNATE, MOO, and MOAE. In addition, they are also better than the baselines in terms of iterations. This confirms our previous assumption that selecting the optimized dimensions during the learning process is better than optimizing fixed dimensions.

Response time. Time experiments have shown that MAB strategies are faster than MOO to

generate the batches but are slower than bi-objective variants. Intuitively, MAB is better than MOO as it may leverage two objectives in a few iterations while MOO optimizes all three objectives during the process.

Insighs on Combining Dimensions. One can see that bi-objective and multi-objective variants are a special case of a MAB strategy where only one arm is available and chosen. Based on that, one can ask the question of whether the outcomes policies of MAB are relying on only one or two variants, for example, they leverage both best variants MOO and MOAE. To answer that, we examine the policies output by MAB.

First, we examine the overall proportions of the selection of each variant in each strategy. The results show that the best strategies (the ones exhibiting the highest mastery rates in lower iterations) UCB and THOMPSON, exhibit a more uniform use of each variant. For example, in UCB each multi-objective variant is selected $\approx 25\%$ of the time. In contrast, we see that SOFTMAX, the worst strategy, relies mainly on two variants, MOEG with $\approx 84\%$ and MOAE with $\approx 13\%$ of the time. This may be the reason for its underperformance. Another interesting insight is that ϵ-GREEDY selects MOO just 9% of the time. This also explains why ϵ-GREEDY has a higher number of iterations and a slightly slower skill progression.

Analyzing these proportions in more detail showed that UCB is more stable and less noisy in selecting the different variants across all simulations. For example, by calculating the standard deviation of MOO selection proportions we found that UCB has the lowest value (≈ 0.04) while SOFTMAX has the highest one (≈ 0.4). This means that the choice of MOO in UCB is similar from one simulation to another while for SOFTMAX this choice looks more random and more noisy.

We now examine the veracity of the hypotheses we made in Section 3.2.1 when we introduced the MAB solution. We assumed that after failing tests, it is more desirable to optimize gap. We also assumed that after obtaining successful answers, aptitude is optimized. Our results show that all strategies tend to leverage gap, in the next two iterations, after learners failed to increase their skill value. For example, UCB and THOMPSON optimize gap 77% and 72% of the time after wrong answers while SOFTMAX does the same 67% of the time. Similarly, our simulations show that UCB, THOMPSON, and ϵ-GREEDY optimize aptitude after successful tests more than 75% of the time, while it is no more than 58% for SOFTMAX. These results also provide insights on why SOFTMAX under-performs compared to the other strategies.

Findings. This experiment finds that choosing automatically the dimensions to optimize at each iteration improves the rate of mastery and the number of iterations needed to achieve it. This justifies the use of a meta-strategy to learn the best combination of objectives to optimize at each iteration.

Summary. In this application, we tackled the question of adaptive upskilling following a mastery learning approach. The originality of our approach lies in adapting the difficulty of tests to the learner's predicted performance, aptitude, and skill gap. We assumed these dimensions to characterize the learners and their behavioral evolution. Based on that, we proposed two approaches: MOO that directly solves our problem and a MAB that chooses among different optimization variants at each iteration. We tested the impact of optimizing these dimensions on skill progression and mastery achievement using a simulated dataset based on

a real one [9]. We also tested the impact of different learner simulation models on mastery achievement. Our experiments showed that `MAB` offers a higher mastery rate and a better final skill gain than `MOO`. Parts of this work have been published in the 2^{nd} International Workshop on Data Systems Education [46]. Its extension was recently submitted and is under review.

3.2.2 Application 2: Recommendation for SQL Groupby Queries

Data summarization provides a bird's eye view of data and groupby queries have been the method of choice for data summarization. Such queries are a useful way to provide declarative and interpretable Exploratory Data Analysis (EDA). They provide the ability to group by some attributes and aggregate by others, and their results can be coupled with visualization to convey insights. As the number of possible groupby queries that can be computed over a dataset is quite large, one naturally calls for developing approaches to aid users in choosing which groupbys best summarize data. Notable behavior analytics systems based on queries were previously proposed such as Qualtrics[10], AIDE's [79], and others [255, 245]. These either require high data expertise, high user effort, and knowledge of the underlying data distributions or predefine reference queries and attributes and rank them based on their interestingness. We consider the setting where a non-expert user wants to explore a large dataset by interactively generating analytics (panels) where each one maps the results of a groupby query to visual elements. We illustrate that with an example.

Motivation: SQL Queries Recommendation

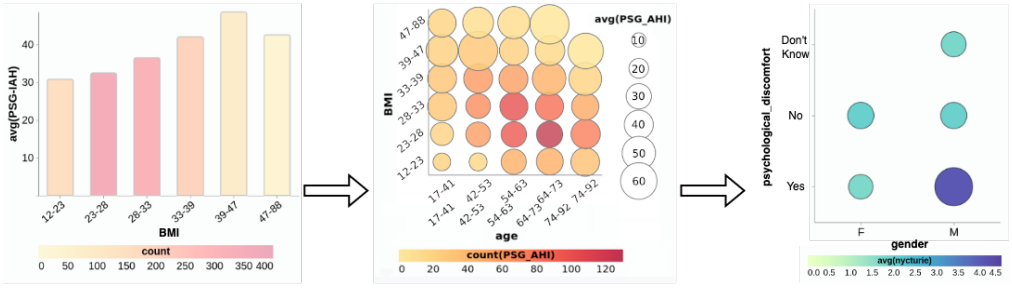


Figure 3.22: Example of the process of exploring panels.

Consider a user who wants to explore a medical dataset and is interacting with a system that displays visual analytics. The user wants to have some knowledge about the effect of smoking and overweight on health. Figure 3.22 illustrates an example of that process. In the beginning, the system recommends a panel that shows the results of "`SELECT` BMI, avg(PSG_AHI), count(PSG_AHI) `FROM` Medical `GROUP BY` BMI". This first panel shows the average AHI index, which determines the severity of sleep apnea, based on the BMI categories. The user rejected the panel. As she needed more details about the grouping, she also gave a reason for this rejection: "Bad Groupby". Based on this feedback, the system refined the

[9] https://github.com/adaptive-learning/matmat-web/blob/master/data/data_description.md
[10] `https://www.qualtrics.com/support/vocalize/widgets/creating-cx-dashboard-pages/`

previous panel and recommended a second one which displays the results of "SELECT BMI, Age, avg(PSG_AHI), count(PSG_AHI) FROM Medical GROUP BY BMI, Age". The system considered the rejection reason and added the attribute "Age" to the query. The user accepted that panel as a relevant one. In the next iteration, the system generates a panel using new attributes that are distinct from the ones previously accepted. The panel that displays the results of "SELECT Gender, avg(nycturie), FROM Medical GROUP BY Gender, psychological_discomfort" was accepted. The iterative process continues until the user is satisfied or until no more panels are available.

Queries Recommendation Challenges

The first challenge that our example identifies is related to the knowledge the users have about the data. In fact, just like in EDA [23, 219, 14, 191], users are assumed to have only partial knowledge of the underlying data. A simple solution that one can incorporate is to advocate a stepwise approach where the best panel is recommended at each step and the user chooses to keep it (positive label) or discard it (negative label). This seemingly simple setting raises new challenges. Given a dataset, we must navigate in a combinatorial space of possible panels that is exponential in the number of attributes of the input data. Therefore, it is necessary to reduce user labeling effort and save time by suggesting to the user the best panel at each step. Only unseen panels are candidates to be shown at each step. We hence propose to learn labels of unseen queries. In fact, this is challenging as we must encapsulate multiple dimensions that define these panels. First, each panel carries an inherent utility e.g., its groupby attribute has high entropy. Second, a panel may be favored over another if it covers a higher number of attributes of the input data that are not already kept. Third, the learning process must also account for user feedback to refine its needs. Finally, one may avoid recommending disparate consecutive panels in order to preserve the user's stream-of-consciousness [222].

Our Contributions

To address the aforementioned challenges, we develop DashBot that guides users in generating dashboards and selects relevant panels by minimizing their time and effort by assuming that users have no prior knowledge of the data. To the best of our knowledge, our work is the first to address feedback-based EDA by learning groupby queries. To develop this solution, we formalize two key problems: identifying an unseen panel that covers at least n attributes (New Panel Generation Problem - NPGP), and accounting for user feedback in choosing the next panel (Panel Refinement Problem - PRP).

During the exploration process, and at each step, a panel is recommended by solving one of these two problems. We address NPGP in the beginning of the process or when a panel is accepted (Yes feedback). To do so, we define several utility functions, coverage, entropy, and variance, to describe the usefulness of the different attributes. These attributes are then ranked to select the ones that build the next best query. On the other hand, we address PRP when users reject panels (No feedback). In this case, we extend users' responses by predefining a set of optional reasons, the user might choose from, to justify the rejection. To solve PRP and refine rejected panels, we leverage Multi-Armed Bandits (MAB) [235]. Each arm represents a potential rejection reason. When a user rejects a panel, DashBot either recommends panels similar to the previous one by exploiting the information obtained from the already seen panels or explores the space and selects completely different and unfamiliar

panels. To find a good trade-off between these two actions and target the user with the best panel, we propose two semantics: `Semantic_1` chooses the next panel based on a distance from a previously rejected one, and `Semantic_2` chooses the next panel based on the likelihood of application of each reason. In addition to solving these two problems, we propose a query inclusion rule to automatically label some unseen panels as negative. We do so to prune panels and reduce the number of candidates in subsequent steps.

Finally, to evaluate our solution, we propose to quantify user effort as a function of the boolean feedback and optional reasons provided by the user. We also measure effectiveness under a limited budget of user effort. We perform an empirical evaluation to show the feasibility and scalability of DashBot for different semantics.

Data Model

We are given a relation R that defines our dataset. R may be an existing relation or the result of a join query. $attr(R)$ represents the set of its attributes. While categorical attributes can be directly used for groupby, numerical ones need to be preprocessed and discretized. To do so, we rely on k-means clustering where each value x of the active domain of the discretized attribute is represented by an interval [min, max] from the clustering. This preprocessing returns an enriched relation of R that contains a discrete version of each numerical attribute. Table 3.9 shows an example where the first four columns represent the original attributes. Preprocessing with $k = 2$ adds two discretized attributes.

Table 3.9: Discretization of the numerical attributes with $k = 2$.

Input Relation R				Discretization	
Name	Age	Gender	BMI	Age*	BMI*
Alice	20	F	25.4	$[10, 25]$	$[18.3, 25.4]$
Bob	25	M	31.8	$[10, 25]$	$[31.8, 31.8]$
Charlie	50	M	18.3	$[50, 63]$	$[18.3, 25.4]$
David	10	M	20	$[10, 25]$	$[18.3, 25.4]$
Eve	63	F	21	$[50, 63]$	$[18.3, 25.4]$

Table 3.10: A dashboard consisting of two panels.

Gender	avg(Age)
F	41.5
M	28.33

BMI*	min(Age)	max(Age)
$[18.3, 25.4]$	20	63
$[18.3, 25.4]$	10	50
$[31.8, 31.8]$	25	25

Panels and Dashboards: A panel is a groupby query of the form:

$$\texttt{SELECT } A'_1, \ldots, A'_{y'}, f_1(B_1), \ldots, f_p(B_p) \texttt{ FROM } R \texttt{ GROUP BY } \quad A_1, \ldots A_y$$

such that $\{A_1, \ldots A_y\} \subset attr(R)$ (with $y \geq 1$) are the groupby attributes. We use strict inclusion because there should be at least one attribute of R that is not a groupby attribute, to have meaningful aggregates. We refer to $A'_1, \ldots, A'_{y'}, f_1(B_1), \ldots, f_p(B_p)$ as the columns of the panel. $\{A'_1, \ldots A_{y'}\} \subseteq \{A_1, \ldots A_y\}$ (with $y' \geq 1$) are the groupby attributes on which query results are projected. We require to have at least an attribute that is part of both of the groupby and then select such that the user can associate the aggregates with some information on the group to which they correspond. Moreover, $\{B_1, \ldots, B_p\} \subseteq attr(R) \setminus \{A_1, \ldots, A_y\}$ are the aggregation attributes. The functions $f_1, \ldots, f_p$ are among the 5 standard SQL aggregate functions (min, max, count, sum, avg). Categorical attributes are only aggregated on the count function. Finally, a dashboard is a set of panels. Table 3.10 represent an example of a dashboard of two panels.

Panel Representation: We model each panel over R with a vector of length $6 \times |attr(R)|$, where for each attribute A we have 6 features (Groupby and the 5 aggregation functions). Table 3.11 shows an example of a vector representation for a relation R with two attributes B, C. The semantics of the vector features are:

- $\text{gb}_A = 0$ if A is not part of the groupby; 1 if A is part of the groupby and not part of the select; 2 otherwise. We stress that, regarding the notation gb_A, if attribute A is numerical, the groupby will be on its discretized version.
- For $f \in \{\text{min, max, count, sum, avg}\}$, $f_A = 1$ if the panel contains the aggregate f on attribute A; 0 otherwise.

Table 3.11: Vector representation of the panel corresponding to the query `SELECT` $B, \min(C), \max(C)$ `FROM` R `GROUP BY` B.

gb_B	min_B	max_B	count_B	sum_B	avg_B	gb_C	min_C	max_C	count_C	sum_C	avg_C
2	0	0	0	0	0	0	1	1	0	0	0

Panel Coverage and Inclusion: During the recommendation process, we aim to avoid narrowing users in restricted spaces where similar panels are displayed. To do so, we rely on a notion of coverage. We define panel coverage with respect to the $attr(R)$. In fact, a panel Q covers n attributes of R if there are n attributes of R present in columns of Q, i.e., $|\{A'_1, \ldots A'_{y'}, B_1, \ldots, B_p\}| = n$. The coverage of a dashboard is then defined as the union of all attributes present in the columns of its panels. In addition to that, we define a distance function between panels. We say that a panel Q' is at distance n from a panel Q if Q' can be obtained by removing or adding n columns of Q. Hence, we say that two panels are close if the distance between them is bounded by some number, reflecting that Q' can be obtained from Q with a small number of changes. We also define inclusion between panels. We say that a panel Q' includes a panel Q if Q' contains all columns of Q and possibly others. We also define a distance function between panels.

Labeled Panels: During the process of creating a dashboard, users label positively or negatively the panels that are recommended to them. These are then used to generate the next best-recommended panels. We define $\mathcal{L}$, a set of labeled panels. Every element $p \in \mathcal{L}$ is a triple $< Q, r, e >$ where Q is a panel over R, r is a Boolean (Yes/No) feedback given by users, and e an optional reason when Q is rejected. We assume a fixed number (seven) possible reasons: Bad groupby attributes, bad aggregation attributes, change agg function min, change agg function max, change agg function count, change agg function sum, change agg function avg. This choice is easily adaptable to a smaller or a larger number of reasons.

As the number of groupby queries that can be specified over a relation R grows exponentially with its number of attributes [11], it is not realistic to ask a user to label every panel. Hence a need of inferring the negative labeling of some panels in order to prune them and reduce user effort. In this case, we leverage the last recommended panel Q_{last}. If it is rejected without a given reason, all panels that include Q_{last} are automatically labeled as rejected. Otherwise, if a reason is given, all panels that satisfy that reason are also considered rejected.

Consequently, $\mathcal{L}$ contains panels that are either explicitly or implicitly labeled. We assume

[11] Let $x = |attr(R)|$. The number of distinct SQL groupby queries is $\sum_{y=1}^{x-1} \left(\binom{x}{y} \times \left(\sum_{z=1}^{y} \binom{y}{z}\right) \times (2^{5\times(x-y)} - 1) \right)$

that the set is always consistent, i.e., if the same panel is recommended several times, the user gives each time the same label. Therefore, we ensure that in the next iteration of the process, only unlabeled panels (not present in $\mathcal{L}$) are recommended.

Groupby Queries Recommendation Problems

Ideally, $\mathcal{L}$ contains all panels the user wants to see on the dashboard and as few panels with explicit No label as possible. So, our goal is to recommend panels that interest the most the user. In order to infer this preference, we assume that the current state of the user is defined by $\mathcal{L}$. Based on the Boolean nature of the user's feedback, we formalize two problems: New Panel Generation Problem (NPGP) when the user accepts the last recommended panelQ_{last} (or when $\mathcal{L}$ is empty), and Panel Refinement Problem (PRP) when the user rejects it.

Problem 2 (The NPGP Problem)**.** *Given a user u, a relation R, a number n, the set of labeled panels $\mathcal{L}$ among which Q_{last} received label Yes, find a panel $Q \notin \mathcal{L}$ to recommend to user u s.t.:*

$$\begin{aligned} &\textit{minimize } |Q_{\text{last}}.columns \cap Q.columns| \\ &\textit{subject to } \ cov(Q, R) = n \end{aligned} \tag{3.3}$$

where $cov(Q, R)$ is the coverage of the panel Q with respect to the relation R. By solving this problem, we find an unlabeled panel Q that covers at least n new attributes of R compared to the last accepted panel Q_{last}. The main intuition is to help the user to discover new panels over new data regions.

Problem 3 (The PRP Problem)**.** *Given a user u, a relation R, a number n, the set of labeled panels $\mathcal{L}$ among which Q_{last} received label No, find a panel $Q \notin \mathcal{L}$ to recommend to user u s.t.:*

$$\textit{minimize } (dist(Q, Q_{\text{last}}, R) - n) \tag{3.4}$$

Where $dist(Q, Q_{\text{last}}, R)$ is the distance between Q and Q_{last} with respect to the relation R. By solving this problem, we find an unlabeled panel Q that is the most similar to Q_{last}. We also assure that the distance is smaller than n.

By solving NPGP and PRP, we address the problem of learning groupby queries for interactive dashboard generation.

Our Proposed Solution: DashBot

Algorithm 3 shows the pseudocode of our solution, DashBot, that learns groupby queries for dashboard generation. The labeled data $\mathcal{L}$ is initially empty (Line 1) and is updated throughout the interactions. At the beginning of the process or after accepting Q_{last}, NPGP is solved. On the other hand, after every rejection, PRP is solved. In both cases, Q_{last} is added to $\mathcal{L}$ (Line 4,7). The process until the halt condition is reached (Line 2). The user may choose to predefine an optional dashboard size, i.e., the maximum number of accepted panels, which represent the default halt condition. Alternatively, the user may stop the process when she is satisfied with the current state of the dashboard. In addition, the process also stops when there is no unlabeled panel to recommend. The overall architecture of DashBot is presented in Figure 3.23. In the following, we present how to solve NPGP and PRP.

Algorithm 3: Interactive learning of groupby queries for dashboard generation

Input: Relation R, user u, number n
Output: Set of panels that u labels Yes

```
1  L ← ∅                                   / Labeled panels, initially empty /
2  while halt condition is not satisfied do
3  |   Q_last ← GenerateNewPanel(R, L, n)
4  |   Ask user u label for Q_last and update L
5  |   while Q_last.r ≠ Yes do
6  |   |   Q_last ← RefinePanel(R, L, n)
7  |   |   Ask user u label for Q_last and update L
8  |   end
9  end
10 return {Q|Q ∈ L, Q.r = Yes}              / Panels with label Yes /
```

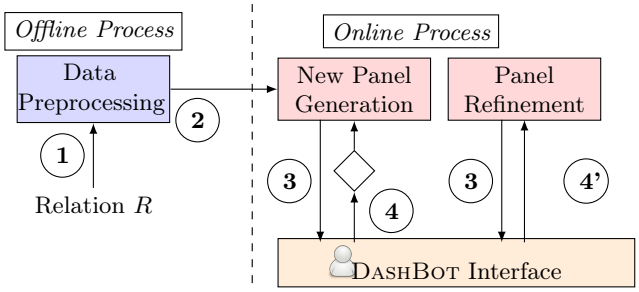


Figure 3.23: Architecture of DashBot. The input relation R (1) is preprocessed into an enriched relation (2). Each edge (3) is a panel recommendation. If the user gives the label Yes (4), the diamond allows the addition of the panel to the dashboard and the verification of the halt condition. If it is false, go to New Panel Generation. Alternatively (4'), the user gives a label No and an optional reason. The last panel is then refined.

New Panel Generation: To solve NPGP, we maintain two ranking lists for groupby and aggregation attributes, respectively. We generate a panel candidate Q that groups by the attributes at the top of the groupby ranking and aggregates on the attributes at the top of the aggregation ranking using all aggregation functions if numeric attributes are selected, and only count function if categorical ones are. In this solution, we also aim to cover at least n attributes not already present on the dashboard. We set the value of n to 2 (One grouping and one aggregating attribute). In the case where the user accepts a panel Q, the covered attributes are moved to the end of the two ranking lists.

Although the initial ranking of lists can be randomly generated, we propose to rely on data-driven utility functions. We use *entropy* and *variance* for groupby and aggregation respectively. In order to rank groupby attributes we define the two following rules:

- Attributes with high entropy are favored. This is because they offer balanced groups with approximatively similar sizes. In this way, the functions min/max/avg are less likely to be redundant and more interesting. We calculate the entropy of an attribute A as follows: $H(A) = -\sum_{x \in adom(A)} P(x) \log_2 P(x)$. $adom(A)$ represents all possible values of attribute A and $P(x)$ is the proportion of tuples, in R, that have the value x for that attribute.

- Attributes that were not preprocessed are favored. This is because discretization yields information loss. Hence we rank first these attributes that have an $adom()$ smaller than k (k-means parameters). We then favor attributes discretized into k clusters. Finally, attributes with $adom()$ larger than k are ranked last.

In the case where two attributes that are not discretized have the same entropy value, the rank is performed randomly. Moreover, if several attributes have an $adom()$ larger than k, they are ranked in increasing order of their $adom()$ size. For example, based on Table 3.9, the groupby attributes rank is Gender > Age* > BMI* > Name. The name is the last as $adom(Name)$ size is too large. BMI* has an entropy (0.72) smaller than Gender and Age* which are equal (0.97). Gender is ranked first as it was not discretized compared to Age*.

In order to rank aggregation attributes, we favor numeric attributes with high variance, for which all aggregation functions might be interesting. We then consider numeric attributes with low variance, for which min/max/avg might be redundant. Finally, categorical attributes for which the count function is the only are ranked last. We compute variance of a numeric attribute A as follows: $V(A) = sum_{t \in R}(t[A] - \frac{\sum_{t \in R} t[A]}{|R|})^2$. $t[A]$ represents the value of attribute A of a tuple $t \in R$. In our example, the aggregation attributes rank is Age > BMI > Gender = Name. Gender and Name are last as they are categorical. Age is ranked first as V(Age) > V(BMI).

Based on these rankings, the first generated panel displays the results of "`SELECT` Gender, min(Age), max(Age), count(Age), sum(Age), avg(Age) `FROM` R `GROUP BY` Gender".

Panel Refinement: We solve PRP after each label No from the user. Our goal is to converge as quickly as possible to a panel that may interest the user. In the case where the user provides one of the optional rejection reasons, DashBot modifies the last assigned panel (Q_{last}) accordingly and recommends the new one. For example, if Q_{last} displays the results of "`SELECT` Gender, min(Age), max(Age), sum(Age) `FROM` R `GROUP BY` Gender" and the user rejects it with reason "Change aggregation function sum", then our solution recommends the panel of "`SELECT` Gender, min(Age), max(Age) `FROM` R `GROUP BY` Gender" by dropping the aggregation function sum. On the other hand, in the case where the user does not provide any reason, we propose an approach based on multi-armed bandits (`MAB`) [235] in order to infer that reason. `MAB` mimics the probable behavior of the user by balancing between exploitation and exploration. During exploitation, `MAB` recommends a panel similar to Q_{last} while during exploration, `MAB` selects a completely different and unfamiliar panel. We propose two different `MAB` semantics that learns how to find the best trade-off between exploration and exploitation:

`Semantic_1` (***ChooseDistance***)**.** This first semantic relies on the distance function between panels to find the trade-off. When exploiting, Q_{last} is slightly modified. The intuition is that this latter is not far from the interesting panel. To do so, we randomly choose $n = 1$ feature from the representation of Q_{last} and invert its value. Therefore, with this minor change, the distance between Q_{last} and the new panel is minimized (equal to n) and their similarity is maximized. When exploring, we aim to investigate panels that are distinct from Q_{last}. To do so, we also randomly choose $n = |attr(R)|/2$ features from the panel vector and invert them. Therefore, the distance between the panels is maximized. In the case where the inversion of features gives an invalid panel Q (Missing groupby or aggregation attributes), we correct it.

We leverage the ε-GREEDY strategy to implement this semantic. In this strategy, we assume a fixed value of ε (0.1). We then exploit most of the time with a probability $1 - \varepsilon$ and explore once in a while with a probability equal to ε. Choosing with a high probability to show a panel close to Q_{last} helps preserve the stream-of-consciousness [222] of the user. Furthermore, seeing multiple similar panels might help the user realize what is not desired, and ideally, provide reasons when rejecting the panels. On the other hand, we explore the space of possible candidates to avoid narrowing the user into a restrictive space and avoid putting the system on the wrong path.

We propose different adaptations to this strategy (Table 3.12). Their names are prefixed with ε-GREEDY-D (D for distance). They all share the same exploitation procedure but differ during exploration. When exploiting, all of them select randomly some features in the vector that represents Q_{last} to change them. When exploring, ε-GREEDY-D-FAR-PANEL changes a large number of randomly-chosen features in the vector of Q_{last}. ε-GREEDY-D-NEW-PANEL shows a completely new panel based on utility, using the new panel generation module. ε-GREEDY-D-ALTERNATING alternates between the previous two strategies.

Semantic_2 (***ChooseReason***)**.** This semantic offers a more sophisticated way of choosing the features to be changed when generating a refined panel. It infers the likelihood of the possible reason for obtaining the label No. These scores are inferred based on all previous reasons given by the user using MAB strategies [19, 210, 235]. Each rejection reason is defined by a bandit's arm. By assuming a set $\{1, \ldots, l\}$ of possible reasons, the performance of each $i \in [\![l]\!]$ to generate an accepted panel is computed as follows: $\hat{\mu}_i = \frac{s_i}{n_i}$ where s_i is the number of times the reason (arm) i has been used in panel refinement such that the user gives a positive label to the refined panel and n_i is the number of times the arm i has been chosen. At the beginning of the process, the MAB strategy [235] chooses each arm once to initialize s_i and n_i. After the initialization, it chooses the most appropriate arm based on the scores of $\hat{\mu}_i$. In the case of exploitation, the most likely arm is chosen to refine Q_{last} while, in the case of exploration, the least likely arm is selected. In both cases, as for Semantic_1, we randomly select $n = 1$ feature from Q_{last} vector according to the chosen arm, and invert it to refine the panel.

We propose different MAB strategies (Table 3.12) that return arms according to a probability matching (SOFTMAX) or an argmax (all the others). To avoid confusion with the ε-GREEDY from Semantic_1, we denote the ε-GREEDY adaptation from Semantic_2 as ε-GREEDY-E. THOMPSON Sampling selects the arm with probability equal to the probability of it being optimal, the upper confidence bound (UCB) which combines $\hat{\mu}_i$ and an uncertainty measure with a confidence degree and SOFTMAX which relies on Boltzmann distribution with temperature (τ).

Experiments

We present an empirical evaluation to analyze the feasibility and scalability of DashBot, for different types of user interactions (Boolean labels with or without given reasons) using the MAB strategies of both presented semantics.

Data: We relied on the MovieLens [111] 100K[12] dataset. We used a natural join of three tables: ratings, users, and movies to form the relation R. This relation has 100k tuples and 29 attributes after removing primary and foreign keys. We also applied a preprocessing on

[12] https://grouplens.org/datasets/movielens/100k/

Table 3.12: Summary of MAB panel refinement strategies implemented in DashBot.

Semantic_1 *ChooseDistance*	Strategy	Exploit	Explore
	ε-GREEDY-D-FAR-PANEL	Change a small number of features in the vector representation of Q_{last}	Change a large number of features in the vector representation of Q_{last}
	ε-GREEDY-D-NEW-PANEL		Show a completely new panel using the New Panel Generation module
	ε-GREEDY-D-ALTERNATING		Alternate between the previous two
Semantic_2 *ChooseReason*	Strategy	Reason chosen at the t^{th} panel refinement	
	ε-GREEDY-E	**return** $\begin{cases} \text{argmax}_{i \in [\![l]\!]} \hat{\mu}_i \text{ , with probability } 1-\varepsilon \quad \textit{(exploit)} \\ \text{a random reason, with probability } \varepsilon \quad \textit{(explore)} \end{cases}$	
	UCB	**return** $\text{argmax}_{i \in [\![l]\!]} \left(\hat{\mu}_i + \sqrt{\frac{2 \ln(t)}{n_i}} \right)$	
	THOMPSON	**for** $i \in [\![l]\!]$ **sample** $\theta_i \sim \text{Beta}(s_i + 1, n_i - s_i + 1)$ **return** $\text{argmax}_{i \in [\![l]\!]} \theta_i$	
	SOFTMAX	**return** arm i with probability $\frac{e^{\hat{\mu}_i / \tau}}{\sum_{j=1}^{K} e^{\hat{\mu}_j / \tau}}$	

numeric attributes using k-means with $k = 7$. We believe that a single dataset suffices to empirically evaluate DashBot as the difficulty of solving NPGP and PRP comes from the size of the schema, i.e., the number of attributes and is thus independent of the actual data. For the purpose of simplicity, we present the attributes with standardized names $(A, B, C, \ldots)$.

Metrics and Variants: In our experiments, we simulate users and their interactions. We define two experimental settings. In the first one only Boolean labels are allowed. The user executes only one click when the panel is displayed. In the second setting, users are obliged to select a reason when rejecting a panel. In this case, the effort of a user is defined by two clicks. As we are simulating users, we also predefine target dashboards. We assume that these dashboards contain all interesting panels for users. For the purpose of generality, we vary the number of panels in each dashboard in $[1, 2, 3, 6, 9, 12]$ and the number of considered attributes in $[4, 6, 8, 10, 12, 15, 20, 25, 29]$. In each case, R is projected on a randomly chosen subset of attributes from $attr(R)$. We assume that a user gives Yes feedback to a recommended panel Q_{last} if this latter is included in one of the panels present in the predefined dashboard.

To evaluate the different MAB strategies, we define three metrics: (1) the number of steps (or clicks) to find all panels in the predefined dashboard (user effort), (2) the average clock time from the beginning of the process until the last Yes label, (3) F1-Score which relies on Precision and Recall (Section 2.1.3). These are computed based on the vector representation of panels where relevant ones form the predefined dashboard.When simulating user feedback, we assume no additional time for reading and understanding a suggested panel. In fact, only the user's wait time is recorded. Thus, the definition of user effort is partial.

We also encounter several variants. In addition to Semantic_1 and Semantic_2 strategies, we consider RANDOM that chooses panels randomly. To compare the different settings of interactions, we consider the variants REASON-x. These are extensions of ε-GREEDY-D-FAR-PANEL where x% of No labels are accompanied by a rejection reason. We vary $x \in \{1, 2, 10, 100\}$. For example, REASON-100 gives always a reason after a No label while REASON-2 does that every 50 step. Reasons are chosen randomly from the set of possible reasons with respect to the targeted panels in the dashboard. REASON-x based on ε-GREEDY-D-NEW-PANEL and ε-GREEDY-D-ALTERNATING exhibit same results.

Panels Implementation: In this section, we give more details about the implementation of the panels. As we discussed in Section 3.2.2, each panel is represented with a vector. More precisely, we assume that if an attribute appears in the groupby, then it should also appear in the select. This results in binary vectors that allow us to slightly reduce the combinatorial space without much expressiveness loss. In order to reduce the vectors' length, we keep only a single feature for the count function instead of $|attr(R)|$ count features. This does not yields expressiveness loss because, under standard SQL multiset semantics [226], count always gives the same result, regardless of the attribute on which it is applied. In addition, for the categorical attributes, we do not keep features for the aggregation functions.

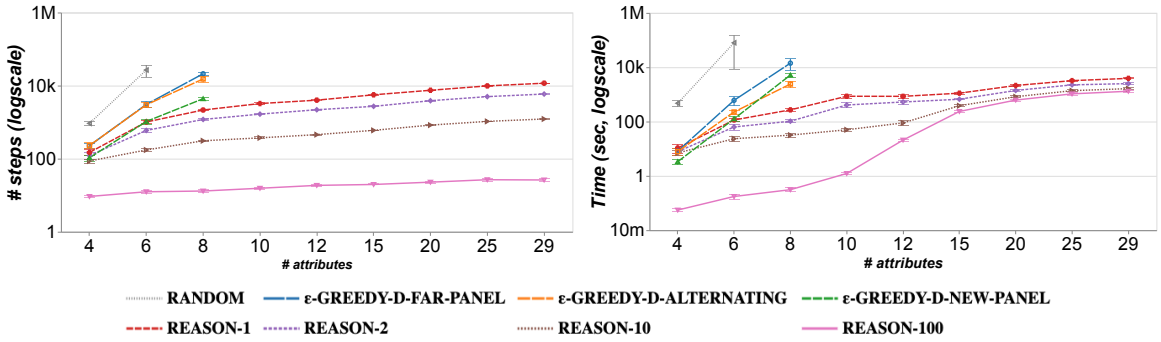


Figure 3.24: Dashboard consisting of a single panel "`SELECT A, avg(B), count(*) FROM R GROUP BY A`".

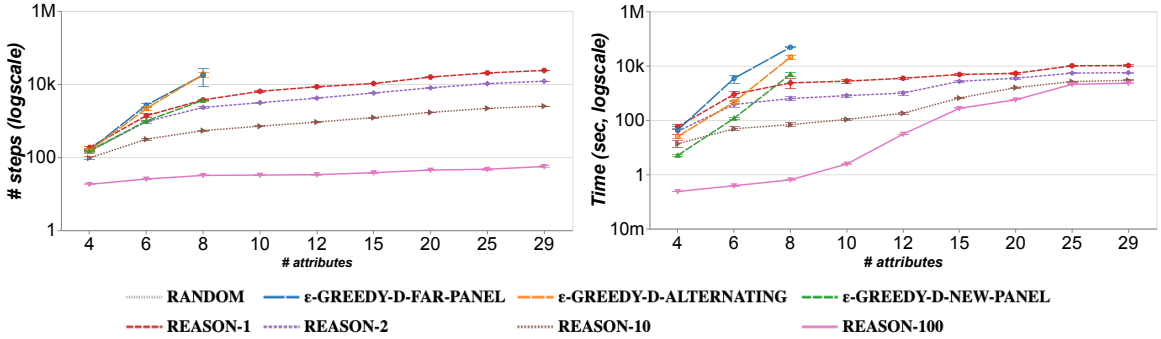


Figure 3.25: Dashboard consisting of two panels "`SELECT A, avg(B), count(*) FROM R GROUP BY A`" and "`SELECT C, sum(D) FROM R GROUP BY C`" .

Experimental Results: Our first experiment studies the scalability of `Semantic_1`, for different types of interactions (Boolean labels with or without explanations). We vary the number of attributes and report the number of steps and the average clock time for a target dashboard containing either one or two panels. Our source code and results are available on a GitHub repository [13].

Figure 3.24 reports the results in the first case. The dashboard contains one panel that displays the results of "`SELECT A, avg(B), count(*) FROM R GROUP BY A`". We observe that a purely `RANDOM` panel refinement strategy is the worst as clock time and the number of steps increase exponentially. We also see that `RANDOM` is already impractical for 6 attributes. One

[13] `https://github.com/MABDashbot/MAB_Dashbot`

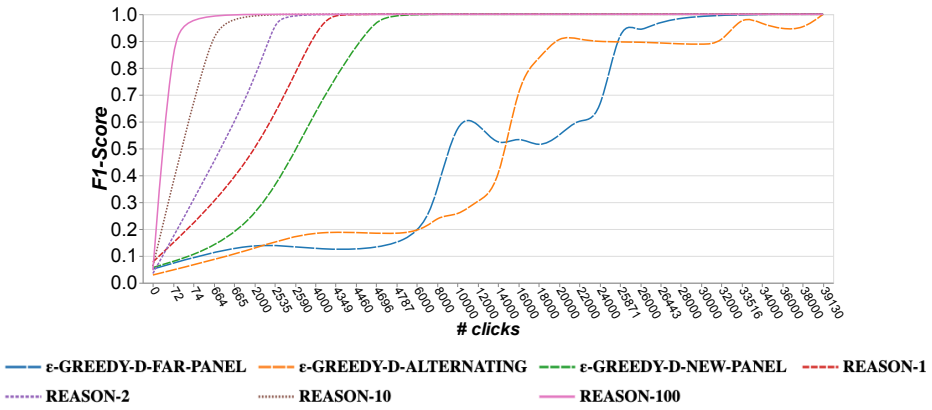


Figure 3.26: F1-Score under a fixed budget of clicks.

can also observe that our `Semantic_1` strategies perform better with a slight advantage for ϵ-`GREEDY-D-NEW-PANEL`. This shows the effectiveness of our data-driven utility functions (entropy and variance) that are used during the exploration phase of ϵ-`GREEDY-D-NEW-PANEL`. Another possible explanation is that, during exploration, ϵ-`GREEDY-D-FAR-PANEL` acts as a random as it relies on a distance function where the n inverted features are randomly chosen. However, one can see that our `MAB` strategies became impractical when the number of attributes is greater than 8. From this figure, we also see the importance of the rejection reason, especially when the number of attributes is too large. Indeed, one can see that the variant `REASON-100` is the best performer where its number of steps remains constant. Also, the performance of `REASON-`x decreases when the frequency of chosen reasons is decreasing. In terms of clock time, we can observe that all `REASON-`x variants spend the same time to refine rejected panels when the number of attributes is high > 12. We can explain that by the fact that many panels were implicitly labeled No with the inclusion rule. As rejection reasons are selected randomly, it becomes hard to find unlabeled panels. The same results can be observed when the dashboard has two panels (Figure 3.25).

In Figure 3.26, we analyze the F1-Score for the case of two panels over 8 attributes. We can see that, as expected, the number of clicks needed to learn the target dashboard (i.e., F1-Score= 1) increases when decreasing the number of allowed reasons from 100% to 1%. This means that users that often choose to select a reason for rejections, increase their likelihood of reaching the target dashboard. Interestingly, DashBot performs in settings where reasons are quite rare (i.e., $x \leq 10\%$ of `REASON-`x) almost as well as when explanations are given at each user interaction. This suggests the usefulness of DashBot's `MAB` strategies for panel refinement in reducing user effort.

Our second experiment studies the utility of `Semantic_2` and aims to verify the assumption that it performs best when a rejection reason benefits multiple panels. We compare the four strategies to `REASON-10` from `Semantic_1`. We chose this latter as it offers the best trade-off between the number of reasons given by the user and the ability to find the target dashboard. In Figure 3.27, we consider dashboards consisting of many panels that are similar in the sense that the same rejection reason is true for all of them (here, none of the panels shows aggregation function sum). From this figure, we see that `Semantic_2` strategies require fewer steps than `REASON-10`. For `THOMPSON`, we stopped at 10 attributes because it requires more

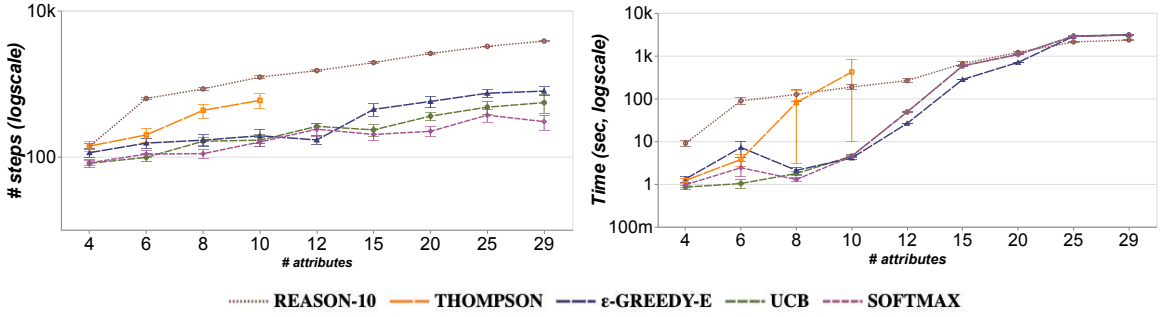


Figure 3.27: Comparison of `Semantic_1` and `Semantic_2` for dashboard consisting of three panels "`SELECT A, min(B),max(B),avg(B),count(*) FROM R GROUP BY A`", "`SELECT A, min(C),max(C),avg(C),count(*) FROM R GROUP BY A`", and "`SELECT A, min(D),max(D),avg(D),count(*) FROM R GROUP BY A`".

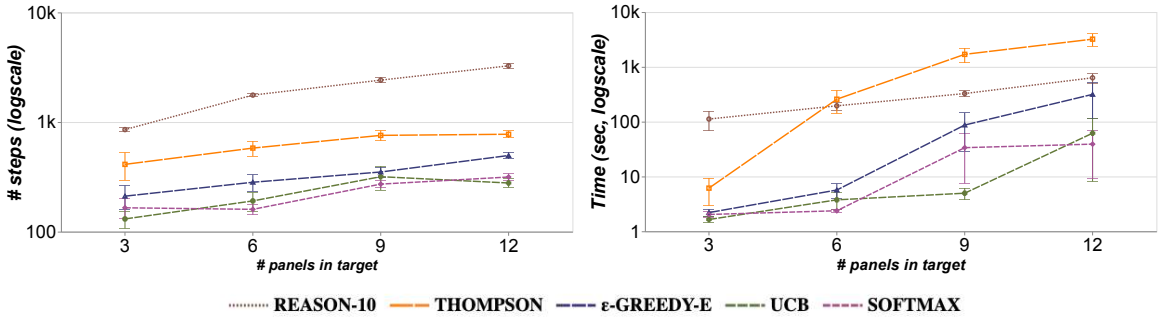


Figure 3.28: Comparison of `Semantic_1` and `Semantic_2` for dashboard consisting of more than three panels. The number of attributes is fixed to 8.

computation time compared to the other strategies that have a smoother behavior. Although all strategies except `THOMPSON` seem to take comparable clock times for a larger number of attributes, the `Semantic_2` strategies are preferred since they require less user effort. A similar behavior is confirmed in Figure 3.28 when varying the number of panels.

Summary. In this application, we tackled the question of recommending visualization panels based on SQL groupby queries to generate dashboards. We developed DashBot that combines data-driven utilities, users' exploration states (accepted and rejected panels), and users-driven feedback to guide them in generating dashboards. Our solution aims to be generic and relies on both `MAB` strategies of different semantics and label inference to reduce users' effort for dashboard generation. It also relies on inferring labels of unseen panels to reduce users' effort. We provided empirical experiments to demonstrate the utility of `MAB` strategies to incorporate user feedback. DashBot has also been demonstrated [68] [14]. More details about the interface and different demonstration scenarios are exposed.

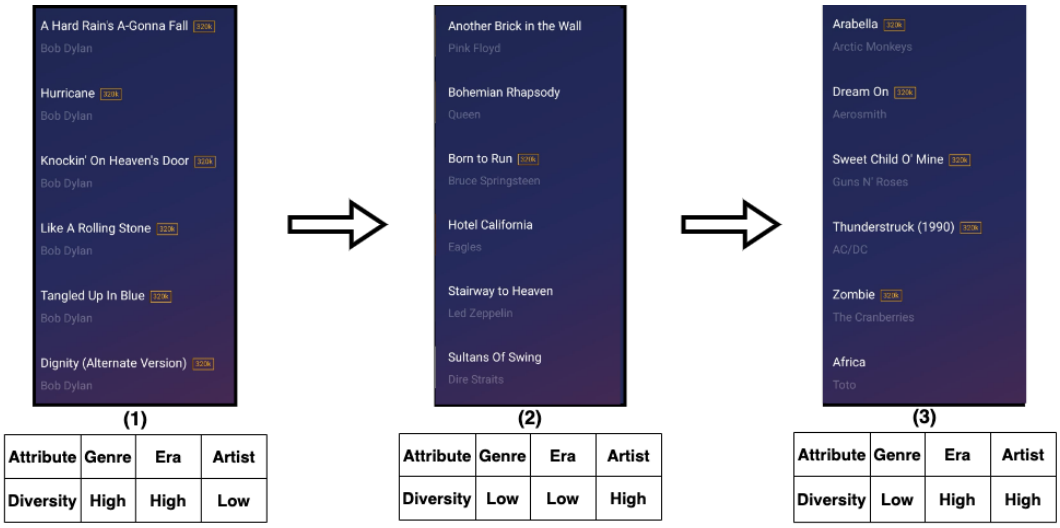
3.2.3 Application 3: Recommendation for Diverse Sessions

Diversity in recommendation has been the topic of a multitude of research efforts [260, 275, 266, 244, 196, 195]. Recent works have shown that diversity improves user satisfaction both

[14] A video is available `https://www.dropbox.com/s/bvwjyqmcvvbiqr0/CIKM_video_srt.mp4?dl=0`

in single sessions and multiple sessions [87]. To enforce the diversity of a set of items, one has to fix the set of attributes used to compute item similarity. This may not capture well the diversity interest and its evolution across time as a single choice of attributes does not fit all users and all sessions. Thus, it is more appropriate to learn diverse attributes in a personalized fashion. The question of learning diversity has received limited attention with only a focus on single sessions [154, 233]. We then consider a setting where the notion of diversity is dynamic and is learned across sessions. We illustrate that with an example.

Motivation: Diverse Session Recommendation



Attribute	Genre	Era	Artist
Diversity	High	High	Low

Attribute	Genre	Era	Artist
Diversity	Low	Low	High

Attribute	Genre	Era	Artist
Diversity	Low	High	High

Figure 3.29: Example of diversity-based multi-session recommendation.

Consider the case of Sydney, a user who listens to music during a trip. Figure 3.29 displays the playlists she receives. Sydney starts with a playlist of Bob Dylan's songs (1) from different eras (60's, 70's, etc) and different genres (Folk, Rock, etc). After some time, she receives a less diversified playlist composed mainly of Rock music from the 70's interpreted by different artists (2). In the end, Sydney listens to a playlist containing mostly Rock songs from a variety of eras (3). The main observation is that attributes that yield the highest diversity differ across sessions since they are data- and user-dependent. The second observation we can make is that in the case where the diversity is defined by one fixed attribute (e.g., Era) across all sessions, this does not always yield the highest diversity. Sydney would benefit from an automated system that is able to capture the diversity of attributes across different sessions and suggests to her a series of playlists that judiciously combines session diversities.

Our Contributions

Diversity in recommendation matters as it increases user retention and decreases churn [17], but it can also evolve and vary. Traditional diversity approaches fail to capture this change in diversity. For this reason, we propose to identify diversity attributes in each session. Given a set of N items that are relevant to a user, our goal is to find the k best items, in terms of relevance and diversity, to return in the first session followed by the k best for the second session, etc. Identifying diversity attributes could be done in two ways: either by

running a traditional diversity approach such as MMR (Maximal Marginal Relevance) [52, 90] and SWAP [266], and finding the attributes that yield the best diversity in each session, or by relying on machine learning techniques to identify those attributes.

The first contribution is to leverage MMR and SWAP to find the best diversity attribute in each session. Our adaptation consists in iterating over all available attributes and items in a session and returning the attributes that maximize the objective function of MMR or SWAP. The set of available items in each session is formed by the items that have not been returned in the previous sessions either because of their low relevance or because they did not contribute to diversity. The approach goes on until the number of items N or the number of desired sessions is completed. Our second contribution is to learn the attributes of diversity in each session. Given the lack of logs, we propose to leverage Reinforcement Learning (RL) to train an agent based on a reward that reflects the best diversity attained in each session. The output of the training is a policy that generalizes single-session diversity and maximizes the underlying objective function across multiple sessions.

In recent works, RL models showed promising results when applied to learning diversity [110],[233]. The application of RL models allows the prediction of long-term goals that are important in a multi-session recommendation problem. Our third contribution is to adapt SMORL [233], a state-of-the-art RL solution for recommendation diversity, to take into account multi-session diversity. The SMORL model is composed of two heads, the self-supervised and RL regularizer. The former head is a fully connected layer that plays the role of a traditional recommender. It ranks all items by predicting their relevance and is trained using a cross-entropy loss. The latter is an RL head which modifies the initial ranking of items as they are trained simultaneously. It learns the Q-function using the Scalarized Deep Q-learning (SDQL) [168]. We extend this model with another fully connected layer that allows us to identify the right attribute that maximizes diversity by learning the Q-function using DQN [166].

Data Model

Let $\mathcal{U}$ denote the set of users and $\mathcal{I}$ the set of items that any user $u \in \mathcal{U}$ can choose from. An item $i \in \mathcal{I}$ is represented with a vector of attributes $< att_1, att_2, .., att_p >$ drawn from a set of attributes $\mathcal{A}$. For instance, in the music domain, items can be represented as $<$ artist, genre, release date$>$, and a song may be represented as $<$ *Pink Floyd*, *Rock*, *1979* $>$. We assume that we are given a distance $d : \mathcal{I} \times \mathcal{I} \times \mathcal{A} \rightarrow \mathbb{R}^+$ that reflects dissimilarity, i.e., the diversity of two items in $\mathcal{I}$ with respect to an attribute att. We use $d_{att}(i, j)$ to reflect the diversity of the two items i, j on attribute att.

Diverse Session Recommendation Problem

Lately, recommendation systems relied on embeddings to encode latent representations of items [17, 110]. Once embeddings are computed, items that are strongly related to each other will have close representations in the item embedding space. To produce such an embedding space we follow a methodology that is used for Spotify [17]. It consists of training a Word2Vec algorithm [163] on user-item interactions. Traditionally, Word2Vec is used in natural language processing to embed words from a corpus of documents. In our case, we used it to produce item embeddings for each attribute. To train the Word2Vec models, we map each item to a

numerical index and use as context the items sharing similar attribute values. The vocabulary size is represented by the number of items. For numerical attributes, we compute the similarity between items using Cosine similarity. For categorical attributes, two items are similar if they share the same value. In the case where the number of similar items exceeds the context size, we choose items with similar attributes randomly.

To compute the diversity of a set of items in a given session S with respect to an attribute $att \in \mathcal{A}$, we leverage the widely used intra-list-distance measure [275] that computes the diversity of items in a session as the average pairwise distance between items in the session. More formally

$$div_{att}(S) = \frac{\sum_{i \in S} \sum_{j \in S \setminus \{i\}} d_{att}(i,j)}{|S|(|S|-1)} \tag{3.5}$$

We can now state our goal: Given user u and a set of l sessions, recommend the k most diverse unseen items in each session. We aim to maximize the diversity of each session by finding the attribute that yields the highest diversity. More formally, in each session S, we look for an attribute att, s.t.:

$$argmax_{att}\ div_{att}(S),\ |S| = k$$

Our Proposed Solutions

To solve our problem, we devise two main approaches to look for the best diversity attribute in each session: the first is a generalization of traditional diversity approaches and the second is based on reinforcement learning to learn diversity attributes in several sessions.

A variety of diversity-based approaches have been developed for recommendation. In this application, we leverage `SWAP` [266] and `MMR` [52], two common diversity approaches, to develop our baselines. We adapt `SMORL`, a state-of-the-art RL architecture [233] for multi-attribute recommendations.

`MMR` Adaptation. A widely used approach to combine relevance and diversity is the greedy algorithm `MMR` [52, 90]. Given a set S of the k most relevant items to recommend to a user, `MMR` selects at each step a new candidate item i^* that maximizes a linear combination of its relevance and the gain in diversity that is achieved according to already selected items. More formally, it chooses i^* s.t.

$$i^* = \underset{i \in \mathcal{I}}{\text{argmax}} \left((1-\alpha) \cdot rel(i) + \alpha \min_{j \in S} d(i,j) \right)$$

where $\alpha \in [0,1]$ is a parameter that tunes the relative importance of each relevance and diversity. Higher values of α mean more importance is put on diversifying the resulting recommendation set.

Algorithm 4 illustrates our adaptation of `MMR` which consists in iterating over all attributes in $\mathcal{A}$ to find the one yielding the highest diversity.

Algorithm 4: Multi-attribute `MMR`

Input: user u, set of items $\mathcal{I}$, set of attributes $\mathcal{A}$, α, # recommendations k, # sessions l
Output: l-session recommendations

$\mathcal{X} \leftarrow \emptyset$
for $j \leftarrow 1$ **to** l **do**
  $C \leftarrow$ [*Items in relevance order to* u]
  $divs\_A \leftarrow []$ (To keep the result set of each attribute)
  foreach *att in* $\mathcal{A}$ **do**
    $S_j \leftarrow C[0]$
    while $|S_j| < k$ **do**
      $i^* = \text{argmax}_{i \in C \setminus S_j} \quad ((1-\alpha) \cdot rel(i) + \alpha \min_{j \in S} div_{att}(i,j))$
      $S_j \leftarrow S_j \cup \{i^*\}$
    $divs\_A.append(S_j)$
  $S^* \leftarrow$ Get the set with the highest diversity in $divs\_A$
  $C \leftarrow C \setminus S^*$
  $\mathcal{X}.append(S^*)$
return $\mathcal{X}$

`SWAP` **Adaptation.** We propose algorithm 5 based on `SWAP`, a re-ranking approach that is divided into two steps. First, a recommender is used to predict the relevance of unseen items to a user u (Line 3). A top-k list of items S is selected (Line 4). Secondly, for each attribute in $\mathcal{A}$ (Line 6), a recommended list (S_{att}) is computed by a standard `SWAP` approach using the same initial S where items that contribute the least to diversity are replaced with ones that maximize it (Lines 8-16). The list S_{att} with the highest diversity is selected as the session to recommend (Line 19).

Learning Multi-Attribute Diversity. We formalize our problem as a Markov Decision Process (MDP) in which the agent interacts with the environment represented by all users. We define the key components of the MDP represented by the tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{P}$, $\mathcal{R}$, γ) as follows:

- State Space ($\mathcal{S}$): It describes the user state at time t. A state s_t is represented by an embedding vector that summarizes the last session, defined by the last k items the user interacted with. More formally, $s_t = F_{emb}(c_{t-1})$ where c_{t-1} is the $(t-1)^{th}$ session. F_{emb} should capture the connections between the different items within the session as well as their order which makes it different from having as input the union of all items of the session.
- Action Space ($\mathcal{A}$): An action permits the transition between two consecutive states. In this work, we consider two types of actions: Choosing the items that define the next session recommended to the user and selecting the attribute for which the diversity of the next session is maximized. More formally, $a_t = (c_t, att_t)$ where c_t is the t^{th} session and att_t is the selected attribute.
- State transition probability ($\mathcal{P}$): $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the probability $p(s_{t+1}|s_t, a_t)$ of transition from s_t to s_{t+1} when the agent selects the action a_t.
- Reward ($\mathcal{R}$): $\mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the instant reward of taking an action a_t at state s_t. More formally, $r(s_t, a_t) = div_{att_t}(c_t)$ where div_{att_t} is the intra-diversity of the selected session

Algorithm 5: Multi-attribute `SWAP`

Input: a user u, a set of items $\mathcal{I}$, a set of attributes $\mathcal{A}$, # recommendations k, # sessions l
Output: l-session recommendation

$\mathcal{X} \leftarrow \emptyset$
for $j \leftarrow 1$ **to** l **do**
  $C \leftarrow$ [*Items in relevance order to* u]
  $S \leftarrow Topk(C)$
  $divs\_A \leftarrow []$ (To keep the result set of each attribute)
  foreach *att in* $\mathcal{A}$ **do**
    $S_{att} \leftarrow S$; $m = 1$
    while $m < k$ **do**
      $pos = k + m$; $m = m + 1$
      for i *in* S_{att} **do**
        if $div_{att}(S_{att}) < div_{att}((S_{att} - \{i\}) \cup C[pos])$ **then**
          $S_{att}.remove(i)$
          $S_{att}.add(C[pos])$
    $divs\_A.append(S_{att})$
  $S^* \leftarrow$ Get the set with the highest diversity in $divs\_A$
  $C \leftarrow C \setminus S^*$
  $\mathcal{X}.append(S^*)$
return $\mathcal{X}$

c_t using the selected attribute att_t.

- $\gamma \in [0, 1]$ which represents the discount factor for future rewards. If $\gamma = 0$ the agent ignores all future rewards and considers only the immediate one. If $\gamma = 1$ the agent ignores the immediate reward and considers all future ones. We set $\gamma = 0.99$.

The goal of this formalization is to train an agent that is able to find a policy π^* that *maximizes the expected cumulative reward*:

$$\pi^* = argmax_\pi \mathbb{E}[\sum_{t=0}^{|\pi|} \gamma^t r(s_t, a_t)] \tag{3.6}$$

where $|\pi|$ is the length of the policy which corresponds to the number of sessions for each user.

To implement our MDP, we propose to adapt `SMORL`, a state-of-the-art architecture for recommendation diversity [233]. An overview of the framework is displayed in Figure 3.30. It is split into three parts: **(A)** represents the summarising of the k previous items defining the last session. We used the same architecture as in [110] by replacing the LSTMs with GRUs. The session is embedded using this layer of GRUs followed by a layer of attention:

$$o_i = GRU(item_i | o_{i-1}),\ o'_i = ReLU(W_1 o_i + b_1)$$

$$S = \sum_{i=0}^{k} o'_i \frac{e^{W_2 o'_i + b_2}}{\sum_{j=0}^{k} e^{W_2 o'_j + b_2}}$$

(B) represents the adaptation of `SMORL` model [233]. `SMORL` is composed of two heads, the self-supervised and the RL regularizer. The former head is a fully connected layer that plays the role of a traditional recommender. It ranks all items by predicting their utility and is trained using a cross-entropy loss. The latter is an RL head which modifies the initial ranking of items as they are trained simultaneously. It learns the Q-function using the Scalarized Deep Q-learning (SDQL) [168] which is an extension of DQN [166]. The part **(C)** has the goal of learning the right attribute that maximizes diversity. It learns the Q-function using DQN.

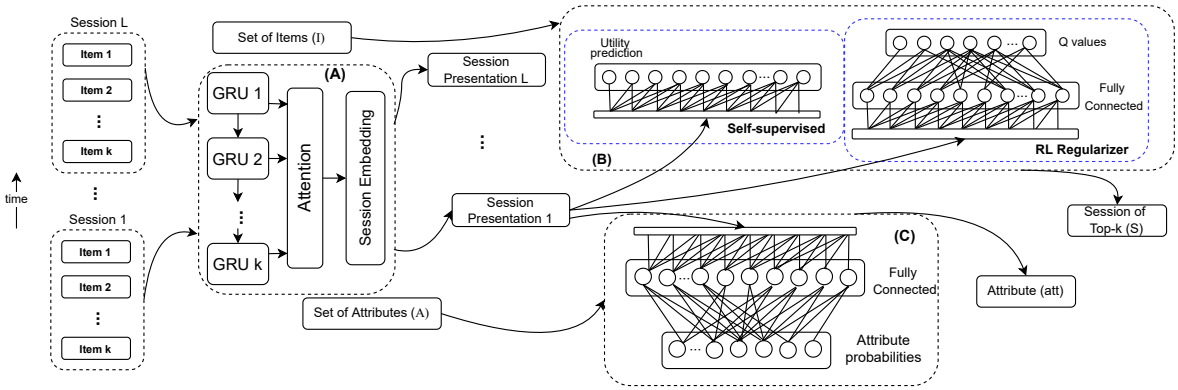


Figure 3.30: Overview of the architecture of the RL framework. **(A)** represents the summarizing of a session into a latent space, **(B)** represents the `SMORL` model [233] to choose the next session items, and **(C)** designates the model for choosing the best attribute to optimize diversity.

Experiments

We run an extensive set of experiments to study the impact of our solutions on diversity as well as relevance and response time. We first evaluate our solutions in a setting of a single session recommendation. We then evaluate them in the context of multiple sessions. Our code as well as our dataset are available on our GitHub repository[15].

Setup: We split the data in a user-wise fashion, i.e., for every user, we chronologically build sessions where each session contains k items. We use the l last sessions, of each user, as a test set and the remaining ones as a training set.

Data: We use a real-world dataset, a merge between `MovieLens` 10M and `IMDb` datasets, which contains 5 real attributes: Genre, Duration, Release Year, Rating, and Type of movie. We generate a semi-synthetic dataset by augmenting the real-world one with $\{10, 20, 30, 95\}$ simulated attributes. The generation of these independent attributes is performed using different distributions: Gaussian, Exponential, Gamma, Uniform, and Zipfian. The choice of the distribution as well as its parameters, to generate a single attribute, is random. The

[15] https://github.com/SessionDiversity/Multi-session-Diversity-Attributes

Table 3.13: Parameter tuning values

Methods	Parameters	Values
SMORL	Batch Size	256, **64**
	Session Size	**100**, 200, 500
	Learning Rate	0.0001, 0.0003, 0.001, **0.003**
MMR	Trade-off of relevance and diversity (α)	0.3, 0.5, **0.6**, 0.9

generation is performed once at the beginning of the process. This dataset is sparse as 98.5% of data is missing.

Variants and Metrics: To compare our algorithms with ones that do not capture attribute diversity, we implement MMR_G and SWAP_G that estimate the diversity in a global fashion.

For each user, we measure precision and diversity. We consider that an item is relevant if it appears in the sessions of the test set. For diversity, we rely on ILD measure [275]. The reader may refer to Section 2.1.3 for details. We also use two different multi-aspect metrics to measure the combination between accuracy and diversity. The first one is an adaptation of the F1-Score:

$$FScore_u@k = \frac{2.Precision_u@k.Diversity_u}{Precision_u@k+Diversity_u}$$

The second one is an adaptation of $nDCG$ [57] called $\alpha nDCG$. It has a tuning parameter $\alpha \in [0, 1]$, which indicates the strength of penalization on the appearance of similar items in the recommended session. In the case where $\alpha = 0$, $\alpha nDCG$ is equivalent to $nDCG$. Given the session $R_u@k$:

$$\alpha nDCG = \frac{\sum ng(r)/log(r+1)}{\sum ng^*(r)/log(r+1)}$$

where $ng(r) = I_u(r)(1-\alpha)^{C_u(r-1)}$ which represents the novelty-biased gain at rank r. $I_u(r)$ is the relevance of the item at the rank r and $C(r) = \sum_{i=1}^{r} I_u(i)$.

Finally, we calculate for each user the response time needed to generate the recommended sessions.

Default values and Parameter tuning. We report results in the case where session size $k = 5$. They are aggregations of 3 runs over sets of sampled users. We performed a grid search over a set of parameters to fine-tune MMR and RL methods and find those yielding the best results. We provide details of all parameters in Table 3.13 and highlight the best ones. The parameter "Session Size" represents the size of the session embedding.

Single Session Recommendation:

Diversity. Figure 3.31 reports the evolution of intra-diversity as a function of the number of attributes. The first observation is that the diversities of the baselines are better than SMORL's with a clear overall advantage for MMR. One possible explanation of the performances of MMR and SWAP is that they calculate the diversity for all attributes and choose always the best one. SMORL on the other hand relies on predicting diversity attributes which makes it more vulnerable and returns more false positives and that affects negatively its performance. For

example, for $\#Attributes = 5$, SMORL has a precision of 25% for choosing the best attributes while SWAP and MMR have a precision of 100%.

The second observation is that the diversity of MMR_G and SWAP_G is smaller than others regardless of the number of attributes. This is because the latter methods tend to select the attributes that maximize diversity while no such optimization is performed for both MMR_G and SWAP_G.

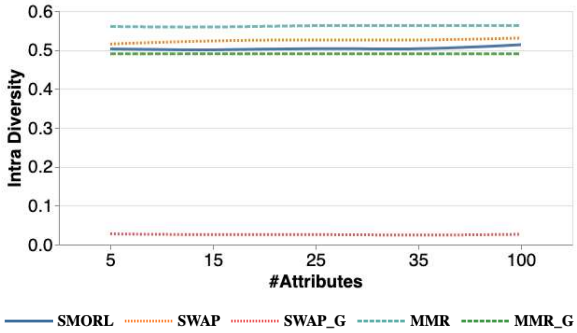


Figure 3.31: Evolution of Diversity as a function of the number of attributes.

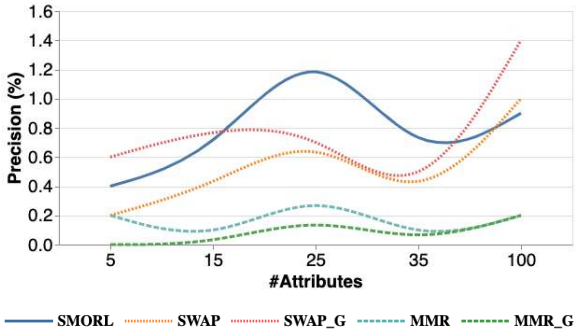


Figure 3.32: Evolution of Precision as a function of the number of attributes.

Accuracy. Figure 3.32 represents the evolution of precision as a function of the number of attributes. From the figure, we can see that, generally, the adaptation of SMORL is the best and outperforms the baselines (MMR and SWAP). As explained in the original paper [233], SMORL can identify users having diverse interests and recommend them suitable items. One possible explanation is that SMORL incorporates a traditional recommender. Indeed, the self-supervised head of SMORL plays that recommendation role and learns the most accurate next items to recommend in a sequential way.

We notice that SWAP performs better than MMR. The reason is that SWAP chooses at each step the best item to swap within the initial list of items while MMR chooses the next item using a linear trade-off function between utility and diversity. We also notice that the two variants of algorithms achieve a similar precision with an advantage of MMR over MMR_G and SWAP_G over SWAP.

Accuracy-Diversity. Figure 3.33 shows the evolution of $\alpha nDCG$ as a function of the number of attributes. The evolution of F-Score is similar to $\alpha nDCG$, so we do not report it. One can see that SMORL is the best performer compared to SWAP and MMR. We can explain that by the fact that the RL head of SMORL is used to introduce more diverse items while the other head provides more accurate ones. The combination between these heads and their mutual learning permits the model to obtain a good balance between diversity and accuracy. Despite MMR having a better and increasing diversity, it is outperformed by SWAP regardless of the number of attributes. Indeed, the reason is, that this latter achieves a far better accuracy which results in having a better balance.

Time. Figure 3.34 reports the evolution of response time as a function of the number of attributes. We see that the baselines have the worst time performance independently of the number of attributes, with better results for SWAP. Indeed, these methods iterate over all attributes to choose the best one. This obviously makes the computation expensive and

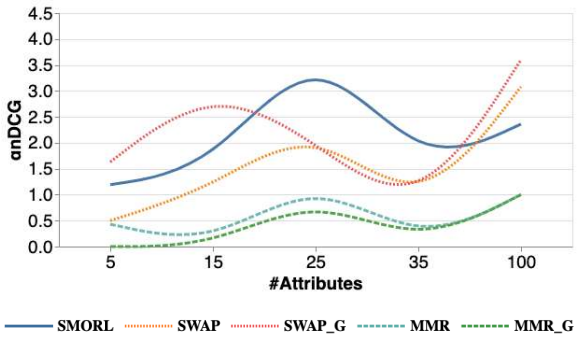


Figure 3.33: Evolution of $\alpha nDCG$ as a function of the number of attributes.

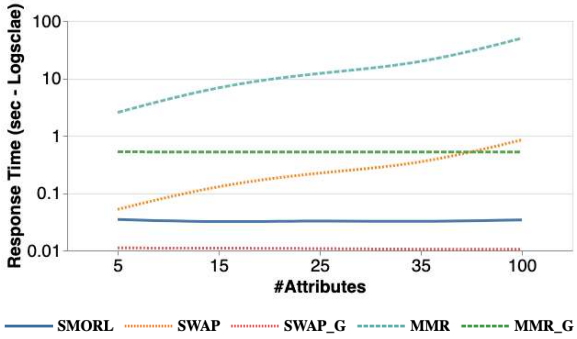


Figure 3.34: Evolution of response time as a function of the number of attributes.

increases with the increase of the number of attributes and items. The second observation is that the RL algorithm has a constant time evolution across the number of attributes. `SWAP_G` is obviously the best performer as it does not iterate over attributes while `MMR_G` outperforms `SWAP` for $\#Attributes = 100$ and `MMR`.

Multiple Session Recommendation: In this section, we fix the number of sessions to $l = 3$.

Diversity. Figure 3.35 shows the evolution of diversity across multiple sessions. We observe that the diversities of all models are mainly constant regardless of the session. The models are designed to optimize the diversity of a session and maintain its maximization for the next ones.

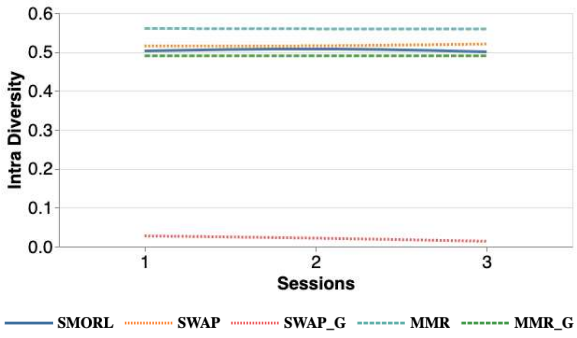


Figure 3.35: Evolution of Diversity across sessions for $\#Attributes = 5$.

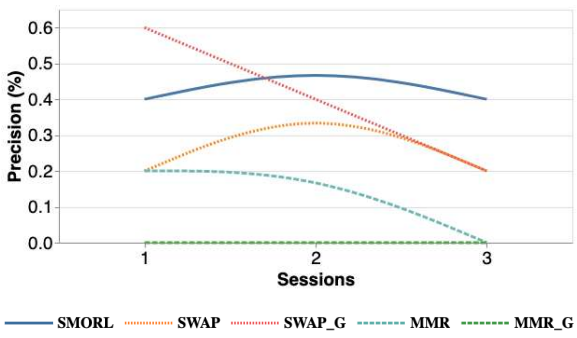


Figure 3.36: Evolution of Precision across sessions for $\#Attributes = 5$.

Accuracy Figure 3.36 shows the evolution of precision across multiple sessions. The main observation is that the precision of `SMORL` and `SWAP` are both increasing and then decreasing with an advantage for `SMORL` while `MMR` precision is continuously decreasing. The other observation is that `MMR_G` is the worst performer and `SWAP_G` is quickly decreasing and outperformed by `SMORL`.

Accuracy-Diversity. Figure 3.37 shows the evolution of F-Score across multiple sessions. We observe that despite the good results of `MMR` on diversity, the trade-off metric is quickly

decreasing. SMORL and SWAP have the opposite behavior as they remain constant. We also observe that methods with attribute selection outperform SWAP_G and MMR_G.

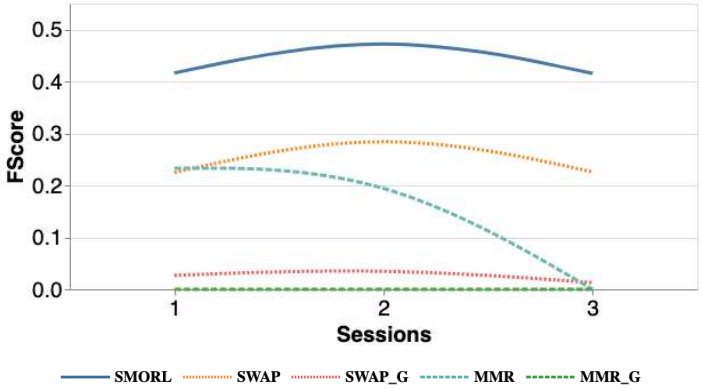


Figure 3.37: Evolution of F-Score across sessions for $\#Attributes = 5$

Learning Transfer: Our last experiment is about policy transfer. We split the users into three classes: users with high, medium, and low diversity using K-means [153]. The diversity of each user is the average diversities of the last 5 sessions in the training set. We split, user-wise, each class in a way that 75% of users, "known users", are used to train a model while the remaining 25% "unknown" ones are used to test the transfer of the model. We train for each class a SMORL model and test it on both types of users. The results are displayed in Figures 3.38, 3.40, and 3.39.

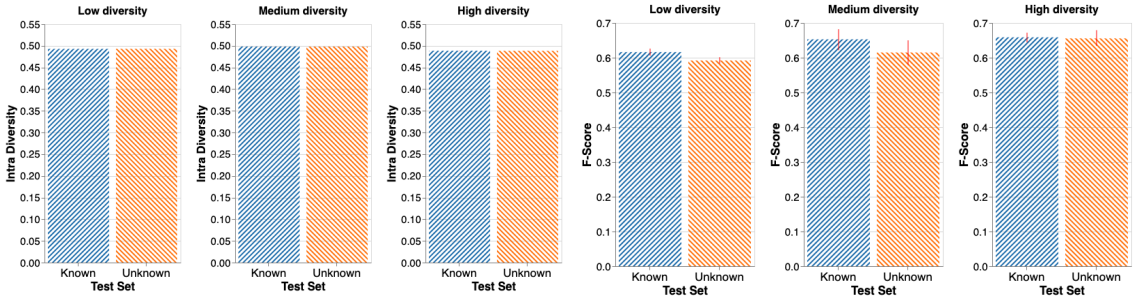


Figure 3.38: Diversity of transfer learning for a single session.

Figure 3.39: F-Score of transfer learning for a single session.

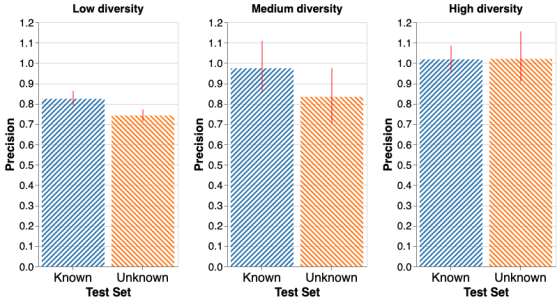


Figure 3.40: Precision of transfer learning for a single session.

From the figures, one can see that, overall, we can transfer SMORL models to users that were unknown to them. Indeed, in the “Low diversity" and “Medium diversity" cases, models do not maintain the level of precision and F-Score but the observed decrease is slight compared to “known" users. For example, we register a loss of 9% and 14% of precision for the “Low" and “Medium diversity" models respectively. In the case of “High diversity", precision and F-Score for “unknown" users are the same as for “known" ones. One can also observe that the results of diversity are the same between the test sets regardless of the type of diversity class.

Summary. In this application, we developed an approach for learning diversity attributes in multi-session recommendations. Our aim is to learn which attribute optimizes the best diversity of each session and target users with suitable diverse items. We also assumed that previous sessions define the states of users. We implemented two different solutions: one based on traditional diversity-based recommenders (MMR and SWAP) and the other based on a state-of-the-art Reinforcement Learning architecture (SMORL). We conducted experiments on semi-synthetic data based on MovieLens and IMDb and demonstrated that SMORL-based approach scales very well and gives the best trade-off between diversity and accuracy. The work in this application was published in IEEE BigData 2022 [43].

3.3 Conclusion

In this chapter, we introduced the notion of dynamic recommendations. We first explored an extension of standard recommenders with users' profiles and states to solve the "Best Selection Problem" in Section 3.1. We proposed a meta-learning methodology that chooses the best recommender for each pair user-item having as input users' profiles (states) and item features. We showed the importance of considering user states in a static environment.

After that, we examined three real-world applications of dynamic recommendations in Section 3.2. These applications are related by the fact that they are interactive where users have a constant change in their behavior. The second link is related to the type of used solutions. In fact, we relied on the same family of algorithms: Markov models. Finally, in all the applications, we leverage multiple dimensions to capture users behaviors and characterize their profiles.

We formalized the AdUp problem based on three learning dimensions to capture the behavior of users to solve the upskilling problem and test recommendations in Section 3.2.1. We proposed two solutions based on either Pareto [24] and multi-objectives (MOO) or Multi-armed bandit [236] (MAB). Our experiments showed that the dynamic solution, MAB, outperforms MOO as it offers more skill gain in fewer iterations.

We also formalized two generic problems: NPGP and PRP to define visualization-based analytics recommendations in Section 3.2.2. We proposed a solution, DashBot based on Multi-armed bandit [236], that targets users with visualization panels that display interesting results of SQL groupby queries. Our solution encounters users' feedback and relies on different data-driven utility functions: Coverage, variance, and entropy. Our experiments show the effectiveness of our solution to incorporate users' feedback and recommended panels that interest them.

Finally, we formalized an attribute-based problem to recommend diverse sessions in Sec-

tion 3.2.3. We proposed two solutions to balance relevance and diversity in session recommendations. The first solution is a generalization of standard diversity-based recommenders algorithms, such as `MMR` [90] and `SWAP`[266] while the second solution extends a known diversity-based Reinforcement Learning architecture, `SMORL` [233]. Our experiments show that the RL solution scales well and offers a better trade-off between relevance and diversity. They also show that our attribute-based solutions outperform standard diversity-based recommenders.

Chapter 4

Collective User Behavior

In this chapter, we aim to extract user collective behavioral patterns based on multiple hypotheses testing. In fact, understanding people and their collective preferences in the Internet of Behaviors [1] requires expressive and statistically-sound methods for data-driven discoveries. A statistical hypothesis test compares two models (the null and the alternative hypotheses) and deems the comparison statistically significant if, according to a significance threshold α, the data is very unlikely to have occurred under the null hypothesis, i.e., the null hypothesis is rejected, and the alternative hypothesis is satisfied. Making sound discoveries for multiple users and in large datasets poses two challenges: the likelihood of accepting a hypothesis by chance, i.e., returning false discoveries, and the pitfall of not being representative of the input, i.e., data coverage.

In this chapter, we propose a solution, GroupTest, that analyzes the collective behavior of users and extracts patterns combining multiple hypothesis testing and coverage. Relying on multiple hypothesis testing minimizes the likelihood of returning false discoveries when ensuring that the results are significant. Optimizing coverage allows us to explore the entire data. Parts of this chapter were published in the WWW 2022 [45] conference. It was also extended to a journal paper published in the Transactions on Large-Scale Data- and Knowledge-Centered Systems [44].

4.1 Motivation: Hypothesis Testing for User Groups

Consider an analyst who seeks to examine movie ratings with the goal of forming an international panel of diverse experts to judge Comedies. In this case, data samples refer to reviewer groups. Figure 4.1 illustrates the example in 3 steps. **Request 1** looks for reviewer groups who provided diverse ratings for Comedies. A one-sample Kolmogorov-Smirnov test identifies three groups that reject the null hypothesis "Rating distribution is unimodal" (and satisfy the alternative hypothesis stated in the request). An additional constraint is needed to ensure that returned groups are representative of the input, i.e. cover most reviewers of Comedies. **Request 2** further refines returned groups by exploring their age subgroups. It applies a two-sample F-test that seeks to reject the null hypothesis "Age groups whose variance for

[1] https://www.gartner.com/smarterwithgartner/gartner-top-strategic-technology-trends-for-2021/

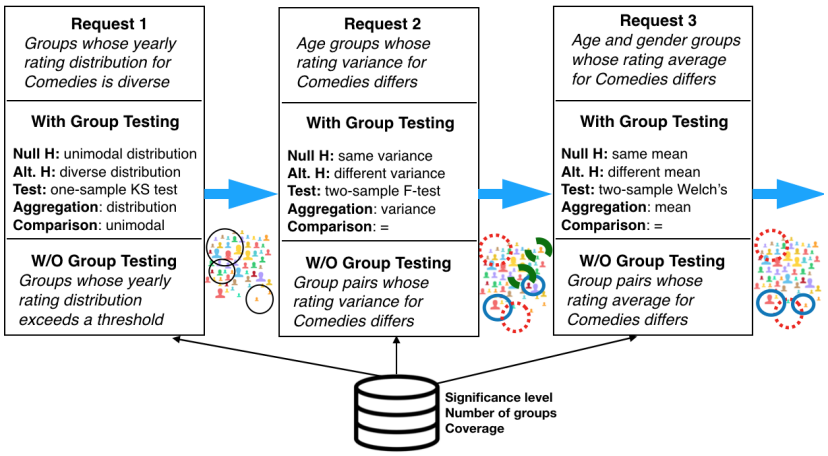


Figure 4.1: A multi-step group testing.

Comedies is the same". Here again, traditional hypothesis testing will only check rating variances. An additional constraint is needed to ensure that the input age groups are covered by the resulting groups. The output groups are fed to a two-sample Welsh test to compare their average ratings in **Request 3**, and return those that significantly differ. Members of those groups can be used by the analyst to form the desired panel.

4.2 Multiple Hypothesis Testing Challenges

Realizing our example requires addressing two challenges: (i) the likelihood of rejecting a null hypothesis and returning groups by chance, when the number of groups grows, and (ii) the pitfall of returning groups that are not representative of the input data of interest. Indeed, when multiple data samples are tested against a hypothesis, the chance of observing a rare event increases, and hence, the likelihood of incorrectly rejecting a null hypothesis (i.e., making a Type I error [208]) increases. There are precise criteria for excluding or not a null hypothesis at a certain significance level [123, 160]. Those criteria depend on the type of test (i.e., one-sample, two-sample, multiple-sample), the aggregation function (e.g., mean, variance), the sample sizes, whether the samples are paired (same subjects), and the comparison operators (e.g., equal, greater). The second challenge is particularly important for large datasets where the number of groups that pass the test increases and the choice of which ones to return may affect how representative they are, i.e. how much they cover the input. This requires revisiting the conditions under which a null hypothesis is rejected to additionally account for data coverage when selecting groups to return.

4.3 Our Contributions

We develop GroupTest, a unified framework that supports a variety of statistical tests to verify if user behavior supports the null or the alternative hypotheses and return qualifying groups. We, first, formulate a generic problem ValMin as an extension of the traditional multiple hypothesis testing problem. ValMin is a top-n problem that seeks n user groups accommodating different hypothesis tests (one-sample, two-sample, or multiple-sample tests)

and additionally satisfying a constraint on data coverage. This problem enforces coverage but does not maximize it. We hence propose to generalize it by formulating CovMax which is also a generic top-n problem that maximizes data coverage while setting a constraint on hypothesis significance. ValMin and CovMax leverage different hypothesis correction methods, FWER [30] and FDR [34].

We show that both problems are NP-hard with a reduction to the Partial Weighted Set Cover Problem and the Maximum Coverage Problem respectively. We develop `VAL_C`, a greedy algorithm that solves ValMin, and `COVER_G`, a greedy algorithm that solves CovMax. To address scalability, we also develop `COVER_`α, a more efficient heuristic algorithm to solve CovMax.

`VAL_C` iterates over the set of all candidate groups and chooses the ones that maximize significance (smallest p-values). Similarly to traditional procedures, `VAL_C` controls the multiple testing error at a given significance level α (usually set to 0.05). `COVER_G` is a greedy algorithm, with a provable approximation guarantee. As `VAL_C`, it iterates over the set of candidate groups and instead of choosing the candidates that minimize significance, it chooses the next candidate that maximizes coverage. It also controls significance by calculating and ranking the p-values of all candidates. This hinders its scalability when the number of groups increases. To address that, we propose `COVER_`α, a heuristic algorithm that builds on α-investing [89], an adaptive sequential method that controls mFDR, the ratio of the expected number of false rejections to the expected number of rejections. Different investing policies of the α-wealth have been proposed previously albeit without considering data coverage [273]. The key idea of `COVER_`α is to invest more significance, referred to as α-wealth, in candidates with the highest coverage. This decision relies on tuning a hyperparameter λ whose value determines the speed at which the α-wealth is consumed, and needs to be explored empirically.

4.4 Data Model

GroupTest is applicable to data modeled as a bipartite graph formed by users and items with their respective attributes (See Section 4.4.1). This model is generic enough to capture many datasets, especially the recommendation ones. We present some examples of the hypothesis and requests GroupTest encounter.

Examples. Our purpose is to develop a powerful framework for group testing. A one-sample, two-sample, or multiple-sample hypothesis is verified when *groups* identified by some *filters*, have statistically *similar, higher, lower aggregates* with respect to some aggregate (mean, variance, rating distribution on group members). This is referred to as the alternative hypothesis that states the desired test and complements the null hypothesis. Therefore, the null hypothesis is said to be rejected by desired groups.

Table 4.1 illustrates the variety of requests we handle in GroupTest with examples on movie ratings along with the type of test that is relevant for each request (third column). The first four types of requests use mean to aggregate group ratings and require different types of tests. R_1 shows the case of a one-sample t-test. Input data is all movie ratings by students. Subgroups such as “Students in California" or “Young students" are generated and their average rating is compared to a reference value (here, 3.5) with a one-sample test. Groups that reject the null hypothesis “Students whose rating average is equal to 3.5" and satisfy the

Table 4.1: Examples of group testing requests in GROUPTEST with "groups", **aggregate**, dimension, and *operator* with the corresponding statistical test.

R_1	"Student groups" whose rating **mean** *is greater than 3.5*	One-sample t-test
R_2	"Female groups" whose rating **mean** *is lower than* "Male groups" within the same period	Two-sample Welch's test
R_3	"Male Groups" whose rating **mean** *changes* between 2 Seasons	Two-sample paired t-test
R_4	"Groups" whose rating **mean** for "80's movies" *differs* in a 3-week period	Multiple mean F-test: ANOVA
R_5	"Groups" whose rating **variance** for "Comedy movies" *is greater than 1*	One-sample variance Chi-square test
R_6	"Group pairs" whose rating **variance** for"70's movies" *differs* in the Spring	Two-sample variance F-test
R_7	"Groups" whose yearly rating **distribution** *does not follow a Gaussian distribution*	One-sample Kolmogorov-Smirnov test
R_8	"Group pairs" whose rating **distribution** for"Drama movies" *differs* in the same season	Two-sample Kolmogorov-Smirnov test

alternative hypothesis "Students whose rating average is greater than 3.5" are returned. The case of a two-sample test is shown in R_2 and R_3. In R_4, we compare groups across 3 weeks and return tuples of groups whose rating mean for 80's movies differs. We use variance as an aggregation in R_6 and rely on a two-sample F-test. It starts with all rating records for movies in the 70s and returns pairs of groups whose rating variance for those movies differs in the Spring. The last two types, R_7 and R_8, compare rating distributions using one-sample and two-sample Kolmogorov-Smirnov tests.

4.4.1 Groups

Given a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$, we define *user data* as a database $\mathcal{D}$ of tuples $\langle u, i, a \rangle$ where $a \in \mathbb{R}$ is a value induced by an action such as browsing, tagging, or rating, of user $u \in \mathcal{U}$, on item $i \in \mathcal{I}$. Users have attributes drawn from a set $A_{\mathcal{U}}$ and items have attributes drawn from a set $A_{\mathcal{I}}$. For example, users can be represented with $A_{\mathcal{U}} = \langle$ uid, age, gender, occupation, location $\rangle$, a user instance may be $\langle$ 568, 18-24, female, student, NY $\rangle$. Similarly, items on Movielens can be represented with $A_{\mathcal{I}} = \langle$ title, genre, year, run time $\rangle$ and the movie *Titanic* as $\langle$ Titanic, Romance & Drama, 1997, 195 $\rangle$.

Definition 1 (Group)**.** *A group g is a set of records where users have at least one* common attribute value *(e.g., same gender) and may have some* common actions *(e.g., rated the same movies).*

For instance, $g = [\langle$gender, female$\rangle$, $\langle$age, 25-34$\rangle$, $\langle$ item title, Titanic $\rangle]$ contains 25-34 aged females who rated the movie Titanic.

We use $\mathcal{G}$ to denote the set of all groups. Hence, $|\mathcal{G}|$ is the powerset of user and item attribute values. For instance, with $|A_{\mathcal{U}}| = 4$ and 3 values per attribute, and $|A_{\mathcal{I}}| = 3$ with 5 values, $|\mathcal{G}| = 2^{(4\times3+3\times5)}$.

In the literature, groups have been referred to with different terms, such as *communities* [169],

tribes [103], *cliques* [47], *cohorts* [128], *teams* [171], *segments* [15], *patterns* [261, 40], *cubes* [129], *clusters* [9, 232] and *partitions* [187]. Our model is designed to be agnostic about the approach used to compute groups.

4.4.2 Group Testing

A hypothesis test considers two hypotheses that contain opposing viewpoints. The null hypothesis H_0 usually states that group aggregates are the same. The alternative hypothesis H_a states a claim that contradicts H_0 and corresponds to desired samples (in our case, user groups). The decision can either be "reject H_0" if the sample favors the alternative hypothesis or "do not reject H_0" if the sample is insufficient to reject that hypothesis.

The GroupTest framework is aimed to be generic and accommodates various types of tests. Different statistical tests qualify depending on group size, group members (paired or unpaired), and the aggregation function `AGG`. Figure 4.2 summarizes the aggregation functions and statistical tests considered in our work. The last column refers to the requests shown in Table 4.1.

Definition 2 (Group testing request)**.** *A group testing request R is a tuple $\langle H_0, H_a, \texttt{MSR}, \texttt{AGG}, \texttt{OP}\rangle$ where H_0 is a null hypothesis, H_a an alternative hypothesis,* `MSR` *is a user behavior dimension (e.g., rating, purchase),* `AGG` *is an aggregation function applied to a behavior dimension (average, variance, distribution), and* `OP` *the operator used to compare aggregates (=, <>, >, and <).*

To simplify our notation, we omit the condition on user and item attributes in R and assume that a request R is evaluated on $D \subseteq \mathcal{D}$ where those conditions are satisfied. The subset D is used to create a set of groups $G \subseteq \mathcal{G}$. To evaluate a request R, we compute a set of *allCandidates* as follows: for one-sample tests, $allCandidates = \{\langle g\rangle\}$; for two-sample tests, $allCandidates = \{\langle g, g'\rangle\}$; for multiple-sample tests $allCandidates = \{\langle g, g', \ldots\rangle\}$ where $g \in G, g' \in G, g <> g'$. We now describe how to compute the significance of each sample in *allCandidates* before we formalize our top-n problems.

Computing P-values The common protocol to compute p-values of each sample in *allCandidates* must first verify the normality and independence of each sample [62]. Without loss of generality, we describe that protocol *for comparing two means with a two-sample t-test.* A value of 0.05 for α is commonly adopted and indicates a 5% risk of concluding that a difference exists between the two means when there is no actual difference [75]:

P-value Computation Protocol:

Normality check: Given a candidate pair $(g, g') \in allCandidates$, verify that the data distribution of each group g and g' is normal, normalize it otherwise;

Independence filtering: Verify that the distributions of g and g' are independent using χ^2 test; Keep independent pairs;

P-value computation: Compute the p-value *pval* of independent (g, g') pairs wrt a request R.

AGGREGATE	INFERENCE and TEST	DEFINITION and Example	
Mean	Test about a mean: One-sample t-test	$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$ with $\bar{x}$ and s, the sample mean and standard deviation, and μ_0 the reference mean, and n the sample size	R1
	Test to compare two means: Two-sample Welch's t-test	$t = \frac{(\bar{x}_1 - \bar{x}_2)}{\eta}$ with $\bar{x}_1 - \bar{x}_2$ the difference between 2 sample means, s the pooled standard deviation, and $\eta = \begin{cases} s\sqrt{2/n}, & \text{for } n = n_1 = n_2 \\ s\sqrt{1/n_1 + 1/n_2}, & \text{for } n_1 \neq n_2 \text{ and } s = s_1 = s_2 \\ \sqrt{s_1/n_1 + s_2/n_2}, & \text{for } n_1 \neq n_2 \text{ and } s_1 \neq s_2 \end{cases}$	R2
	Test about a mean with paired data: Paired difference t-test	$t = \frac{\bar{d} - d_0}{s_d/\sqrt{n}}$ with $\bar{d}$ and s_d the average and standard deviation of the differences all pairs and the reference difference d_0	R3
	Test to compare K multiple means: F-test for one way ANOVA	$F = \frac{MST}{MSE}$ with $MST = \frac{\sum_{i=1}^{K} n_i(\bar{x}_i - \bar{x})^2}{K-1}$, $MSE = \frac{\sum_{i=1}^{K}(n_i - 1)s_i^2}{n-K}$ and $n = n_1 + \cdots + n_K$, $\bar{x} = \frac{\sum_{i=1}^{K} x_i}{n}$	R4
Variance	Test about a population variance: Chi-squared test	$T = (n-1)\ s^2/\sigma_0^2$ with s^2 the sample variance, n the sample size and σ^2 the reference variance	R5
	Test to compare two population variances: F-test	$F = s_1^2/s_2^2$ with s_1^2 and s_2^2 the sample variances of the 2 populations	R6
Distribution	Test about a distribution: One-sample Kolmogorov–Smirnov test	$D_n = sup_x \lvert F_n(x) - F_0(x) \rvert$ with F_n the empirical distribution function, F the reference distribution and sup the supremum function	R7
	Test to compare two distributions: Two-sample Kolmogorov-Smirnov test	$D_{n,m} = sup_x \lvert F_{1,n}(x) - F_{2,m}(x) \rvert$ with $F_{1,n}, F_{2,m}$ the empirical distribution functions of the two samples and sup is the supremum function	R8

Figure 4.2: Summary of statistical tests considered in GroupTest.

4.5 GroupTest Problems Formalization

In the following problems, the set *Candidates* contains the tuples $(g, g', pval)$ in D that passed p-value computation protocol wrt the hypothesis in R ($pval \leq \alpha$), $c.groups$ denotes the group g, the pair (g, g') and the tuple $(g, g', ...)$ in c in the case of one-sample, two-sample and multiple-sample tests respectively. $cover(g, D)$ is defined as the intersection between users in group g and all users in dataset D. Formally, for a given group g, $cover(g, D) = g.users \cap D.users$. Moreover, *cover* is defined in the same way for all statistical tests and determined by the intersection between the union of all users in the groups $c.groups$ and the users in the dataset D. For example, coverage of a two-sample set of candidates $C = \{c_1, c_2\}$ is:

$$
\begin{aligned}
cover(\bigcup_{c.groups \in C}, D) &= cover(c_1.groups \cup c_2.groups, D) \\
&= cover((g_1 \cup g_1') \cup (g_2 \cup g_2'), D) \\
&= ((g_1.users \cup g_1'.users) \cup (g_2.users \cup g_2'.users)) \cap D.users
\end{aligned}
\tag{4.1}
$$

The definition of *Cover* for a larger set of candidates is straightforward. We now formalize our problems which are summarized in Table 4.2 with their solutions.

Table 4.2: Summary of GroupTest problems.

Problem Name	Description	Solutions
Problem 4: ValMin	It optimizes significance while setting a constraint on data coverage	`VAL_C` (Section 4.6.1)
Problem 5: CovMax	It aims to maximize data coverage while controlling significance	`COVER_G` (Section 4.6.2) `COVER_`α (Section 4.6.3)

Problem 4 (ValMin Problem)**.** *Given a request R, a dataset $D \subseteq \mathcal{D}$ that satisfies user and item conditions, a significance threshold θ on p-values, a minimum coverage value cov_{min}, a maximum number of desired results n, find a set C s.t.*

$$
\begin{aligned}
C = & \underset{C \subseteq Candidates}{\textit{argmin}} \sum_{c \in C} c.pval \\
\textit{subject to} & \\
& |C| \leq n, \\
& \forall c \in C\ , c.pval \leq \theta, \\
& |cover(\bigcup_{c.groups \in C}, D)| \geq cov_{min}
\end{aligned}
\tag{4.2}
$$

The ValMin Problem extends the multiple hypothesis testing by enforcing data representativity and having a constraint on coverage but it does not assure its maximization. We propose the CovMax Problem to face this limitation and capture data coverage.

Problem 5 (CovMax Problem)**.** *Given a request R, a dataset $D \subseteq \mathcal{D}$ that satisfies user and item conditions, a significance threshold θ on p-values, a maximum number of desired results n, find a set C s.t.*

$$
\begin{aligned}
C = & \underset{C \subseteq Candidates}{\textit{argmax}} |cover(\bigcup_{c.groups \in C}, D)| \\
\textit{subject to} & \\
& |C| \leq n, \\
& \forall c \in C\ \ c.pval \leq \theta
\end{aligned}
\tag{4.3}
$$

As the number of candidates increases, the likelihood that spurious hypotheses pass the test increases, causing Type I errors [208]. The significance level of p-values can be adjusted to control the expected proportion of incorrectly rejected null hypotheses. The simplest way

to do so is to use the conservative Bonferroni correction [30], a Family-Wise Error Rate (FWER) control method. Bonferroni is preferred when false discoveries are not acceptable (in particular for critical decision-making, e.g., accepting a new medical treatment) or when it is expected that most null hypotheses would be true. A more powerful adjustment method is the Benjamini-Yekutieli False Discovery Rate (FDR) procedure [34] that allows to control the expected proportion of incorrectly rejected null hypotheses. FDR control is preferred in exploratory research, where the number of potential hypotheses is large and false discoveries are not so critical [107].

Our problem formulation is generic and aims to accommodate existing significance adjustment procedures by adapting the definition of the significance threshold θ as follows:

- For Bonferroni (BN): $\theta = \frac{\alpha}{m}$
- For Benjamini-Yekutieli (BY): $\theta = \frac{\alpha \times k}{m} \left(\sum_{i=1}^{m} 1/i \right)^{-1}$, where $k = max \left\{ i : p_i \leq \frac{\alpha \times i}{m \times \sum_{i=1}^{m} 1/i} \right\}$, p_i the i^{th} smallest p-value.

The value m is the number of groups in *Candidates* and α is the significance level usually set to 0.05 [75].

The drawback of these traditional adjustment methods is that to control FDR (or FWER) at a given level α, one has to previously calculate the p-values of all m candidates and then rank them to determine the correct threshold θ for rejecting the null hypothesis. This has two main limitations: (1) the calculation of p-values for all candidate groups is expensive and (2) there could be settings where we do not have prior knowledge on the number of hypothesis m. To overcome this, we leverage the marginal FDR (*mFDR*) that was defined by Foster and Stine [89] that computes the ratio of the expected number of false rejections to the expected number of rejections as follows:

$$mFDR_{\eta}(j) = \frac{E[V(j)])}{E([R(j)]) + \eta} \tag{4.4}$$

where $V(j)$ designates the number of false discoveries (wrongly rejected null hypothesis) and $R(j)$ the total number of discoveries. The parameter η is used to weigh the impact of cases for which the number of discoveries is small, and η is usually set to 1 or $(1 - \alpha)$ [273]. We can now reformulate the CovMax problem as follows:

Problem 6 (CovMax Problem reformulation)**.** *Given a request R, a dataset $D \subseteq \mathcal{D}$ that satisfies user and item conditions, a significance level α, a parameter η, a maximum number of desired results n, find a set C s.t.*

$$\begin{aligned} C = & \underset{C \subseteq Candidates}{\texttt{argmax}} \; |cover(\bigcup_{c.groups \in C}, D)| \\ & \textit{subject to} \\ & |C| \leq n, \\ & mFDR_{\eta}(j) \leq \alpha \end{aligned} \tag{4.5}$$

where j denotes the total number of hypothesis tests that have been performed.

Theorem 1. ValMin *is NP-hard.*

Proof. Given $S = \{S_1, S_2, ..., S_m\}$ a collection of m sets where each set S_i is a subset of D which represents the set of all data elements. We assign to each set S_i a weight p_i. We want to identify the n sets that minimize the total weight and whose union covers a fraction β of D. This is known as the Partial Weighted Set Cover Problem [55] and is formulated as follows:

$$\begin{aligned} & C = \underset{i \subseteq \{1,..,m\}}{\texttt{argmin}} \sum p_i \\ \text{subject to} \quad & \\ & |C| \leq n, \\ & \frac{|\bigcup_{i \in C} S_i|}{|D|} \geq \beta \end{aligned} \tag{4.6}$$

This problem is equivalent to the ValMin when we add the significance constraint and we correspond each set S_i to the coverage of a candidate c_i from a set of m candidates, $S_i = cover(\cup_{c_i.groups}, D)$, each weight p_i to the p-value of the test of the candidate c_i, and the fraction β to the minimum coverage value cov_{min}.

The Partial Weighted Set Cover Problem represents a generalization of the Weighted Set Cover Problem [55] which is proved to be NP-hard [130]. This is sufficient to prove that our ValMin problem is NP-hard.

□

Theorem 2. CovMax *is NP-hard.*

Proof. Given $S = \{S_1, S_2, ..., S_m\}$ a collection of m sets where each set S_i is a subset of D which represents the set of all data elements. We want to identify the n sets that maximize the total coverage of D. This is known as the Maximum Coverage Problem and is formulated as follows:

$$\begin{aligned} & C = \underset{i \subseteq \{1,..,m\}}{\texttt{argmax}} \, | \bigcup_{i \in C} S_i| \\ \text{subject to} \quad & \\ & |C| \leq n, \end{aligned} \tag{4.7}$$

This problem is equivalent to the CovMax when we add the significance constraint and we correspond each set S_i to the coverage of a candidate c_i from a set of m candidates, $S_i = cover(\cup_{c_i.groups}, D)$.

The Maximum Coverage Problem is proved to be NP-hard [116] which makes it sufficient to prove by reduction that our CovMax problem is NP-hard.

□

4.6 Our Proposed Solutions

Without loss of generality, we illustrate our algorithms in the case of a request that requires two-sample tests. We first describe how the set of candidate groups *Candidates* is generated. We define a sub-routine *GenerateCandidates* that takes a request R, a dataset D, and generates the set of all groups *allCandidates*. For example, for R_2 in Table 4.1, it generates all groups that share the attribute value *female* and all groups that share the attribute value *male* and creates *allCandidates* that contains pairs of groups (g, g') formed by a Cartesian product between the two sets. After that, another sub-routine *ComputePvalues* computes the p-value of each pair (g, g') in *allCandidates*, discards all pairs that have a p-value above the significance level α and outputs a set *Candidates* of pairs along with their p-values.

4.6.1 `VAL_C` Solution

Algorithm 6: Minimum coverage algorithm (`VAL_C`) – illustrated with the Benjamini-Yekutieli correction

Input: a request R, a dataset D, a significance level α, a minimum coverage value cov_{min}, a number of results n

Output: C

1 $allCandidates \leftarrow GenerateCandidates(R, D)$
2 $Candidates \leftarrow ComputePvalues(allCandidates, \alpha)$
3 $C \leftarrow \emptyset$
4 $m \leftarrow |Candidates|$
5 $L = Sortbypval\ (Candidates)$
6 $k = \text{argmax}_{0 \leq j \leq m} P[j] \leq \frac{\alpha \times j}{m} \left(\sum_{i=1}^{m} 1/i \right)^{-1}$
7 $C \leftarrow Top\text{-}n(L)$
8 $cov_C \leftarrow |cover(C.groups, D)|$
9 $L \leftarrow L \setminus C$
10 $i \leftarrow n + 1$
11 **while** $cov_C < cov_{min}$ *and* $i \leq k$ **do**
12 $c^* \leftarrow L[i]$
13 $i \leftarrow i + 1$; $L \leftarrow L \setminus \{c^*\}$
14 $C_{future} \leftarrow C$
15 **for** *each group* $\in C$ **do**
16 $C^* \leftarrow Swap(C, group, c^*)$
17 **if** $|cover(C^*.groups, D)| > cov_C$ **then**
18 $cov_C \leftarrow |cover(C^*.groups, D)|$
19 $C_{future} \leftarrow C^*$
20 $C \leftarrow C_{future}$
21 **return** C

To solve ValMin, we propose a greedy algorithm `VAL_C` (Algorithm 6). It first generates *allCandidates* by calling the routine *GenerateCandidates()* (Line 1), computes their p-values (Line 2), and keeps the ones having significant p-values (*Candidates*). Then, it sorts the candidates by the increasing order of their p-values into a list L (Line 5), before applying the significance adjustment procedure (Line 6). After that, it picks the set C of n candidates that have the smallest p-values and that are below the significance threshold $P[k]$ (Line 7),

calculates their coverage (Line 8), and removes them from the set of candidates L (Line 9). If the selected groups C do not satisfy the minimum coverage constraint (cov_{min}) (Line 11), the algorithm will greedily scan the next candidates and at each step, it will swap the pre-selected group in C that contributes the least to coverage with the considered candidate. Formally, `VAL_C` iterates through the remaining candidates (Line 12-13). The next candidate c^* is used to maximize the coverage in an iterative way (Line 15) where at each iteration a single pre-selected group from C is replaced by c^* generating by that a new set of groups C^* (Line 16). If the new set brings more coverage (Line 17), the procedure takes it as a potential future set of results (Line 19). The best set (the one that maximizes the most data coverage) is used to replace C (Line 20), otherwise C remains unchanged. The procedure stops either if the minimal coverage is reached or if all significant candidates were scanned.

The worst-case complexity of `VAL_C` is $O(m \cdot p + m \cdot log\ m + (k-n) \cdot n)$. The first term $m \cdot p$ represents the complexity of calculating the p-values of all m candidates by assuming that p is the worst-case complexity for computing a single p-value. The term $m \cdot log\ m$ is for sorting the candidates by their p-values and the term $(k-n) \cdot n$ is the complexity of scanning the remaining candidates and optimizing coverage. $k-n$ represents the number of the remaining candidates (k being the highest number of hypotheses that satisfy the constraint of multiple testing) and n represents the number of swaps performed for each candidate (Line 15-16 in Algorithm 6). We note that the number of candidates m is equal to the power set of attributes in the worst case.

4.6.2 `COVER_G` Solution

To solve CovMax, we propose `COVER_G` (Algorithm 7), a greedy algorithm. It first generates *allCandidates* by calling *GenerateCandidates()* (Line 1), computes their p-values, and discards the nonsignificant ones (Line 2), and then it sorts the candidates by increasing p-values into a list L (Line 3). The next step (Line 6) applies the significance adjustment procedure. We illustrate it here with Benjamini-Yekutieli. In this case, the algorithm calculates the greatest number k for which $L[j] \leq \frac{\alpha \times j}{m} \left(\sum_{i=1}^{m} 1/i \right)^{-1}$ is verified. After that, it greedily picks the next candidate that maximizes data coverage (Line 8) and removes it from the set of candidates (Line 9). It adds it to the final set of results if its p-value is smaller than the significance threshold $L[k]$ (Lines 10-11). This procedure is repeated until the size of the results reaches n. Even, if it scans candidates in decreasing order of their coverage, `COVER_G` controls the false discovery rate at level α by adjusting their p-values using the Benjamini-Yekutieli procedure.

We highlight that `COVER_G` has a $(1-1/e)$ approximation guarantee [8] as our problem has a one-to-one reduction to Maximum Coverage (proof sketch in Section 4.5).

The worst-case complexity of `COVER_G` is $O(m \cdot p + m \cdot log\ m + m \cdot n)$. The complexity of calculating a single p-value depends on the type of test and the size of the compared groups. Assuming the worst-case complexity for computing a single p-value is p, the term $m \cdot p$ denotes the complexity of calculating the p-values of all m candidates. The term $m \cdot log\ m$ is for sorting the candidates by their p-values and the term $m \cdot n$ is the complexity of the greedy scans to select the next best candidate at each step.

Algorithm 7: Greedy coverage algorithm (`COVER_G`) – illustrated with the Benjamini-Yekutieli correction

Input: a Request R, a dataset D, a significance level α, number of desired results n
Output: C

1 $allCandidates \leftarrow GenerateCandidates(R, D)$
2 $Candidates \leftarrow ComputePvalues(allCandidates, \alpha)$
3 $L = Sortbypval\ (Candidates)$
4 $C \leftarrow \emptyset$
5 $m \leftarrow |Candidates|$
6 $k = \text{argmax}_{0 \leq j \leq m} L[j] \leq \frac{\alpha \times j}{m} \left(\sum_{i=1}^{m} 1/i \right)^{-1}$
7 **while** $|C| \leq n$ **do**
8 $\quad c^* = \text{argmax}_{c \in Candidates} |cover(C.groups \cup \{c.groups\}, D)|$
9 $\quad Candidates \leftarrow Candidates \setminus \{c^*\}$
10 $\quad$ **if** $c^*.pval \leq L[k]$ **then**
11 $\quad\quad C \leftarrow C \cup \{c^*\}$
12 **return** C

4.6.3 `COVER_`α Solution

To solve Problem 6, we revisit α-investing, a method for multiple hypothesis testing that controls specifically *mFDR*. α-investing was originally introduced by Foster and Stine [89], and generalized by Zhao *et al.* for data exploration [273]. Intuitively, α-investing works as follows: it starts with an initial wealth, set to $\eta.\alpha$, then at each step j a specific threshold α_j is defined, which is below the current available wealth. If the null hypothesis is accepted ($p_j > \alpha_j$) a fraction of the invested value is lost and is subtracted from the current available wealth $W(j)$. If the null hypothesis is rejected ($p_j \leq \alpha_j$), we obtain a "return" on investment $\omega \leq \alpha$. The testing procedure stops when the available α-wealth is totally consumed, i.e., reaches 0. We choose $\eta = 1$ and $\omega = \alpha$ as it was shown in [89], any α-investing algorithm controls $mFRD_\eta$ at level α for $W(0) = \eta \cdot \alpha$ and $\omega = \alpha$ for any $\eta, \alpha \in [0, 1]$.

The key idea of `COVER_`α is to re-adjust the quantities of the α-wealth that are invested according to the coverage of each selected candidate. Different policies for investing the wealth could be explored [273]. In our solution, we design `COVER_`α in such a way that it invests more α-wealth on candidates that bring higher coverage of the input data D. In Section 4.7, we also implement existing variants and compare them to `COVER_`α.

Algorithm 8 contains the pseudo-code of `COVER_`α. It starts with generating the set of candidates by calling the sub-routine *GenerateCandidates()* (Line 1). However, unlike `COVER_G`, it does not compute p-values of all candidates as it relies on *mFDR*. It initializes α-wealth (Line 2) with η set to 1. The adjustment value of the hypothesis testing is initialized with a fixed parameter λ (Line 3) which controls how much of the available α-wealth is invested during each step. In our experiments (Section 4.7.4), we vary the parameter λ and show that higher values are preferred. The set of returned results is initialized with the empty set (Line 4). While the number of results added to the set is less than the number of desired results and the value of α-wealth remains positive, the algorithm picks the next candidate that maximizes coverage (Line 7) and removes it from the set of candidates (Line 8). It then sets the current α_j value according to the coverage of the selected candidate (Line 9) and checks the

Algorithm 8: α-investing coverage algorithm (COVER_α)

Input: a request R, a dataset D, a significance level α, number of results n, parameters λ and η
Output: C

1 $Candidates \leftarrow GenerateCandidates(R, D)$
2 $W(0) \leftarrow \eta \cdot \alpha$
3 $\alpha^* = \frac{W(0)}{\lambda + W(0)}$
4 $C \leftarrow \emptyset$
5 $j \leftarrow 1$
6 **while** $W(j-1) > 0$ *and* $|C| \leq n$ **do**
7 $\quad c^* = \text{argmax}_{c \in Candidates} |cover(C.groups \cup \{c.groups\}, D)|$
8 $\quad Candidates \leftarrow Candidates \setminus \{c^*\}$
9 $\quad \alpha_j = \alpha^* (\frac{|cover(c^*.groups, D)|}{|D|})^{1/2}$
10 $\quad$ **if** $W(j-1) - \frac{\alpha_j}{1-\alpha_j} \geq 0$ **then**
11 $\quad\quad$ **if** $c^*.pval \leq \alpha_j$ **then**
12 $\quad\quad\quad W(j) \leftarrow W(j-1) + \alpha$
13 $\quad\quad\quad C \leftarrow C \cup \{c^*\}$
14 $\quad\quad$ **else**
15 $\quad\quad\quad W(j) \leftarrow W(j-1) - \frac{\alpha_j}{1-\alpha_j}$
16 $\quad j \leftarrow j + 1$
17 **return** C

availability of α-wealth (Line 10). Unlike COVER_G, the threshold for each hypothesis test is not fixed but is dependent on how much coverage the candidate adds to the current solution. COVER_α compares the p-value of the selected candidate c^* with its threshold α_j. If the null hypothesis is rejected (Line 11), α is added to the current α-wealth (Line 12), and c^* is added to the set of results (Line 13). Otherwise, if the null hypothesis is accepted, the amount $\frac{\alpha_j}{1-\alpha_j}$ is lost and is subtracted from the α-wealth accordingly (Line 15).

COVER_α greedily scans n times the set of candidates of size m, retrieves at each step the best candidate c^* that has the maximum coverage, and computes its p-value with an $O(p)$ worst-case complexity. This gives us a $O(m \cdot n \cdot p)$ worst-time complexity for COVER_α which is much faster than COVER_G. It benefits from computing p-values on-the-fly and only for candidates with the highest coverage at each iteration.

4.7 Experiments

Our experiments aim to: (1) demonstrate the expressivity of GroupTest on realistic scenarios (Section 4.7.1); (2) study the results of VAL_C in terms of coverage and scalability and compare it to different baselines (Section 4.7.3); (3) study hypothesis significance and the interplay between coverage and significance for CovMax algorithms (Section 4.7.4). Most of our experiments use request R_8 in Table 4.1 that returns the highest number of results. Our code and complete results are available on our Github repository[2].

[2] https://github.com/statistical-group-testing/statistically-soundgrouping

4.7.1 Addressing Information Needs

We describe two scenarios using a Traditional multiple hypothesis algorithm based on Benjamini-Yekutieli (`TRAD_BY`) and `COVER_G` to illustrate the expressivity of GroupTest.

Group Evolution: We use `TRAD_BY` with a multiple mean F-test to verify the request "Groups whose average rating for a movie genre changes monthly (in a 3-month period)" (akin to R_4 in Table 4.1 with an unlimited n). Table 4.3 reports for each movie genre the number of groups that exhibit the monthly change along with examples and the movie genres for which that change is observed. For instance, the average rating of [35-44] aged females who rated Comedies from the 60's changes monthly.

Table 4.3: Number and examples of groups per genre that satisfy the request "Groups whose average rating for a movie genre changes monthly (in a 3-month period)".

Genres	#Groups	Example groups
All	6	[18-24] aged females who rated 50's movies
Drama	2	[35-44] aged females who rated 70's movies
Horror	2	Males who rated 80's movies
Action	2	Reviewers whose occupation is in customer-service
Science-Fiction	2	Male students under 18
Comedy	4	[35-44] aged females who rated 60's movies

Group Comparison: We use `COVER_G` with a two-sample Kolmogorov-Smirnov test to compare rating distributions of group pairs within the same age category (akin to R_8 in Table 4.1 with $n = 20$). Table 4.4 reports for each age category, the number of returned pairs along with examples. For instance, we find 14 pairs for age [25-34] among which the two groups who rated War movies and 90's Thrillers significantly differ in their rating distribution.

Table 4.4: Number and examples of pairs per age that satisfy the request "Group pairs whose rating distribution for Drama movies differs in the same season (Summer)".

Age	#Pairs	Example pairs
<18	2	(Drama movies - Comedy movies)
18-24	7	(Male college student- Male users)
25-34	14	(War movies - 90's Thriller movies)
35-44	10	(Males who rated 80's movies - Male who rated 70's movies)
45-49	5	(Males who rated 2000's movies - Females who rated 90's movies)
50-55	4	(Users who rated 90's Thrillers - Users who rated 50's movies)

4.7.2 Experimental Setup

Data: We report results on `MovieLens` 1M. Similar results were found on `Yelp`, `Tafeng`, and `BookCrossing` datasets. Our complete results are available on our Github repository. The `MovieLens` 1M dataset contains user-movie ratings collected from a movie recommendation service. It contains around 1M ratings given by $6,040$ users to $3,900$ movies. The data also contain user attributes: gender, age group, occupation, and location, as well as item attributes that correspond to a set of genres.

Metrics: We examine for each request, the groups that are found in terms of (1) their significance (min/max/sum p-values), (2) data coverage, (3) power (#true positives/#results

in ground truth) and FDR (#false positives/#results), (4) response time. All results are averages of 10 runs.

ValMin Baselines and Variants: In section 4.7.3, we implement two variants of VAL_C, one with Bonferroni noted VAL_C_BN and the other one with Benjamini-Yekutieli noted VAL_C_BY. We compare the algorithm against an adaptation of the Subfamily wise Multiple Testing Procedure (SMT) [254] where we add a constraint on coverage cov_{min} (called SMT_{cov}). This method controls the FWER in multiple testing where hypotheses are organized into families. The reader is referred to Section 2.2 for more details.

In our adaptation, the families are generated randomly using one hyperparameter. It represents the number of hypotheses in each family. We compare different variants by varying the hyperparameter in {10,50,100,500,1000,5000}. In the experiments, we compare VAL_C against the traditional correction method, referred to as TRAD with Benjamini-Yekutieli (TRAD_BY), the original SMT and its adaptation (SMT_{cov}) varying cov_{min} in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$.

CovMax Baselines and Variants: In section 4.7.4, we compare variants of COVER_α by varying the hyper parameter λ in $\{20, 50, 100, 200, 500\}$. We compare our algorithms, the best COVER_α and the two variants of COVER_G, one with Bonferroni noted COVER_G_BN and the other one with Benjamini-Yekutieli noted COVER_G_BY, against the traditional correction method, referred to as TRAD. Similarly to COVER_G, we implemented two variants of TRAD, one with Bonferroni correction that we note TRAD_BN and the other with Benjamini-Yekutieli that we note TRAD_BY. We also implemented previously proposed α-investing policies [273] as baselines for our comparisons. The reader may refer to Section 2.2 for details of these policies.

Each α-investing policy was tested with different parameter values (in Table 4.5). We selected the best value (last column in the table) based on power and FDR. For instance, for our algorithm COVER_α, we observed that $\lambda = 500$ yields the best power and FDR since smaller λ make COVER_α consume its α-wealth faster. α is set to 0.05.

Table 4.5: Parameters for α-investing policies.

Investigation policies	Parameter	Values	Best Value
COVER_α	λ	$20, 50, 100, 200, 300, 500$	**500**
β-Farsighted	β	$0.25, 0.5, 0.75, 0.9$	**0.9**
γ-Fixed	γ	$20, 50, 100, 200, 300, 500$	**500**
δ-Hopeful	δ	$20, 50, 100, 200, 300, 500$	**500**
ϵ-Hybrid	ϵ	$0.25, 0.5, 0.75, 0.9$	**0.75**
ψ-Support	ψ	$1/2, 1/3, 1/4, 1/5, 1/6$	**1/2**

4.7.3 ValMin Results

Study of SMT_{cov} Variants: In this section, we study the significance and coverage of the different SMT_{cov} variants by varying the value of the size of families (number of hypotheses in each family). As in [273], we use TRAD with Bonferroni (TRAD_BN) as ground truth since it is the most conservative correction and compare the results of SMT_{cov} variants. We consider data samples ranging from 10% to 100% of *Candidates* and report the results for R_8.

Figure 4.3 reports power for $n = 100$ for R_8. FDR average results are all the same and equal

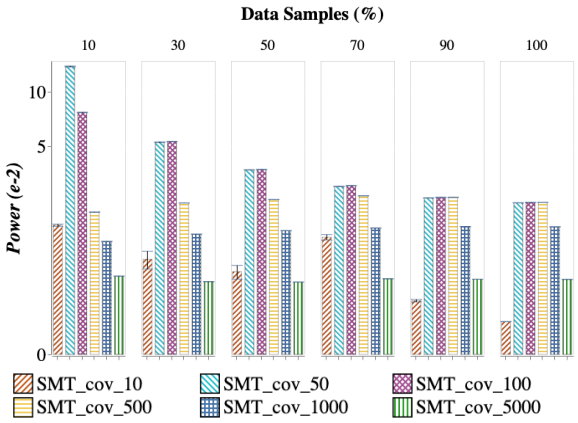


Figure 4.3: Impact of coverage optimization on Power with $cov_{min} = 0.5$ and number of results $n = 100$ for different percentages of data samples.

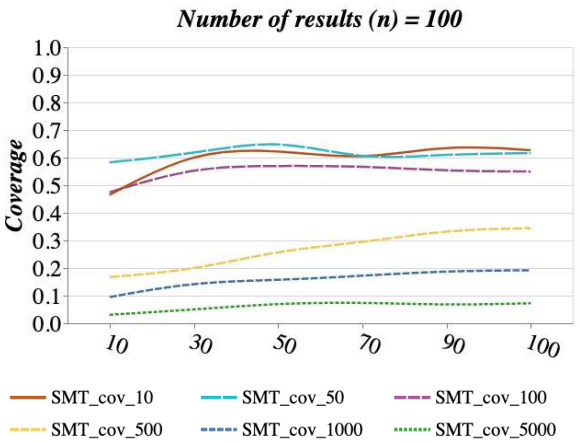


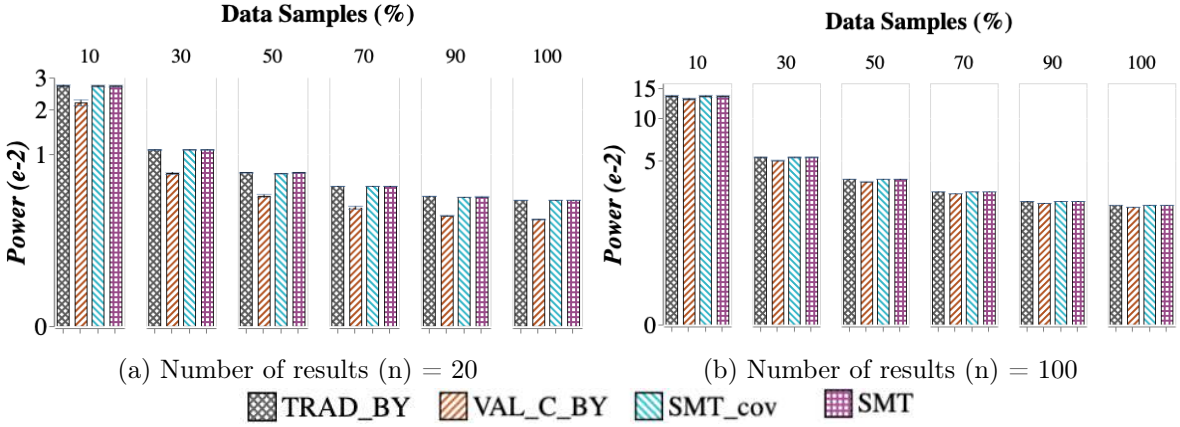
Figure 4.4: Coverage as a function of the number of data samples with $cov_{min} = 0.5$ and the number of results $n = 100$.

to zero. The main observation is that $\texttt{SMT}_{cov}$ with a family size of 50 performs better than all other variants. Significance power decreases as the value of family size increases. Large values $(500, 1000, 5000)$ make the families too massive with hypotheses. As $\texttt{SMT}_{cov}$ rejects one and only one hypothesis in each family, many valid and potential true discoveries will be skipped. This will result in a weak power score. A similar effect can be observed with small values of size (10). The families will be, in this case, narrow but many true discoveries may be part of the same family.

Figure 4.4 reports results of coverage as a function of the number of data samples with results number $n = 100$ for $\texttt{R}_8$. We note that the coverage is nearly constant as the number of samples increases for all cases. One can observe that $\texttt{SMT}_{cov}$ variants with small family size achieve a coverage greater, and in some cases slightly smaller, than $cov_{min} = 0.5$ while variants with large size don't cover more than 35% of data.

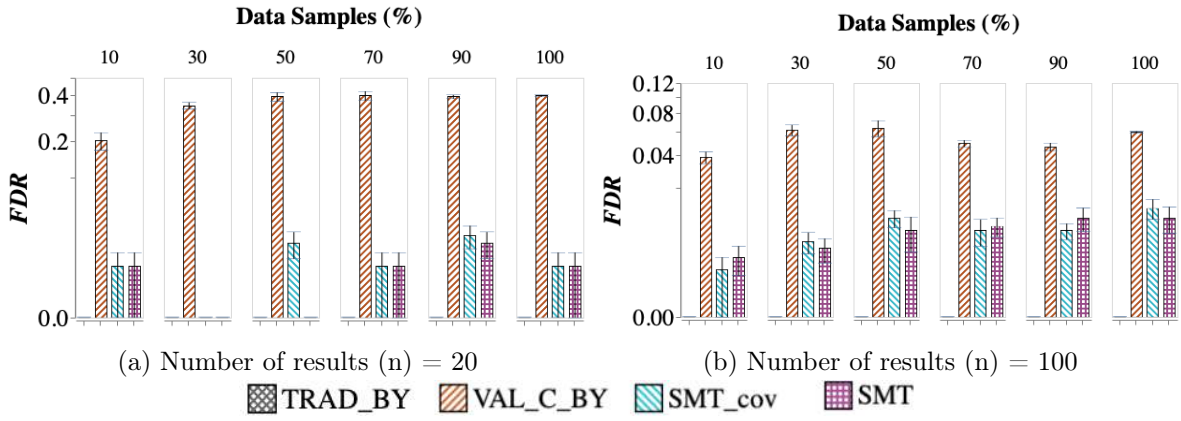
`VAL_C` Hypothesis Significance: We study, in this part, the significance by comparing `VAL_C_BY`, the best variant of `SMT` and $\texttt{SMT}_{cov}$ (with family size 50) and `TRAD_BY` for $\texttt{R}_8$.

Figures 4.5, 4.6 report power and FDR for $n = 20$ **(left)** and $n = 100$ **(right)** with $cov_{min} = 0.7$ respectively. We observe that, for a small number of results, `SMT` and its adaptation $\texttt{SMT}_{cov}$ outperform `VAL_C_BY` in terms of power and FDR. One can also observe that `SMT` behaves similarly to `TRAD_BY` in terms of power. On the other hand, for larger results values, the performances of `VAL_C_BY` are better and are slightly outperformed by the baselines for power but still much worse for FDR regardless of the sample size. One possible explanation for this difference is the closeness between the ground truth and the results of the baseline $\texttt{SMT}_{cov}$. Indeed, as explained in Section 4.7.2 `SMT` procedure controls the Family-Wise Error rate (FWER) as well as `TRAD_BN` that is used to generate the ground-truth. On the contrary, `VAL_C_BY` controls FDR which makes it disadvantageous in terms of significance. For the small number of results, $\texttt{SMT}_{cov}$ behaves approximately like `TRAD_BN` due to the similarity of their policies while `VAL_C_BY` explores more the set of potential discoveries in order to satisfy the coverage constraint making by that more false discoveries. As n increases, $\texttt{SMT}_{cov}$ explores



(a) Number of results (n) = 20 (b) Number of results (n) = 100

Figure 4.5: Impact of coverage optimization on significance (Power) with $cov_{min} = 0.7$, number of results $n = 20$ **(left)** and $n = 100$ **(right)** for different percentages of data samples.



(a) Number of results (n) = 20 (b) Number of results (n) = 100

Figure 4.6: Impact of coverage optimization on significance (FDR) with $cov_{min} = 0.7$, number of results $n = 20$ **(left)** and $n = 100$ **(right)** for different percentages of data samples.

more the set of candidates resulting in a relative increase of its false discoveries causing by that a more similar power rate compared to `VAL_C_BY`.

`VAL_C` Data Coverage: Figure 4.7 reports results of coverage as a function of the number of data samples with results number $n = 20$ **(left)** and $n = 100$ **(right)** for $\mathtt{R}_8$. We observe that for the small number of results, the coverage is nearly constant for $\mathtt{SMT}_{cov}$ and increases for `VAL_C_BY` as the number of samples increases. One can see that `VAL_C_BY` achieves a coverage rate greater than the threshold $cov_{min} = 0.7$ while the best $\mathtt{SMT}_{cov}$ algorithm doesn't satisfy this constraint. Obviously, one can see that `TRAD_BY` and `SMT` are the worst in terms of coverage as they are not developed to satisfy it. For a large number of results, `VAL_C_BY` still outperforms the baseline and satisfies the coverage constraint for all data samples unlike $\mathtt{SMT}_{cov}$ which does not satisfy it for small and medium data samples $(10\%, 30\%, 50\%)$. One possible explanation is that $\mathtt{SMT}_{cov}$ returns more granular and by that more overlapping groups. Indeed, significant groups with high coverage rates may be part of families where more significant groups but less covered are also members. As the most significant group is chosen from each family, the granular one is finally returned.

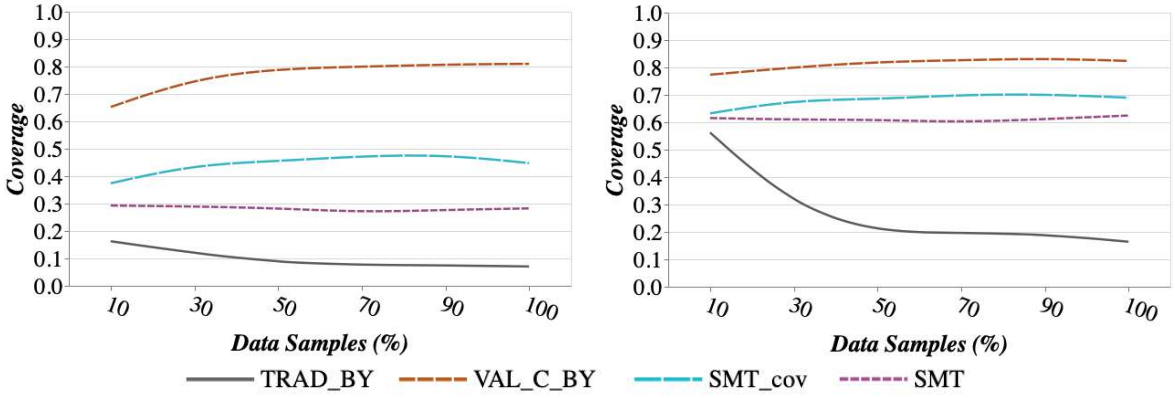


Figure 4.7: Coverage as a function of the number of data samples with $cov_{min} = 0.7$, number of results $n = 20$ **(left)** and $n = 100$ **(right)**.

We now examine simultaneously the evolution of the cumulative coverage and p-values. Results are depicted in Figure 4.8 for $n = 20$. The graph clearly shows that, `VAL_C_BY` reaches cov_{min} by iteration 19 while $\texttt{SMT}_{cov}$ never does. One can also observe that after satisfying the constraint, the cumulative p-value of `VAL_C_BY` increases immediately while the baselines are still equal to zero. One possible reason for that is that larger groups may be less significant than smaller ones even if they control the multiple testing error. We performed other experiments on greater values of n and we observed the same results for cumulative p-values. We also observed that both $\texttt{SMT}_{cov}$ and `VAL_C_BY` reach the coverage threshold at the same time.

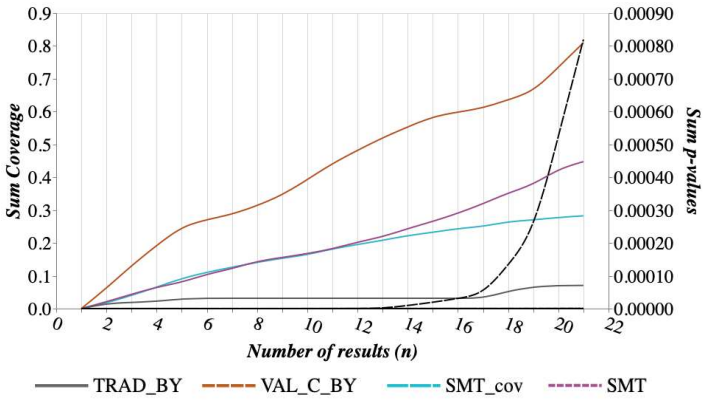


Figure 4.8: Coverage and p-values as a function of the number of results n with $cov_{min} = 0.7$, data samples = 100%.

`VAL_C` **Scalability:** We finally study the evolution of response time as a function of the input data sample (Figure 4.9 - left) and of the number of results n (Figure 4.9 - right). First, it shows that response time increases with the increase of data sample but remains mostly constant with the increase of n. It also shows that $\texttt{SMT}_{cov}$ outperforms `VAL_C_BY` in terms of scalability regardless of the data sample or returned results. In these experiments, we only report results with $cov_{min} = 0.7$. We varied the value of cov_{min} in $\{0.1, 0.3, 0.5, 0.9\}$. The results are mostly similar to the reported ones. In summary, the use of `VAL_C_BY` to solve

ValMin is more appropriate than $\texttt{SMT}_{cov}$. Indeed, even if it has a smaller power significance for small values of n, it succeeds to achieve the coverage constraint while $\texttt{SMT}_{cov}$ fails.

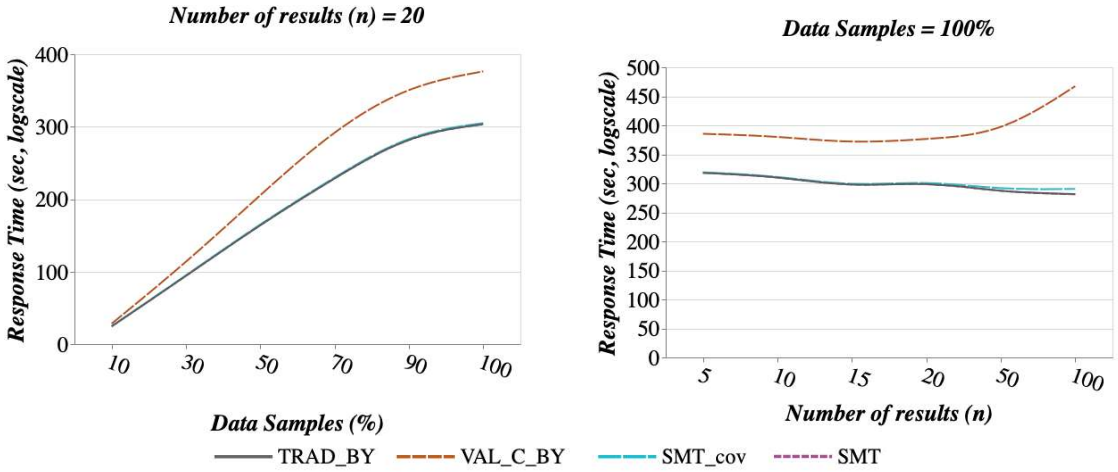


Figure 4.9: Response time as a function of number of data samples **(left)** and number of results n **(right)** with $cov_{min} = 0.7$.

4.7.4 CovMax Results

Study of COVER_α Variants: In this section, we study the significance, coverage, and scalability of the different COVER_α variants by varying the value of the hyperparameter λ. We set TRAD with Bonferroni (TRAD_BN) as a ground truth, and compare the results of COVER_α variants. We consider data samples ranging from 10% to 100% of *Candidates* and report the results for $\texttt{R}_8$.

Figure 4.10 reports power and average FDR. The main observation is that $\texttt{COVER\_}\alpha_{500}$ performs better than all other COVER_α variants with a different λ parameter on both average FDR and power. Small values of λ make COVER_α consume its α-wealth quickly and thus make its discovery power smaller and its average FDR more important.

We now examine simultaneously the evolution of the cumulative coverage and p-values. Results are depicted in Figure 4.11. The graph clearly shows that all COVER_α variants perform similarly and reach full coverage by iteration 49. We also notice that the difference in coverage between the variants is significant in the first iterations but it narrows to the point of being negligible for high iterations. Indeed at the iteration 50, COVER_α_50 covers 99% of data while COVER_α_500 covers 97%. One can also observe that in terms of the sum of p-values, the difference between variants is more significant, and COVER_α_500 outperforms all the other variants.

The last experiment studies the evolution of response time as a function of input data samples (Figure 4.12 - left) and of the number of results n (Figure 4.12 - right). It shows that $\texttt{COVER\_}\alpha_{20}$ is the worst performer as it consumes a large amount of its α-wealth in earlier iterations. It is left with a very small wealth that requires many iterations to reach n results whose p-values quality. The figure shows that the remaining COVER_α variants have practically the same response time with a slight advantage for $\texttt{COVER\_}\alpha_{50}$ and $\texttt{COVER\_}\alpha_{100}$.

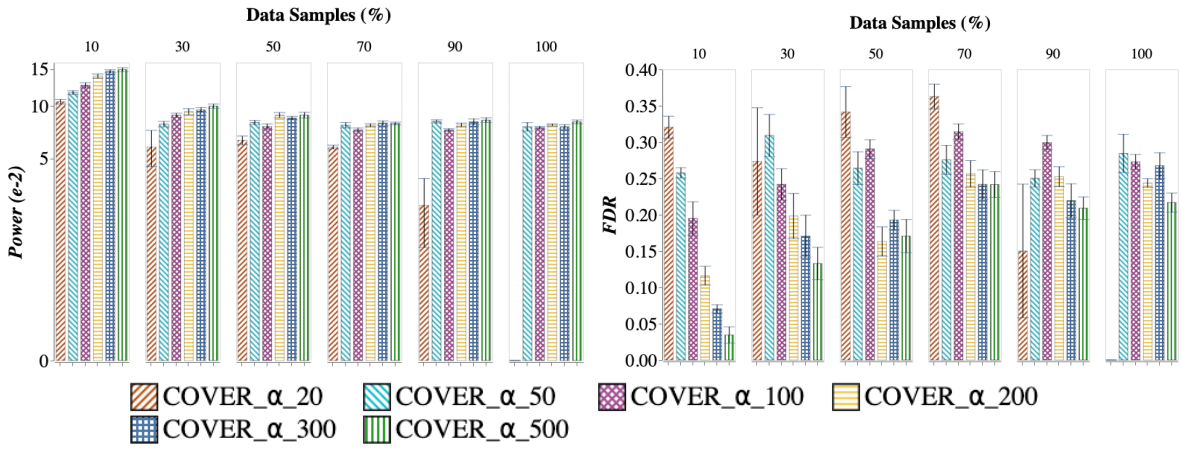


Figure 4.10: Impact of coverage optimization on significance (Power and FDR) with results number $n = 50$ for different percentages of data samples.

This final result shows that an α-investing strategy with a high λ value can attain high coverage while ensuring sound group testing in reasonable times.

`COVER_`α **Hypothesis Significance:** We study the impact of adjustment and coverage on significance. We first study the impact of adjustment on significance using the traditional corrections (`TRAD` and `COVER_G`). We ran all requests in Table 4.1. The complete results are shown in Table 4.6.

The first observation is that in all two-sample tests, the p-value computation protocol (Section 4.4.2) reduces the number of candidates by one order of magnitude. We also observed that both TRAD variants return by far the highest number of results when n is unlimited since they do not set a coverage constraint. Since `TRAD_BY` is less stringent, it consistently yields more results than `TRAD_BN`. This is apparent in the sum of p-values that are significantly higher for `TRAD_BY`. In some cases (R_8), the smallest p-value returned by `COVER_G` is higher than `TRAD` since it optimizes coverage and does not necessarily reach the lowest p-values. The second observation is that when n is capped, `COVER_G` achieves higher coverage than both `TRAD` variants (up to 5x), which leads us to conclude that combining coverage maximization with significance adjustment is necessary for sound group testing.

We now examine how optimizing coverage affects significance. We compare the results of `COVER_`α, `COVER_G` and α_investing policies. Figure 4.13 reports power and FDR for R_8. We observe that `COVER_`α performs better than all other α-investing variants on both power and FDR. The second observation is that `COVER_`α outperforms `COVER_G_BY` regardless of the sample size. Additionally, `COVER_`α attains similar power as `COVER_G_BN`, especially for smaller sample sizes. To better understand the difference between groups returned by `TRAD_BY`, `COVER_G` and `COVER_`α, we analyze the occurrence of attribute-value pairs appearing in the results of R_8. For instance, for $n = 20$, we find that `TRAD_BY` returns groups that cover a total of 11 attribute-value pairs such as gender (male, female), occupation (doctor, scientist, lawyer), age ([25-34], [45-49], [35-44]) and movies of 2 decades (90's, 70's). On the other hand, `COVER_G_BY` and `COVER_`α return groups that cover 24 and 23 attribute-value pairs respectively. Groups contain all gender and age values, more user occupation attributes (6

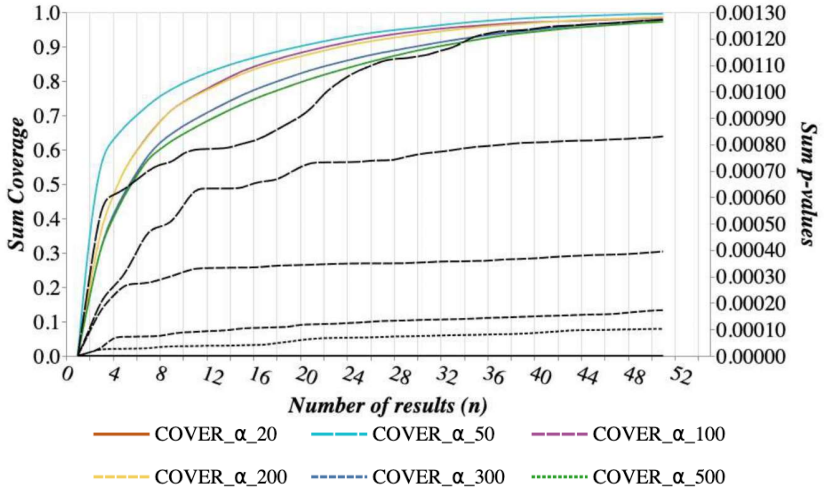


Figure 4.11: Coverage and p-values as a function of the number of results n, data samples = 100%, number of results $n = 50$.

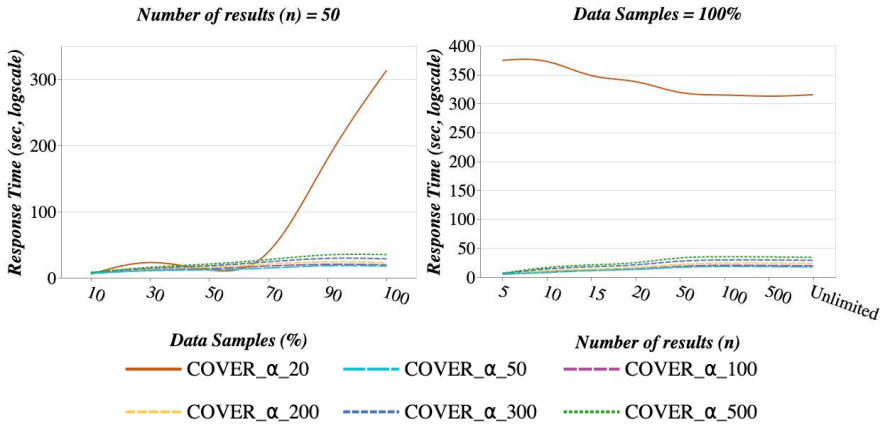


Figure 4.12: Response time as a function of number of data samples **(left)** and number of results n **(right)**.

for `COVER_G_BY` and 5 for `COVER_`α), and all movies from the 50s to the 2000s.

`COVER_`α **Data Coverage:** We now seek to find if our formulation of CovMax hurts the significance of retrieved groups. We compare the results of $\mathtt{R}_8$ using `TRAD_BY`, `COVER_G`, `COVER_`α and all other α-investing policies. We examine simultaneously the evolution of the cumulative coverage and p-values. Results are depicted in Figure 4.14. The graph clearly shows that `COVER_`α performs closely to `COVER_G_BN` and reaches 79% coverage by iteration 20. We see that `COVER_G_BY` is slightly better as it achieves 92% coverage. Additionally, one can observe that `TRAD_BY` and all α-policies cover, at best, only 9% and 30% of the input data respectively. We also notice that β-`Farsighted` invests most of the α-wealth in testing insignificant results and that `COVER_G_BY` is the second worst.

`COVER_`α **Scalability:**

Our last experiment studies the evolution of response time as a function of the number of data

Table 4.6: Results of running all requests in Table 4.1 on `MovieLens'1M` (equal values are shown only once in each cell)

Ri	n	Methods	#allCandidates #Candidates	Benjamini-Yekutieli (BY) / Bonferroni (BN)				
				#Results no-adjustment/BY/BN	Min p-value	Max p-value	Sum p-value	Total Cov
R_1	Unlimited	TRAD	1663 1663	907 / 583 / 337	4.18 e-35	4.23 e-03/ 5.5 e-05	0.3 / 1.91 e-03	1
		COVER_G		3	4.78 e-25	1.9 e-13	1.9 e-13	1
	20	TRAD		20	4.18 e-35	4.43 e-22	6 e-22	0.98
		COVER_G		3	4.78 e-25	1.9 e-13	1.9 e-13	1
R_2	Unlimited	TRAD	1 329 259 108 478	20 505 / 8208 / 1351	7.28 e-22	1.92 e-03 / 2 e-06	3.62 / 6.57 e-04	0.99 / 0.94
		COVER_G		17 / 23	1.14 e-17 / 3.93 e-04	2 e-06 / 3.02 e-07	8.88 e-04 / 4 e-06	0.99 / 0.94
	15	TRAD		15	7.28 e-22	4.08 e-17	1.34 e-16	0.22
		COVER_G		15	1.14 e-17	3.93 e-04 / 2 e-06	8.88 e-04 / 4 e-06	0.99 / 0.92
R_3	Unlimited	TRAD	198 17	6 / 4 / 2	2 e-03	9.09 e-03 / 2 e-03	2.22 e-02 / 4.01 e-03	0.02
		COVER_G		2 / 1	2 e-03	1.8 e-02 / 2 e-03	2 e-02 / 2 e-03	0.12 / 0.02
R_4	Unlimited	TRAD	749 749 44 368	19 472 / 5320 / 616	1.95 e-15	1.26 e-03 / 2.45 e-06	1.80 / 4.45 e-04	1 / 0.96
		COVER_G		7	1.95 e-15	2.93 e-05 / 2.03 e-06	6.93 e-05 / 3.75 e-06	1 / 0.96
	5	TRAD		5	1.95 e-15	1.95 e-15	9.77 e-15	0.17
		COVER_G		5	1.95 e-15	3.93 e-05 / 2.02 e-06	4.78 e-05 / 2.66 e-06	0.97 / 0.92
R_5	Unlimited	TRAD	6 344 6 344	79 / 70 / 39	1.67 e-18	8.39 e-03 / 4.01 e-04	0.11 / 2.14 e-03	0.05 / 0.04
		COVER_G		25 / 15	1.67 e-18	8.39 e-3 / 4.01 e-01	3.71 e-02 / 8.84 e-04	0.05 / 0.04
	10	TRAD		10	1.67 e-18	1.58 e-11	5.14 e-11	0.01
		COVER_G		10	5.3 e-16 / 1.67 e-18	3.54 e-03 / 1.27 e-04	5.16 e-03 / 1.76 e-04	0.04 / 0.03
R_6	Unlimited	TRAD	429 025 42 657	6 076 / 1412 / 1008	0	1.22 e-03 / 5 e-06	0.2 / 4.3 e-05	0.85 / 0.74
		COVER_G		43 / 36	0	8.09 e-04 / 5 e-06	6.25 e-03 / 1.1 e-05	0.85 / 0.74
	20	TRAD		20	0	0	0	0.33
		COVER_G		20	0	8.09 e-04 / 5 e-06	2.85 e-03 / 5 e-06	0.76 / 0.66
R_7	Unlimited	TRAD	85 908 85 908	85 908	0	3.37 e-13	5.14 e-11	1
		COVER_G		1	0	0	0	1
R_8	Unlimited	TRAD	695 772 49 260	26 772 / 16 397 / 4 790	0	2.83 e-03 / 1.86 -06	6.58 / 1.3 e-03	1 / 0.97
		COVER_G		33 / 51	6.21 e-11 / 3.34 e-18	2.44 e-03 / 1.59 e-06	6.4 e-03 / 1.11 e-05	1 / 0.97
	10	TRAD		10	0	2.02 e-22	5.8 e-22	0.21
		COVER_G		10	6.21 e-11 / 6.83 e-14	2.44 e-03 / 1.59 e-06	2.79 e-03 / 3.19 e-06	0.87 / 0.72

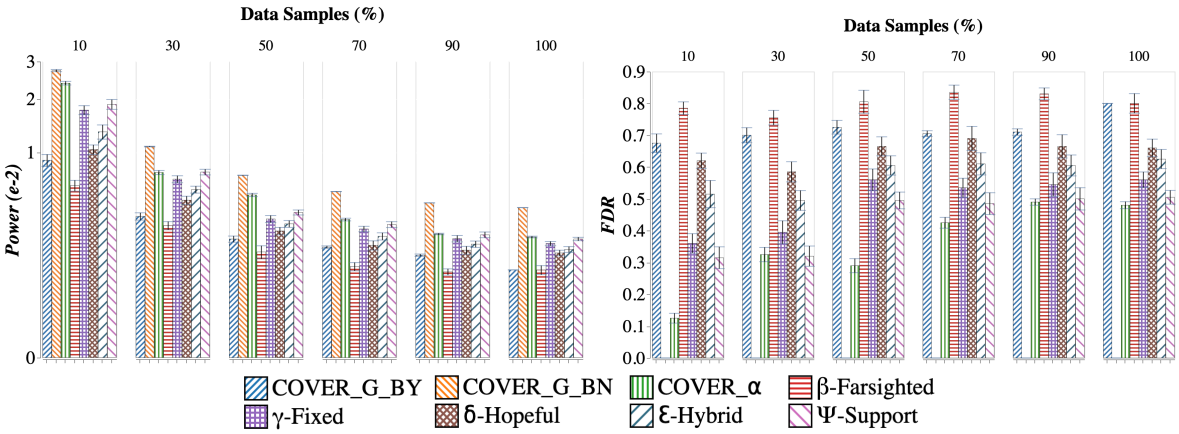


Figure 4.13: Impact of coverage optimization on significance (Power and FDR) with results number $n = 20$ for different percentages of data samples.

samples (Figure 4.15) and the number of results n (Figure 4.16) for R_8. The first observation is that the response time of `COVER_`α remains stable with the increase of both the sample size and the number of results n. Indeed, `COVER_`α computes p-values only for candidates with the highest coverage. Therefore, it clearly outperforms `COVER_G_BN`, `COVER_G_BY` and TRAD_BY. However, `COVER_`α performs worse than the other α-investing algorithms, which is mainly due to `COVER_`α performing at each step a scan over all the remaining candidates to select the one with the highest coverage. But despite, being slightly slower, `COVER_`α performs marginally better in terms of (1) coverage: reaches almost twice the coverage of the best performing α-investing (2) FDR: makes up to 8 times less false discoveries than β-`Farsighted` and up

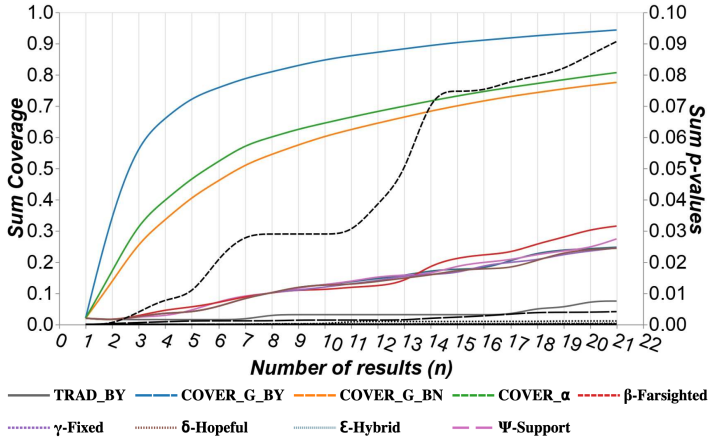


Figure 4.14: Coverage and p-values as a function of the number of results n.

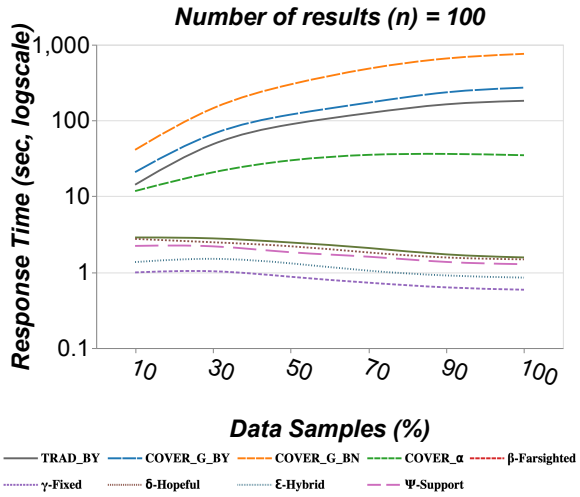


Figure 4.15: Response time as a function of the number of data samples.

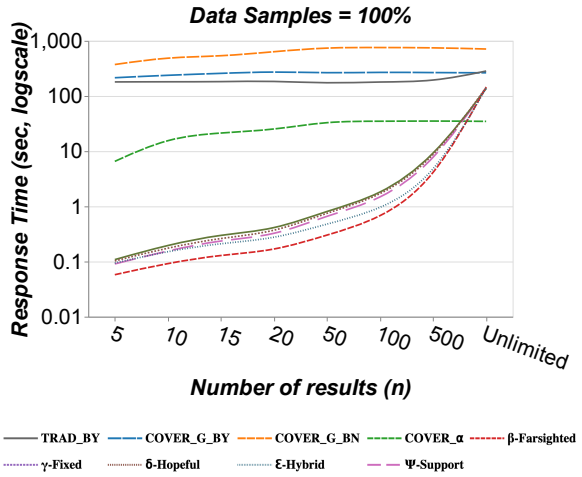


Figure 4.16: Response time as a function of the number of results n.

to 4 times less than ϵ-Hybrid for instance.

Summary on the Remaining Datasets: We summarize the results obtained on the other datasets: Yelp, Tafeng, and BookCrossing.

- **On significance:** For one-sample tests, α_investing policies generally have more "power" but also more "FDR" than our COVER_α. For two-sample tests, COVER_α is the best. Our interpretation is that in one-sample tests the ratio of ground truth to the total number of candidates is usually much higher than in other tests. As α_investing policies return more results than COVER_α, they find more true positives.
- **On coverage:** Results are generally the same for all datasets wherein COVER_α outperforms all α-investing policies. This can be explained by the fact that COVER_α optimizes coverage by design independently of the dataset and type of request.
- **On scalability:** COVER_α runs faster than COVER_G and slightly slower than α-investing

policies on all datasets and for all requests. It becomes clear that the bigger the input dataset, the higher the difference between response times.

In summary, our results confirm that `COVER_`α is the method of choice to attain high coverage of the dataset while ensuring sound group testing at reasonable times.

4.8 Conclusion

In this chapter, we developed GroupTest, a framework that enables data-driven discoveries by combining statistical testing with optimizing data coverage. In that framework, we formalized two generic top-n problems: ValMin and CovMax. The first optimizes significance while setting a constraint on data coverage while the second extends this first problem and aims to maximize data coverage while controlling significance. We showed that these problems are Np-hard and proposed different solutions to solve them. We proposed one greedy solution `VAL_C` to solve ValMin. This solution is based on traditional hypothesis corrections, Bonferroni [30] and Benjamini-Yekutieli [34]. We also proposed two solutions to solve CovMax: a greedy one `COVER_G`, and a heuristic `COVER_`α. While the first relies on traditional corrections, the second leverages α-investing [89]. Our experiments demonstrate the efficiency of these solutions and the necessity of optimizing coverage with significance.

We believe this work to be the first to propose a generic and principled framework for multiple hypothesis testing on large datasets. The framework helps to have general observations about the collective behavior of users, extracting patterns, and understanding the difference between the actions taken by these users. GroupTest also allowed us to verify how different groups are related to subsets of items accommodating different types of tests and comparisons. We illustrated that with two scenarios in the experiments and showed the expressivity of GroupTest.

Chapter 5

Conclusion & Perspectives

In this thesis, we studied individual and collective user behavior based on recommendations and test hypotheses respectively. In this chapter, we conclude the thesis in Section 5.1 and discuss some perspectives and future work in Section 5.2.

5.1 Conclusion

Chapter 2 provided an overview of recommender systems and the metrics that are used to evaluate the quality of recommendations. We discussed the most popular approaches focusing on the ones that were relevant to the work done in this thesis. We summarized the limitations of standard recommenders. In this chapter, we also introduced the Multiple Hypothesis Testing problem and the different procedures used to solve it. We discussed briefly the limitations and challenges of each procedure.

In Chapter 3, we first extended static recommenders by incorporating users' profiles to select the best recommendation approach (Section 3.1). We proposed a meta-learning methodology that considers users' activity statistics and demographics as well as item features and chooses the best recommender for each situation. We applied this methodology to both explicit and implicit data. We demonstrated with our methodology the importance of considering users' states, and their profiles to improve the quality and relevance of recommendations. We showed that our methodology outperformed all standard recommenders. Then, we studied a more realistic context where users' behaviors are constantly changing (Section 3.2). We explored three real-world applications where dynamic recommendations are important: Test recommendations, SQL groupby query recommendations, and diverse sessions recommendations. In Section 3.2.1, we tackled the question of test recommendations in educational systems. We defined three dimensions (expected performance, aptitude, gap) that characterize the learning of users and their evolution. We then defined the AdUp problem for upskilling. We proposed two solutions: `MOO` that directly solves the problem and `MAB` that chooses the dimensions to optimize before solving it. Our experiments showed that optimizing for all three proposed users' dimensions offers a higher upskilling in fewer iterations compared to a state-of-the-art baseline. We also demonstrated that choosing the optimized dimensions dynamically improves learning. In fact, our experiments showed the effectiveness of `MAB` in gaining more skills and achieving mastery quickly. In Section 3.2.2, we tackled the question of recommending visual

analytics based on SQL groupby queries. We defined two generic problems: NPGP and PRP. We proposed DashBot that solves these two problems by incorporating users' feedback. To solve NPGP, we defined three data-driven utility functions: Coverage, variance, and entropy to select the best queries. On the other hand, to solve PRP, we defined two semantics based on Multi-armed bandits. Our experiments showed that our developed tool incorporated well users' feedback while reducing their efforts. In fact, DashBot was able to find all targeted dashboards (targeted panels) independently of their sizes and the number of considered attributes. This shows that our `MAB` solutions offered interesting and relevant analytics. In Section 3.2.3, we tackled the problem of recommending diverse sessions. We defined the problem of selecting the most appropriate attribute that maximizes the diversity of a session. We proposed two solutions: one that generalizes standard diversity-based recommenders and another one based on Reinforcement Learning. In our experiments, we showed that considering a dynamic notion of diversity captures well its maximization than assuming a fixed diversity. In fact, the solutions we proposed outperformed standard diversity-based recommenders for all measures. In addition to that, we showed that using sequence-based approaches (Reinforcement Learning) recommends the finest sessions as they exhibit the best trade-off of relevance and diversity while being more efficient in terms of response time.

Finally, in Chapter 4, we developed GroupTest a solution that enables data-driven discoveries by combining statistical testing and data coverage. This solution allowed us to analyze the collective behavior of users and extract their relations with the different characteristics of sets of items. We formalized two generic top-n problems: ValMin and CovMax. We proposed a greedy solution, `VAL_C`, based on standard multiple hypotheses testing procedures [30, 34] to solve ValMin. To solve CovMax, we proposed two solutions: a greedy one `COVER_G` that is also based on standard multiple hypotheses testing procedures, and a heuristic, `COVER_`α that is based on α-investing [89]. In the experiments of this work, we showed that considering coverage in addition to significance gives more interpretable results. Moreover, we showed that the solutions we proposed are better than the baselines. Our solutions returned groups with higher statistical significance and better representativity of data while making fewer false discoveries. In addition to that, `COVER_`α ensured a finer discovery with better running times.

5.2 Perspectives

In this section, we present future perspectives and potential improvements for this thesis. We outline four different directions: *evaluation*, *optimization*, *concepts*, and *visualization tools*. In the *evaluation* perspectives (Section 5.2.1), we discuss new evaluation protocols that are missed in this thesis. After that, in *optimization* perspectives (Section 5.2.2), we discuss how we directly improve the formalization of the problems we examined in this thesis and discuss possible directions for their solutions. In *concepts* perspectives (Section 5.2.3), we discuss some concepts that can be used to extend our work. Finally, in Section 5.2.4, we discuss new directions in developing *visualization tools* that are directly related to this thesis.

5.2.1 Evaluation Perspectives

In all the works presented in this thesis, we performed extensive experiments using real-world datasets. However, one limitation is that we relied on offline methodologies. For future work,

our experiments can be improved by relying on online protocols. In the following, we present two cases of our work to show how we may evaluate our solutions with real users.

In the test assignment application, we simulated users and their correct answers to evaluate our solutions. In future work, we may deploy our solutions in real systems to test with real learners. One possibility is to use crowdsourcing [118] which represents an established and efficient way of recruiting subjects to complete tasks that are relatively easy for humans. Many online platforms have been developed [1] to enable that. The users have to give their level for the predefined skill before the beginning of the test process. They are then targeted with tests chosen from the set of all tests using our methods for which they have a fixed time to answer. However, relying on crowdsourcing to test with real users may generate new challenges that we may have to address. For example, the main challenge is related to the final financial reward to give to the attendees. In addition to crowdsourcing, we may also deploy our solutions in educational environments at the University Grenoble Alpes like LabNbook [2] or TitrAB [3]. The first one scaffolds students' activity as they learn to write experimental protocols while the second allows one to learn how to develop the protocol for acid-base titrations through 16 tests with 4 difficulty levels. Experimenting with real learners will help to verify the findings we exhibit, and test learners' performances after attaining mastery. To achieve that, a new measure, *score* [102], is leveraged which computes the precision of correct answers after attaining mastery of each learner. This will extend our previous experiments where we reported only pre-mastery results. Using this measure will allow one to study the quality of post-mastery answers.

In the SQL groupby query recommendations, we already developed a system for dashboard generation based on DashBot [69] that real users can interact with to search interesting panels. One may rely on this visualization tool to conduct a user study. It consists of two different parts. In the first part, users interact with DashBot using only one semantic (See Section 3.2.2) and the system records their click efforts, wait time, and the number of iterations to provide accepted panels. The users may also be asked about the quality of the recommended panels. In the second part, we aim to compare all semantics presented in this thesis. Users are targeted at each iteration with two different panels for which they are asked to choose the best one. Users that are part of the study may also give feedback about their experience when using the system. This study will help us better understand the strengths and limitations of our proposed solution.

5.2.2 Optimization Perspectives

The second future direction is to improve the optimization of our problems by either incorporating new objectives or capturing additional data from users. In both cases, we discuss potential solutions that incorporate these extensions.

Optimizing Additional Objectives. One way to improve our solutions is to enhance the formalization of our problems. The test assignment problem can be extended to consider multiple skills instead of one skill and leverage course materials in addition to the tests. The challenge of considering multiple skills is understanding their dependencies and choosing the

[1] http://www.mturk.com, https://www.foulefactory.com, https://www.crowdflower.com
[2] https://labnbook.fr/en
[3] http://titrab.imag.fr/

best one at each iteration that yields the highest overall learning gain. Different strategies to choose that skill can be used. One would be alternating between the skills or choosing the one for which the learner has the lowest level. Different skills can be also combined in one iteration. For example, the work by [164] considers many learners' skills. The authors propose to model a user profile based on the expected performance of all skills. A k-means is then applied using this profile to select the overall ability of learners and recommend tests. For course materials, the work proposed by [27] aims to learn policies that model the relationships between course activities, learner actions, and educational outcomes. They propose a solution that generates a sequence of courses that aim to minimize the effort of teachers and maximize the learning gain of students. However, they recommend only one test which denotes the end of the course sequence. Another work [172] proposes to learn a reinforcement learning policy that generates course planning for students by incorporating their requirements as constraints. They also assume two types of courses: core and elective and define their dependencies. However, in this work, tests are not considered. In future work, the problem formalization of AdUp can be extended to integrate the recommendation of courses in addition to tests. By doing that, some domain-related rules that define the complex relationship between these materials need to be formalized. The developed solutions aim to produce the best heterogeneous sequences by alternating courses and tests and maximize the skill gain of learners.

In the SQL query problem, we aim to minimize user effort to find interesting panels. We formalized the effort as the number of iterations (or the number of clicks [4]). However, our definition of effort is partial. In fact, we ignore the time users spend thinking about the recommended panel. Intuitively, we can say that if a panel is highly relevant/irrelevant, the user accepts/rejects it immediately. Otherwise, the slowness of the answer reflects the hesitation of the user. Hence, users' response time gives implicit feedback about the effectiveness of the recommendation and their preferences. For this reason, we can extend the user effort formalization to integrate response time. As our solution is based on Multi-armed bandits, the user effort is captured by the reward (See Section 3.2.2). We then may also extend our reward and integrate response time to represent the strength of the taken decision. One possible solution is to add the response time as a weight to the actual reward. By doing that, the arms that give interesting panels in terms of content and time will have high values of reward. Otherwise, arms that give negative panels that were quickly rejected will have small values of reward.

We can also improve the diverse sessions recommendations by incorporating different types of diversity. In fact, as stated in [140], many diversity measures were proposed (e.g., Intra-list, entropy, gini coefficient, etc.), and the need for a system that generalizes them is essential. One of our future works is to offer a solution to that problem. We consider each diversity measure as a different objective independent from the others and aim to maximize them. The same idea of `SMORL` [233] can be used to integrate these objectives. For each diversity objective, a Q-function is learned and combined them using a scalarization function (See RL regularize is Section 3.2.3). Another possibility is to define a dominance relation between candidate sessions based on the diversity measures and leverage a multi-objective optimization of reward as proposed in [177]. Then, an agent is trained with this reward. It will learn how to balance all these measures and recommends the session that offers the best trade-off.

[4] Users perform either one click or two clicks in each iteration. So the number of clicks is positively correlated to the number of iterations. Optimizing for the first is equivalent to optimizing for the second.

Another extension that would benefit this work is to optimize in addition to the intra-diversity of sessions the inter-diversity between them. For example, the work proposed by [87] defines four bi-objective optimization problems where intra/inter diversity may be either minimized or maximized (max intra-max inter, max intra-min inter, min intra-max inter, min intra-min inter) but their solutions are static as they assume that diversity attributes are given as input. In future work, the integration of inter-diversity is done through our reinforcement learning solution. This is challenging as a new intra-diversity measure has to be defined while adapting the reward function of the agent. An intuitive solution is to train an RL agent to make all these decisions and takes a tuple of actions: (intra-attribute, inter-attribute, set of items) where intra-attributes and inter-attributes are both maximized attributes and the set of items defines the recommended session. One may agree that this is still challenging as we need to design a powerful reward that captures all these dimensions. An alternative solution exists where the agent is trained to guide the recommendation. In fact, the agent decides for both attributes, and the set of items is either generated by the static bi-objective solutions proposed in [87] or by adapting the standard `MMR` approach [52].

Incorporating Additional User Data. Another way to improve our solutions is to incorporate additional data about users. This data may help to better capture the interests and preferences of users as it improves the design of their profiles and states. In the meta-learning methodology, we may extend our work to consider heterogeneous data, e.g., leveraging both implicit and explicit data of the user. This question was previously studied in [147, 179]. For example, the meta-learning methodology may use both ratings and clicks of the users when they provide them. In addition, our methodology can be extended using multiple rating dimensions. For example, in restaurant recommendations, our methodology may leverage food, atmosphere, service, and price ratings. We may use multi-criteria recommenders [3] to handle this data which is an important concern in subjective databases [151]. Another solution is to leverage textual reviews to either generate missing ratings or extend users' profiles. With the progress made these last years in natural language processing and the emergence of large language models [48, 56], one may use them to generate users' embeddings, infer their sentiments, or extract tags and important words from the reviews. These dimension extensions are likely to help reduce data sparsity, enhance the users' profiles and increase performance.

In the SQL query recommendation, one limitation of our solution is the restricted feedback the users can make when targeted with a panel. In fact, we assume Boolean feedback (Yes/No) with a predefined and optional set of reasons only available for the No feedback. This feedback may be enriched by capturing users' reasons for accepting a panel. This additional feedback can help to gain insights about users' preferences and make more informed recommendations. Additionally, one might find these reasons too restrictive or generic. Hence, with the progress made in the fields of natural language processing (NLP) [250], we may extend our system by allowing users to express explicitly their reasons with their own words and use these NLP techniques to interpret their intent and extract the exact reasons to refine the queries. Another limitation is that the predefined reasons are all related to the groupby and aggregation attributes in the query, whereas a rejection or an acceptance can be related to the type of visualization. For example, in the case where the proposed analytic is displayed with a chart bar but the user prefers it with a circular chart, our system changes the query instead of the display. Hence, user feedback can be extended by capturing visualization-based reasons. An approach that exploits this feedback to map the queries to the most appropriate visualization panel can be used [119].

In diverse sessions recommendations, we proposed a solution that captures only the best attribute which yields the highest diversity within sessions. Relying on one attribute may capture partially the diverse preferences of users. For example, in the context of music recommendations, a user may aim to maximize the diversity of two attributes (e.g., both genre and era in iteration 1 of the motivating example in Section 3.2.3. By selecting just one attribute (e.g., genre), we may not maximize the diversity of the other one. Thus, the direct extension of this work is to capture and combine multiple attributes within the same session. The main challenge is related to the number of considered attributes. An intuitive solution is to choose at each iteration the k best attributes that maximize diversity. Another possible solution is to consider all attributes in each session and learn their weights for aggregation. A similar but static solution was proposed in [77] where users are clustered into fixed classes for all attributes. A user may belong to different classes for different attributes. Each class has a fixed and predefined weight. The diversity is then the result of the weighted aggregation of the diversities of attributes.

Finally, in all the applications, contextual factors [5] that affect the users may be integrated. We may select geographic and temporal information as they are easy to collect compared to other contextual factors such as mood and social [5]. Their integration in the users' profiles [109] may enhance the performance of the solutions. In fact, relying on the geographical contexts showed improvement in different recommendation fields [11]. On the other hand, we can also extend our solutions by integrating explicitly temporal information in the states using time-binning [136].

5.2.3 New Concepts Perspectives

Our third area of future work is related to incorporating new and wider concepts that would require deeper investigation. This would enhance our solutions by capturing new theories and exploring new research areas.

Supporting New Learning Theories. Our work about test assignment may benefit from incorporating other learning theories in addition to mastery learning [188] and ZPD [249]. In fact, there are many learning theories in the physical world, such as situated learning [142] and collaborative learning [49]. Collaborative learning is effective in online learning environments like MOOCs, and studies showed that rich interactions such as peer feedback and discussion promote learning [73, 59, 258]. In this theory, groups of two or more learners work together to solve problems and complete tasks. For example, one work [158] showed through crowdsourcing that learners tend to have better learning when they are aware of the answers of other learners. Situated learning, on the other hand, focuses on the relationship between learning and the social situation in which it takes place. It can be represented as an apprenticeship where knowledge is propagated from experts to novice learners based on the principle of Legitimate Peripheral Participation [142]. If novice learners can directly observe the practices of experts, they understand in which their own efforts fit.

Supporting New Queries. Our SQL recommendation solution, DashBot, may also benefit from the extension of considered queries. We may do so by including the comparison predicate `WHERE` to have a condition on considered rows and specify which ones to retrieve in the returned results. To permit this, the panel representation as well as the inclusion and coverage concepts presented in Section 3.2.2 have to be enriched. The solution for new panel generation has to

be enhanced by defining utility functions to choose the attributes on which the condition is going to be applied. The panel refinement procedures also need to be extended by integrating new acceptance/rejection reasons for `WHERE` attributes and considering more arms in the `MAB` strategies.

Supporting New Group Dimensions. In the collective behavior analysis, our work leverages significance and coverage. It may benefit from different dimensions that define the quality of the generated demographic groups. Exploring other dimensions is one of the future directions. For example, diversity can be used to return user groups that are different from each other in order to provide complementary information. Diversity may be computed based on the description of the demographic groups. Another dimension that can be enhanced is novelty to return at each time, demographic groups that were not seen previously by the user. Other dimensions are discussed in [97] like conciseness (concise demographic groups) or peculiarity (distance-based demographic groups).

5.2.4 Visualization Tools Perspectives

Our last direction is related to developing new visualization tools. In collective user analysis, our future work is to provide an interactive tool in which users can experiment with different data using different requests and compare results in a sound manner. In this tool, users may have the ability to navigate among many data regions to find interesting demographic groups. In addition to that, the users may also have the possibility to choose the type of hypothesis to explore (One-sample, Two-samples, Multiple-samples), the aggregation dimension that defines the behavior of users (e.g., ratings, clicks), and the aggregation function (mean, variance, distribution). This would require developing visual analytics and combining them with our solutions. These visualizations may describe the input groups the user chose, all significant subgroups with their coverage and statistical significance. One reason to use these visualizations is because they display hypothesis testing results in an interpretable fashion. When using this tool, users may dive into different groups and verify some hypotheses or explore other groups in different regions of data. By doing so, users create navigation pipelines of demographic groups which this tool will store and display in a dynamic way.

As users generate navigation pipelines in an iterative fashion, we may also design a solution that mimics their actions and generates automatically this pipeline. This solution may be integrated into our visualization tool. As this problem is iterative, reinforcement learning appears to be a genuine solution. The agent chooses at each iteration which input demographic group to consider. It also chooses to whether exploit this group and dive into it to analyze the behavior of its subgroups or to explore other regions of the data and jump to unrelated demographic groups. The agent may rely on the multiple hypothesis testing solutions that we proposed in this thesis to return the most significant and representative groups. The agent has also to address the challenge of choosing between the different types of hypotheses, aggregation dimensions, and functions. To develop this type of agent, a reward function that captures the significance and coverage of the returned demographic groups has to be designed. The reward should also capture the case where exploration is needed and the agent chose exploitation and inversely. Since an agent is trained to generate that pipeline, the visualization tool offers two navigation modes: manually as described before where users make their decisions to generate the pipeline, fully automatic where the trained agent generates the pipeline entirely.

Bibliography

[1] Solmaz Abdi, Hassan Khosravi, Shazia Sadiq, and Ali Darvishi. Open learner models for multi-activity educational systems. In *Artificial Intelligence in Education: 22nd International Conference, AIED 2021, Utrecht, The Netherlands, June 14–18, 2021, Proceedings, Part II*, pages 11–17. Springer, 2021.

[2] Himan Abdollahpouri, Zahra Nazari, Alex Gain, Clay Gibson, Maria Dimakopoulou, Jesse Anderton, Benjamin Carterette, Mounia Lalmas, and Tony Jebara. Calibrated recommendations as a minimum-cost flow problem. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 571–579, 2023.

[3] Gediminas Adomavicius, Nikos Manouselis, and YoungOk Kwon. Multi-criteria recommender systems. In *Recommender systems handbook*, pages 769–803. Springer, 2010.

[4] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[5] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2010.

[6] Gediminas Adomavicius and Jingjing Zhang. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems (TMIS)*, 3(1):1–17, 2012.

[7] M Mehdi Afsar, Trafford Crump, and Behrouz Far. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys*, 55(7):1–38, 2022.

[8] Alexander A Ageev and Maxim I Sviridenko. Approximation algorithms for maximum coverage and max cut with given sizes of parts. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 17–30. Springer, 1999.

[9] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[10] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[11] Marie Al-Ghossein. *Context-aware recommender systems for real-world applications.* PhD thesis, Université Paris-Saclay (ComUE), 2019.

[12] Xavier Amatriain, Alejandro Jaimes*, Nuria Oliver, and Josep M Pujol. Data mining methods for recommender systems. In *Recommender systems handbook*, pages 39–71. Springer, 2010.

[13] Xavier Amatriain, Alejandro Jaimes, Nuria Oliver, and Josep M Pujol. Data mining methods for recommender systems. In *Recommender systems handbook*, pages 39–71. Springer, 2011.

[14] S. Amer-Yahia, T. Milo, and B. Youngmann. Exploring Ratings in Subjective Databases. In *SIGMOD*, pages 62–75, 2021.

[15] Sihem Amer-Yahia, Sofia Kleisarchaki, Naresh Kumar Kolloju, Laks VS Lakshmanan, and Ruben H Zamar. Exploring rated datasets with rating maps. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1411–1419. International World Wide Web Conferences Steering Committee, 2017.

[16] Rohan Anand and Joeran Beel. Auto-surprise: An automated recommender-system (autorecsys) library with tree of parzens estimator (tpe) optimization. In *Fourteenth ACM Conference on Recommender Systems*, pages 585–587, 2020.

[17] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. Algorithmic effects on the diversity of consumption on spotify. In *Proceedings of The Web Conference 2020*, pages 2155–2165, 2020.

[18] Elham Asani, Hamed Vahdat-Nejad, and Javad Sadri. Restaurant recommender system based on sentiment analysis. *Machine Learning with Applications*, 6:100114, 2021.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning*, 47(2-3):235–256, 2002.

[20] Enkh-Amgalan Baatarjav, Santi Phithakkitnukoon, and Ram Dantu. Group recommendation system for facebook. In *On the Move to Meaningful Internet Systems: OTM 2008 Workshops: OTM Confederated International Workshops and Posters, ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent+ QSI, ORM, PerSys, RDDS, SEMELS, and SWWS 2008, Monterrey, Mexico, November 9-14, 2008. Proceedings*, pages 211–219. Springer, 2008.

[21] Anirudhan Badrinath, Frederic Wang, and Zachary Pardos. pybkt: An accessible python library of bayesian knowledge tracing models. In *Proceedings of the 14th International Conference on Educational Data Mining*, pages 468–474, 2021.

[22] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[23] O. Bar El, T. Milo, and A. Somech. Automatically Generating Data Exploration Sessions Using Deep Reinforcement Learning. In *SIGMOD*, pages 1527–1537, 2020.

[24] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. Efficient sort-based skyline evaluation. *ACM Transactions on Database Systems (TODS)*, 33(4):1–49, 2008.

[25] Ashok R Basawapatna, Alexander Repenning, Kyu Han Koh, and Hilarie Nickerson. The zones of proximal flow: guiding students through a space of computational thinking

skills and challenges. In *Proceedings of the ninth annual international ACM conference on International computing education research*, pages 67–74, 2013.

[26] Pierpaolo Basile, Marco De Gemmis, Anna Lisa Gentile, Leo Iaquinta, Pasquale Lops, and Giovanni Semeraro. An electronic performance support system based on a hybrid content-collaborative recommender system. *Neural Network World*, 17(6):529, 2007.

[27] Jonathan Bassen, Bharathan Balaji, Michael Schaarschmidt, Candace Thille, Jay Painter, Dawn Zimmaro, Alex Games, Ethan Fast, and John C Mitchell. Reinforcement learning for the adaptive scheduling of educational activities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2020.

[28] Joeran Beel, Corinna Breitinger, Stefan Langer, Andreas Lommatzsch, and Bela Gipp. Towards reproducibility in recommender-systems research. *User modeling and user-adapted interaction*, 26(1):69–101, 2016.

[29] Joeran Beel, Alan Griffin, and Conor O'Shea. Darwin & goliath: a white-label recommender-system as-a-service with automated algorithm-selection. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 534–535, 2019.

[30] Gleb Beliakov, Simon James, Juliana Mordelová, Tatiana Rückschlossová, and Ronald R. Yager. Generalized bonferroni mean operators in multi-criteria aggregation. *Fuzzy Sets Syst.*, 161(17):2227–2242, 2010.

[31] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *icdm*, volume 7, pages 43–52. Citeseer, 2007.

[32] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *KDD Exploration*, 2008.

[33] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

[34] Yoav Benjamini and Daniel Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Annals of Statistics*, 29:1165–1188, 2001.

[35] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, 2007.

[36] Idir Benouaret and Sihem Amer-Yahia. A comparative evaluation of top-n recommendation algorithms: Case study with total customers. In *Proceedings of the twenty-first international conference on Big Data*, 2020.

[37] bigdata ustc. Educdm. `https://github.com/bigdata-ustc/EduCDM`, 2021.

[38] Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10:147–180, 2000.

[39] Allan Birnbaum. Some latent trait models and their use in inferring an examinee's ability. *Statistical theories of mental test scores*, 1968.

[40] Mario Boley, Michael Mampaey, Bo Kang, Pavel Tokmakov, and Stefan Wrobel. One click mining: Interactive local pattern discovery through implicit preference and per-

formance learning. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, pages 27–35. ACM, 2013.

[41] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20, 2007.

[42] Nassim Bouarour, Idir Benouaret, and Sihem Amer-Yahia. How useful is meta-recommendation? an empirical investigation. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 600–606. IEEE, 2021.

[43] Nassim Bouarour, Idir Benouaret, and Sihem Amer-Yahia. Learning diversity attributes in multi-session recommendations. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 465–474. IEEE, 2022.

[44] Nassim Bouarour, Idir Benouaret, and Sihem Amer-Yahia. Optimizing data coverage and significance in multiple hypothesis testing on user groups. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems LI: Special Issue on Data Management-Principles, Technologies and Applications*, pages 64–96. Springer, 2022.

[45] Nassim Bouarour, Idir Benouaret, and Sihem Amer-Yahia. Significance and coverage in group testing on the social web. In *Proceedings of the ACM Web Conference 2022*, pages 3052–3060, 2022.

[46] Nassim Bouarour, Idir Benouaret, Cédric D'Ham, and Sihem Amer-Yahia. Adaptive test recommendation for mastery learning. In *Proceedings of the 2nd International Workshop on Data Systems Education: Bridging Education Practice with Education Research*, DataEd '23, page 18–23, New York, NY, USA, 2023. Association for Computing Machinery.

[47] Coen Bron and Joep Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Communications of the ACM*, 16(9):575–577, 1973.

[48] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[49] Kenneth A Bruffee. *Collaborative learning: Higher education, interdependence, and the authority of knowledge*. ERIC, 1999.

[50] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[51] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

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

[53] Pablo Castells, Neil Hurley, and Saul Vargas. Novelty and diversity in recommender systems. In *Recommender systems handbook*, pages 603–646. Springer, 2021.

[54] Hao Cen, Kenneth Koedinger, and Brian Junker. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *Intelligent Tutoring Systems: 8th International Conference, ITS 2006, Jhongli, Taiwan, June 26-30, 2006. Proceedings 8*, pages 164–175. Springer, 2006.

[55] Chandra Chekuri, Kent Quanrud, and Zhao Zhang. On approximating partial set cover and generalizations. *arXiv preprint arXiv:1907.04413*, 2019.

[56] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[57] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666, 2008.

[58] Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, and Matthew Sartin. Combing content-based and collaborative filters in an online newspaper. In *Proc. of Workshop on Recommender Systems-Implementation and Evaluation*, 1999.

[59] Derrick Coetzee, Seongtaek Lim, Armando Fox, Bjorn Hartmann, and Marti A Hearst. Structuring interactions for large-scale synchronous peer learning. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pages 1139–1152, 2015.

[60] Andrew Collins and Joeran Beel. A first analysis of meta-learned per-instance algorithm selection in scholarly recommender systems. In *Workshop on Recommendation in Complex Scenarios, 13th ACM Conference on Recommender Systems (RecSys)*, 2019.

[61] Andrew Collins, Dominika Tkaczyk, and Joeran Beel. A novel approach to recommendation algorithm selection using meta-learning. In *AICS*, pages 210–219, 2018.

[62] David Colquhoun. An investigation of the false discovery rate and the misinterpretation of p-values. *Royal Society Open Science*, 1(3):140216, 2014.

[63] Michelle Keim Condliff, David D Lewis, David Madigan, and Christian Posse. Bayesian mixed-effects models for recommender systems. In *ACM SIGIR*, volume 99, pages 23–30. Citeseer, 1999.

[64] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[65] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, 2010.

[66] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8(12):2024–2027, 2015.

[67] Tiago Cunha, Carlos Soares, and André CPLF de Carvalho. Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering. *Information Sciences*, 423:128–144, 2018.

[68] S. Da Col, R. Ciucanu, M. Soare, N. Bouarour, and S. Amer-Yahia. DashBot: An ML-Guided Dashboard Generation System. In *CIKM*, 2021. Accepted, to appear.

[69] Sandrine Da Col, Radu Ciucanu, Marta Soare, Nassim Bouarour, and Sihem Amer-Yahia. Dashbot: An ml-guided dashboard generation system. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4696–4700, 2021.

[70] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.

[71] Sriharsha Dara, C Ravindranath Chowdary, and Chintoo Kumar. A survey on group recommender systems. *Journal of Intelligent Information Systems*, 54(2):271–295, 2020.

[72] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296, 2010.

[73] Dan Davis, Guanliang Chen, Claudia Hauff, and Geert-Jan Houben. Activating learning at scale: A review of innovations in online learning strategies. *Computers & Education*, 125:327–344, 2018.

[74] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

[75] Giovanni Di Leo and Francesco Sardanelli. Statistical significance: p value, 0.05 threshold, and applications to radiomics—reasons for a conservative approach. *European radiology experimental*, 4(1):1–8, 2020.

[76] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. An analysis of users' propensity toward diversity in recommendations. In *Proceedings of the 8th ACM Conference on Recommender Systems*, pages 285–288, 2014.

[77] Tommaso Di Noia, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. Adaptive multi-attribute diversity for recommender systems. *Information Sciences*, 382:234–253, 2017.

[78] Jörg Diederich and Tereza Iofciu. Finding communities of practice from user profiles based on folksonomies. In *Innovative approaches for learning and knowledge sharing, ec-tel workshop proc*, pages 288–297. Citeseer, 2006.

[79] K. Dimitriadou, O. Papaemmanouil, and Y. Diao. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. *TKDE*, 28(11):2842–2856, 2016.

[80] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856, 2016.

[81] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492, 2005.

[82] Zhenhua Dong, Zhe Wang, Jun Xu, Ruiming Tang, and Jirong Wen. A brief history of recommender systems. *arXiv preprint arXiv:2209.01860*, 2022.

[83] Gintare Karolina Dziugaite and Daniel M Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

[84] Michael Ekstrand and John Riedl. When recommenders fail: predicting recommender failure for algorithm selection and combination. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 233–236, 2012.

[85] Michael D Ekstrand and Maria Soledad Pera. The demographics of cool. *Poster Proceedings at ACM RecSys. ACM, Como, Italy*, 2017.

[86] Frank Emmert-Streib and Matthias Dehmer. Understanding statistical hypothesis testing: The logic of statistical inference. *Machine Learning and Knowledge Extraction*, 1(3):945–962, 2019.

[87] Mohammadreza Esfandiari, Ria Mae Borromeo, Sepideh Nikookar, Paras Sakharkar, Sihem Amer-Yahia, and Senjuti Basu Roy. Multi-session diversity to improve user satisfaction in web applications. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 1928–1936, 2021.

[88] Maryam Etemadi, Sepideh Bazzaz Abkenar, Ahmad Ahmadzadeh, Mostafa Haghi Kashani, Parvaneh Asghari, Mohammad Akbari, and Ebrahim Mahdipour. A systematic review of healthcare recommender systems: Open issues, challenges, and techniques. *Expert Systems with Applications*, page 118823, 2022.

[89] D. Foster and R. A. Stine. Alpha-investing: A procedure for sequential control of expected false discoveries. In *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, 2008.

[90] Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. Top-k bounded diversification. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 421–432, 2012.

[91] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[92] Alejandro G. Martín, Alberto Fernández-Isabel, Isaac Martín de Diego, and Marta Beltrán. A survey for user behavior analysis based on machine learning techniques: current models and applications. *Applied Intelligence*, 51(8):6029–6055, 2021.

[93] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommenda-

tions. In *2010 IEEE International Conference on Data Mining*, pages 176–185. IEEE, 2010.

[94] Diego Garcıa-Saiz and Marta Zorrilla. Metalearning-based recommenders: towards automatic classification algorithm selection. In *Conferencia de la Asociación Espanola para la Inteligencia Artificial*, pages 749–758, 2015.

[95] Dragan Gašević, Vitomir Kovanović, Srećko Joksimović, and George Siemens. Where is research on massive open online courses headed? a data analysis of the mooc research initiative. *International Review of Research in Open and Distributed Learning*, 15(5):134–176, 2014.

[96] Yingqiang Ge, Shuya Zhao, Honglu Zhou, Changhua Pei, Fei Sun, Wenwu Ou, and Yongfeng Zhang. Understanding echo chambers in e-commerce recommender systems. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 2261–2270, 2020.

[97] Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006.

[98] Nicolas Gillis. The why and how of nonnegative matrix factorization. *Regularization, optimization, kernels, and support vector machines*, 12(257):257–291, 2014.

[99] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[100] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

[101] Yue Gong, Joseph E Beck, and Neil T Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting. *ITS2010 Intelligent Tutoring Systems. LNCS*, 6094:35–44, 2010.

[102] José P González-Brenes and Yun Huang. " your model is predictive–but is it useful?" theoretical and empirical considerations of a new paradigm for adaptive tutoring evaluation. *International Educational Data Mining Society*, 2015.

[103] Amit Goyal, Francesco Bonchi, and Laks VS Lakshmanan. Discovering leaders from community actions. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 499–508. ACM, 2008.

[104] Asela Gunawardana and Guy Shani. Evaluating recommender systems. In *Recommender systems handbook*, pages 265–308. Springer, 2015.

[105] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[106] Yue Guo, Carsten Binnig, and Tim Kraska. What you see is not what you get! detecting simpson's paradoxes during data exploration. In *Proceedings of the 2nd Workshop on Human-In-the-Loop Data Analytics*, pages 1–5, 2017.

[107] Wilhelmiina Hämäläinen and Geoffrey I. Webb. A tutorial on statistically sound pattern discovery. *Data Min. Knowl. Discov.*, 33(2):325–377, 2019.

[108] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206, 2010.

[109] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. Contextual and sequential user embeddings for large-scale music recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 53–62, 2020.

[110] Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. Shifting consumption towards diverse content on music streaming platforms. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 238–246, 2021.

[111] F. M. Harper and J. A. Konstan. The MovieLens Datasets: History and Context. *ACM TiiS*, 5(4):19:1–19:19, 2016.

[112] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912*, 2018.

[113] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[114] Markus Hegland. The apriori algorithm–a tutorial. *Mathematics and computation in imaging science and information processing*, pages 209–262, 2007.

[115] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 843–852, 2018.

[116] Dorit S Hochbaum and Anu Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

[117] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70, 1979.

[118] Jeff Howe et al. The rise of crowdsourcing. *Wired magazine*, 14(6):176–183, 2006.

[119] Kevin Hu, Michiel A Bakker, Stephen Li, Tim Kraska, and César Hidalgo. Vizml: A machine learning approach to visualization recommendation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[120] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

[121] Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications*, 28:94–101, 2018.

[122] Sheena S Iyengar and Mark R Lepper. When choice is demotivating: Can one desire too much of a good thing? *Journal of personality and social psychology*, 79(6):995, 2000.

[123] Mohieddin Jafari and Naser Ansari-Pour. Why, when and how to adjust your p values? *Cell Journal (Yakhteh)*, 20(4):604, 2019.

[124] Dietmar Jannach and Michael Jugovac. Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems (TMIS)*, 10(4):1–23, 2019.

[125] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[126] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. On offline evaluation of recommender systems. *arXiv preprint arXiv:2010.11060*, 2020.

[127] Xin Jia, Wenjie Zhou, Xu Sun, and Yunfang Wu. Eqg-race: Examination-type question generation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 13143–13151, 2021.

[128] Dawei Jiang, Qingchao Cai, Gang Chen, HV Jagadish, Beng Chin Ooi, Kian-Lee Tan, and Anthony KH Tung. Cohort query processing. *Proceedings of the VLDB Endowment*, 10(1):1–12, 2016.

[129] Niranjan Kamat, Prasanth Jayachandran, Karthik Tunga, and Arnab Nandi. Distributed and interactive cube exploration. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 472–483. IEEE, 2014.

[130] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[131] Kim Kelly, Yan Wang, Tamisha Thompson, and Neil Heffernan. Defining mastery: Knowledge tracing versus n-consecutive correct responses. *Student Modeling From Different Aspects*, page 39, 2016.

[132] Hassan Khosravi, Kirsty Kitto, and Joseph Jay Williams. Ripple: A crowdsourced adaptive platform for recommendation of learning activities. *arXiv preprint arXiv:1910.05522*, 2019.

[133] Hassan Khosravi, Shazia Sadiq, and Dragan Gasevic. Development and adoption of an adaptive learning system: Reflections and lessons learned. In *Proceedings of the 51st ACM technical symposium on computer science education*, pages 58–64, 2020.

[134] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[135] Joseph A Konstan, Sean M McNee, Cai-Nicolas Ziegler, Roberto Torres, Nishikant Kapoor, and John Riedl. Lessons on applying automated recommender systems to information-seeking tasks. In *AAAI*, volume 6, pages 1630–1633, 2006.

[136] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.

[137] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender systems handbook*, pages 77–118. Springer, 2015.

[138] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[139] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[140] Matevž Kunaver and Tomaž Požrl. Diversity in recommender systems–a survey. *Knowledge-based systems*, 123:154–162, 2017.

[141] Miklós Kurucz, András A Benczúr, and Károly Csalogány. Methods for large scale svd with missing values. In *Proceedings of KDD cup and workshop*, volume 12, pages 31–38. Citeseer, 2007.

[142] Jean Lave and Etienne Wenger. *Situated learning: Legitimate peripheral participation*. Cambridge university press, 1991.

[143] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[144] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol Natsev. Collaborative deep metric learning for video understanding. In *Proceedings of the 24th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 481–490, 2018.

[145] Yong-Won Lee and Yasuyo Sawaki. Cognitive diagnosis approaches to language assessment: An overview. *Language Assessment Quarterly*, 6(3):172–189, 2009.

[146] Erich Leo Lehmann, Joseph P Romano, and George Casella. *Testing statistical hypotheses*, volume 3. Springer, 2005.

[147] Lukas Lerche and Dietmar Jannach. Using graded implicit feedback for bayesian personalized ranking. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 353–356, 2014.

[148] Lei Li, Li Zheng, Fan Yang, and Tao Li. Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications*, 41(7):3168–3177, 2014.

[149] Qing Li, Jia Wang, Yuanzhu Peter Chen, and Zhangxi Lin. User comments for news recommendation in forum-based social media. *Information Sciences*, 180(24):4929–4939, 2010.

[150] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 305–314, 2017.

[151] Yuliang Li, Aaron Feng, Jinfeng Li, Saran Mumick, Alon Y. Halevy, Vivian Li, and Wang-Chiew Tan. Subjective databases. *Proc. VLDB Endow.*, 12(11):1330–1343, 2019.

[152] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698, 2018.

[153] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[154] Yong Liu, Yinan Zhang, Qiong Wu, Chunyan Miao, Lizhen Cui, Binqiang Zhao, Yin Zhao, and Lu Guan. Diversity-promoting deep reinforcement learning for interactive recommendation. *arXiv preprint arXiv:1903.07826*, 2019.

[155] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[156] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[157] Frederic M Lord. *Applications of item response theory to practical testing problems.* Routledge, 1980.

[158] Masaki Matsubara, Ria Mae Borromeo, Sihem Amer-Yahia, and Atsuyuki Morishima. Task assignment strategies for crowd worker ability improvement. *Proc. ACM Hum. Comput. Interact.*, 5(CSCW2):1–20, 2021.

[159] Rishabh Mehrotra, Niannan Xue, and Mounia Lalmas. Bandit based optimization of multiple objectives on a music streaming platform. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3224–3233, 2020.

[160] Rosa J. Meijer and Jelle J. Goeman. Multiple testing of gene sets from gene ontology: Possibilities and pitfalls. *Briefings Bioinform.*, 17(5):808–818, 2016.

[161] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, 23:187–192, 2002.

[162] Bettina Mieth, Marius Kloft, Juan Antonio Rodríguez, Sören Sonnenburg, Robin Vobruba, Carlos Morcillo-Suárez, Xavier Farré, Urko M Marigorta, Ernst Fehr, Thorsten Dickhaus, et al. Combining multiple hypothesis testing with machine learning increases the statistical power of genome-wide association studies. *Scientific reports*, 6(1):1–14, 2016.

[163] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[164] Sein Minn. Bkt-lstm: Efficient student modeling for knowledge tracing and student performance prediction. *arXiv preprint arXiv:2012.12218*, 2020.

[165] Tom Michael Mitchell et al. *Machine learning*, volume 1. McGraw-hill New York, 2007.

[166] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[167] Israel Molina, Jordi Gómez i Prat, Fernando Salvador, Begoña Treviño, Elena Sulleiro, Núria Serre, Diana Pou, Sílvia Roure, Juan Cabezos, Lluís Valerio, et al. Randomized trial of posaconazole and benznidazole for chronic chagas' disease. *New England Journal of Medicine*, 370(20):1899–1908, 2014.

[168] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

[169] Mark EJ Newman. Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330, 2004.

[170] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, pages 677–686, 2014.

[171] Alexander G Nikolaev, Shounak Gore, and Venu Govindaraju. Engagement capacity and engaging team formation for reach maximization of online social media platforms. In *KDD*, pages 225–234, 2016.

[172] Sepideh Nikookar, Paras Sakharkar, Baljinder Smagh, Sihem Amer-Yahia, and Senjuti Basu Roy. Guided task planning under complex constraints. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 833–845. IEEE, 2022.

[173] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 37–76. Springer, 2015.

[174] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, volume 83. WoUongong, 1998.

[175] Behrooz Omidvar Tehrani. *Optimization-based User Group Management: Discovery, Analysis, Recommendation.* PhD thesis, Université Grenoble Alpes (ComUE), 2015.

[176] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, Pierre-Francois Dutot, and Denis Trystram. Multi-objective group discovery on the social web. In *ECML/PKDD*, pages 296–312. Springer, 2016.

[177] Behrooz Omidvar-Tehrani, Aurelien Personnaz, and Sihem Amer-Yahia. Guided text-based item exploration. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 3410–3420, 2022.

[178] Feiyang Pan, Qingpeng Cai, Pingzhong Tang, Fuzhen Zhuang, and Qing He. Policy gradients for contextual recommendations. In *The World Wide Web Conference*, pages 1421–1431, 2019.

[179] Weike Pan, Hao Zhong, Congfu Xu, and Zhong Ming. Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowledge-Based Systems*, 73:173–180, 2015.

[180] Higher Education Standards Panel. Final report–improving retention, completion and success in higher education, 2017.

[181] Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings 41st annual symposium on foundations of computer science*, pages 86–92. IEEE, 2000.

[182] Zachary A Pardos and Neil T Heffernan. Kt-idem: Introducing item difficulty to the knowledge tracing model. In *International conference on user modeling, adaptation, and personalization*, pages 243–254. Springer, 2011.

[183] Eli Pariser. *The filter bubble: What the Internet is hiding from you.* penguin UK, 2011.

[184] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. Performance factors analysis–a new alternative to knowledge tracing. *Online Submission*, 2009.

[185] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27:313–331, 1997.

[186] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[187] Pedro Pedreira, Chris Croswhite, and Luis Bona. Cubrick: indexing millions of records per second for interactive analytics. *Proceedings of the VLDB Endowment*, 9(13):1305–1316, 2016.

[188] Radek Pelánek and Jiří Řihák. Experimental analysis of mastery learning criteria. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pages 156–163, 2017.

[189] Leonardo Pellegrina, Matteo Riondato, and Fabio Vandin. Hypothesis testing and statistically-sound pattern mining (tutorial). In *Proc. of the 25th ACM SIGKDD Intl. Conf. on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 3215–3216, 2019.

[190] Radek Pelánek. Application of time decay functions and elo system in student modeling. *Proc. of Educational Data Mining*, pages 21–27, 01 2014.

[191] A. Personnaz, S. Amer-Yahia, L. Berti-Équille, M. Fabricius, and S. Subramanian. Balancing Familiarity and Curiosity in Data Exploration with Deep Reinforcement Learning. In *aiDM@SIGMOD*, pages 16–23, 2021.

[192] Alexandrin Popescul, Lyle H Ungar, David M Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. *arXiv preprint arXiv:1301.2303*, 2013.

[193] Ivens Portugal, Paulo Alencar, and Donald Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227, 2018.

[194] Bruno Pradel, Savaneary Sean, Julien Delporte, Sébastien Guérif, Céline Rouveirol, Nicolas Usunier, Françoise Fogelman-Soulié, and Frédéric Dufau-Joel. A case study in a recommender system based on purchase data. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–385. ACM, 2011.

[195] Shameem A Puthiya Parambath, Nicolas Usunier, and Yves Grandvalet. A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 15–22, 2016.

[196] Lijing Qin and Xiaoyan Zhu. Promoting diversity in recommendation by entropy regularizer. In *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer, 2013.

[197] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)*, 51(4):1–36, 2018.

[198] Idris Rabiu, Naomie Salim, Aminu Da'u, and Akram Osman. Recommender system based on temporal models: a systematic review. *Applied Sciences*, 10(7):2204, 2020.

[199] George Rasch. *Probabilistic models for some intelligence and attainment tests.* ERIC, 1993.

[200] Shaina Raza and Chen Ding. Deep dynamic neural network to trade-off between accuracy and diversity in a news recommender system. *arXiv preprint arXiv:2103.08458*, 2021.

[201] Mark D Reckase and Mark D Reckase. Unidimensional item response theory models. *Multidimensional item response theory*, pages 11–55, 2009.

[202] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[203] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[204] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: introduction and challenges. *Recommender systems handbook*, pages 1–34, 2015.

[205] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender Systems Handbook*. Springer, 2011.

[206] Elaine Rich. User modeling via stereotypes. *Cognitive science*, 3(4):329–354, 1979.

[207] Joseph Rollinson and Emma Brunskill. From predictive models to instructional policies. *International Educational Data Mining Society*, 2015.

[208] Etienne Roquain. Type i error rate control for testing many hypotheses: a survey with proofs. *Journal de la Société Française de Statistique*, 152(2):3–38, 2011.

[209] Leonid Rozenblit and Frank Keil. The misunderstood limits of folk science: An illusion of explanatory depth. *Cognitive science*, 26(5):521–562, 2002.

[210] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen. A Tutorial on Thompson Sampling. *Foundations and Trends in Machine Learning*, 11(1):1–96, 2018.

[211] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[212] Mark Sanderson and W Bruce Croft. The history of information retrieval research. *Proceedings of the IEEE*, 100(Special Centennial Issue):1444–1451, 2012.

[213] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[214] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[215] Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, et al. Analysis of recommendation algorithms for e-commerce. In *EC*, pages 158–167, 2000.

[216] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[217] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112, 2015.

[218] Peter Sedlmeier, Juliane Eberth, Marcus Schwarz, Doreen Zimmermann, Frederik Haarig, Sonia Jaeger, and Sonja Kunze. The psychological effects of meditation: a meta-analysis. *Psychological bulletin*, 138(6):1139, 2012.

[219] M. Seleznova, B. Omidvar-Tehrani, S. Amer-Yahia, and E. Simon. Guided Exploration of User Groups. *PVLDB*, 13(9):1469–1482, 2020.

[220] Giovanni Semeraro, Pierpaolo Basile, Marco de Gemmis, and Pasquale Lops. User profiles for personalizing digital libraries. In *Handbook of Research on Digital Libraries: Design, Development, and Impact*, pages 149–158. IGI Global, 2009.

[221] Juliet Popper Shaffer. Multiple hypothesis testing. *Annual review of psychology*, 46(1):561–584, 1995.

[222] D. Shahaf and C. Guestrin. Connecting the Dots Between News Articles. In *KDD*, pages 623–632, 2010.

[223] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

[224] Victor Shnayder and David Parkes. Practical peer prediction for peer assessment. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 4, pages 199–208, 2016.

[225] Zbyněk Šidák. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633, 1967.

[226] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts, Sixth Edition*. McGraw-Hill Book Company, 2011.

[227] Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.

[228] Monika Singh. Scalability and sparsity issues in recommender datasets: a survey. *Knowledge and Information Systems*, 62:1–43, 2020.

[229] Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *Ieee internet computing*, 21(3):12–18, 2017.

[230] Ian Soboroff and Charles Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, volume 99, pages 86–91. sn, 1999.

[231] Bo Song, Xin Yang, Yi Cao, and Congfu Xu. Neural collaborative ranking. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1353–1362, 2018.

[232] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. *Future generation computer systems*, 13(2-3):161–180, 1997.

[233] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. Choosing the best of both worlds: Diverse and novel recommendations through multi-objective reinforcement learning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 957–965, 2022.

[234] John D Storey and Robert Tibshirani. Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, 100(16):9440–9445, 2003.

[235] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[236] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[237] Zachari Swiecki, Hassan Khosravi, Guanliang Chen, Roberto Martinez-Maldonado, Jason M Lodge, Sandra Milligan, Neil Selwyn, and Dragan Gašević. Assessment in the age of artificial intelligence. *Computers and Education: Artificial Intelligence*, 3:100075, 2022.

[238] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Major components of the gravity recommendation system. *Acm Sigkdd Explorations Newsletter*, 9(2):80–83, 2007.

[239] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 323–332, 2015.

[240] Shailesh Tripathi, Galina V Glazko, and Frank Emmert-Streib. Ensuring the statistical soundness of competitive gene set approaches: gene filtering and genome-scale coverage are essential. *Nucleic acids research*, 41(7):e82–e82, 2013.

[241] John W Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pages 99–114, 1949.

[242] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. *Advances in neural information processing systems*, 26, 2013.

[243] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist*, 46(4):197–221, 2011.

[244] Saúl Vargas, Linas Baltrunas, Alexandros Karatzoglou, and Pablo Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 209–216, 2014.

[245] M. Vartak, S. Rahman, S. Madden, A. G. Parameswaran, and N. Polyzotis. SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. *PVLDB*, 8(13):2182–2193, 2015.

[246] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, volume 8, page 2182. NIH Public Access, 2015.

[247] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[248] V Vijayakumar, Subramaniyaswamy Vairavasundaram, R Logesh, and A Sivapathi. Effective knowledge based recommender system for tailored multiple point of interest recommendation. *International Journal of Web Portals (IJWP)*, 11(1):1–18, 2019.

[249] Lev Vygotsky. Zone of proximal development. *Mind in society: The development of higher psychological processes*, 5291:157, 1987.

[250] Jing Wang, Huan Deng, Bangtao Liu, Anbin Hu, Jun Liang, Lingye Fan, Xu Zheng, Tong Wang, and Jianbo Lei. Systematic evaluation of research progress on natural language processing in medicine over the past 20 years: bibliometric study on pubmed. *Journal of medical Internet research*, 22(1):e16816, 2020.

[251] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Defu Lian. A survey on session-based recommender systems. *ACM Comput. Surv.*, 54(7):154:1–154:38, 2022.

[252] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*, 2019.

[253] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th international conference on world wide web*, pages 391–400, 2017.

[254] Geoffrey I. Webb and François Petitjean. A multiple test correction for streams and cascades of statistical hypothesis tests. In *Proc. of the 22nd ACM SIGKDD Conf. on Knowledge Discovery and Data Mining, San Francisco, USA, Aug. 2016*, pages 1255–1264, 2016.

[255] Y. Wen, X. Zhu, S. Roy, and J. Yang. QAGView: Interactively Summarizing High-Valued Aggregate Query Answers. In *SIGMOD*, pages 1709–1712, 2018.

[256] Yuhao Wen, Xiaodan Zhu, Sudeepa Roy, and Jun Yang. Qagview: Interactively summarizing high-valued aggregate query answers. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1709–1712, 2018.

[257] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[258] Amy S Wu, Rob Farrell, and Mark K Singley. Scaffolding group learning in a collaborative networked environment. International Society of the Learning Sciences (ISLS), 2002.

[259] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503, 2017.

[260] Qiong Wu, Yong Liu, Chunyan Miao, Yin Zhao, Lu Guan, and Haihong Tang. Recent advances in diversified recommendation. *arXiv preprint arXiv:1905.06589*, 2019.

[261] Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. Discovering interesting patterns through user's interactive feedback. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 773–778. ACM, 2006.

[262] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 931–940, 2020.

[263] Emine Yilmaz, Javed A Aslam, and Stephen Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 587–594, 2008.

[264] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *arXiv preprint arXiv:1205.6700*, 2012.

[265] Brit Youngmann, Sihem Amer-Yahia, and Aurelien Personnaz. Guided exploration of data summaries. *arXiv preprint arXiv:2205.13956*, 2022.

[266] Cong Yu, Laks Lakshmanan, and Sihem Amer-Yahia. It takes variety to make a world: diversification in recommender systems. In *Proceedings of the 12th international conference on extending database technology: Advances in database technology*, pages 368–378, 2009.

[267] Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 85(1-2):41–75, 2011.

[268] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 549–553. SIAM, 2006.

[269] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)*, 52(1):1–38, 2019.

[270] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116, 2004.

[271] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103, 2018.

[272] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1040–1048, 2018.

[273] Zheguang Zhao, Lorenzo De Stefani, Emanuel Zgraggen, Carsten Binnig, Eli Upfal, and Tim Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, pages 527–540. ACM, 2017.

[274] Lei Zheng, Chun-Ta Lu, Fei Jiang, Jiawei Zhang, and Philip S Yu. Spectral collaborative filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 311–319, 2018.

[275] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.