RecRec: Algorithmic Recourse for Recommender Systems

Sahil Verma
vsahil@cs.washington.edu
University of Washington
Seattle, WA, USA

Ashudeep Singh
ashudeepsingh@pinterest.com
Pinterest, Inc.
San Francisco, CA, USA

Varich Boonsanong
varicb@cs.washington.edu
University of Washington
Seattle, WA, USA

John P. Dickerson
john@cs.umd.edu
University of Maryland
College Park, MD, USA

Chirag Shah
chirags@uw.edu
University of Washington
Seattle, WA, USA

Abstract

Recommender systems play an essential role in the choices people make in domains such as entertainment, shopping, food, news, employment, and education. The machine learning models underlying these recommender systems are often enormously large and black-box in nature for users, content providers, and system developers alike. It is often crucial for all stakeholders to understand the model's rationale behind making certain predictions and recommendations. This is especially true for the content providers whose livelihoods depend on the recommender system. Drawing motivation from the practitioners' need, in this work, we propose a recourse framework for recommender systems, targeted towards the content providers. Algorithmic *recourse* in the recommendation setting is a set of actions that, if executed, would modify the recommendations (or ranking) of an item in the desired manner. A recourse suggests actions of the form: "if a feature changes X to Y, then the ranking of that item for a set of users will change to Z." Furthermore, we demonstrate that RecRec is highly effective in generating valid, sparse, and actionable recourses through an empirical evaluation of recommender systems trained on three real-world datasets. To the best of our knowledge, this work is the first to conceptualize and empirically test a generalized framework for generating recourses for recommender systems. Full version of the paper is available at http://arxiv.org/abs/2308.14916.

CCS Concepts

• **Information systems → Recommender systems**; • **Computing methodologies** → *Machine learning*.

Keywords

Algorithmic Recourse, Recommender Systems, Explainable Recommender Systems

ACM Reference Format:
Sahil Verma, Ashudeep Singh, Varich Boonsanong, John P. Dickerson, and Chirag Shah. 2023. RecRec: Algorithmic Recourse for Recommender Systems. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3583780.3615181

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3615181

1 INTRODUCTION

Recommender systems are ubiquitous in online platforms today. They have a major influence on our choices in domains ranging from entertainment, social media, and shopping to news and education. These systems operate by filtering items from a large set to provide the most relevant ones to the user. Recommender systems can broadly be classified into two categories, content filtering and collaborative filtering [2, 5, 18, 21]. Content filtering represents each item using a set of features and recommends items to users based on the similarity to items consumed by the user in the past [3, 7, 14]. For example, if a user has purchased a cell phone recently, a content filtering system would recommend a phone case to the user based on its similarity to the phone. Collaborative filtering-based recommender systems recommend items to a user based on other users' interests who have a similar user history. For example, if a user recently bought a cell phone, a collaborative filtering-based recommender system would recommend the user with a phone case based on the information that other users who bought a cell phone also bought a phone case in the past.

Modern recommender systems are black-box models that has spurred the development of inherently interpretable recommender models or techniques to explain the factors influencing the recommendations [27]. We have also seen the rapid adoption of incorporating explainability in real-world recommender systems. Facebook offers "Why am I seeing this ad?" for every sponsored advertisement on its platform, and Amazon offers reasons why a product is recommended to you tab. These explanations are broadly termed as *feature attribution explanations* as they highlight a part of the features that lead to the recommendations.

All these explanations are primarily geared towards the end-users of the recommender platforms However, recommender systems are usually multi-stakeholder platforms with content providers and the system developers' interests baked into the system [1, 20, 28]. And these stakeholders also need transparency into the system. Since the content providers are dependent on the platform for their livelihood, they are interested in understanding the factors that influence their product's rank in the recommendations [6, 10, 11]. There have been several studies to understand the perspective of content providers offering services on several kinds of such platforms. Jhaver et al. [12] did a study with several Airbnb hosts to understand their perspectives. Rahman [16] interviewed freelancers working on Upwork. Razaq et al. [17] interviewd sellers on handmade product platform Etsy. Most content providers expressed the helplessness they face in understanding the factors that influence their product's rank on the platform [8, 12, 16, 17], and would like to gain transparency into it.

The kind of explainability that the content providers seek is similar to counterfactual explanations in classification systems [23, 24]. In general, counterfactual explanations describe a causal situation of the form: 'If I change X to Y, the outcome will change to Z'. Counterfactual explanations are frequently employed to answer questions like "What change in my features would help me to get the loan?". In recommender systems, a set of factors that improves the rank of a product offer a causal explanation of what changes would lead to an improved ranking for it. This set of factors are termed as *recourse* if the content providers can alter them to improve their product's rank. Since studies have shown that providing transparency into the algorithmic system improves the user's trust and adoption of the platforms [13], the motivating reason for incorporating counterfactual explanations in recommender systems is even more compelling.

The primary contribution of our work is to conceptualize a generalized framework for generating algorithmic recourse-based explanations for recommender systems. In this work, we use the terms counterfactual explanations and algorithmic recourse interchangeably, as we are using counterfactuals to provide recourses to the content providers, system developers, or curious end-users of a recommender system. Under this framework, we propose a novel algorithm, RecRec, that generates algorithmic recourses for a real-world recommender systems. RecRec casts the problem of finding recourses as an optimization problem that is solved using gradient descent in the feature space.

We first establish the desirable properties of a recourse in recommender systems (Section 2) and then cast the recourse generation problem as an optimization problem (Section 3), which RecRec solves. We conduct experiments with three different recommender systems and show RecRec's efficacy in generating recourses that achieve high success rate while satisfying the other desirable properties (Section 4). We make the following key contributions:

(1) We establish the desirable properties for algorithmic recourses in recommender systems (Section 2).
(2) We propose a novel approach called RecRec to generate recourses for a broad class of recommender system architectures (Section 3).
(3) We empirically demonstrate the effectiveness of RecRec through extensive experiments on three recommender systems trained on real-world datasets (Section 4).

2 DESIRABLE PROPERTIES OF ALGORITHMIC RECOURSE

To be effective for a content provider, a recourse in a recommender system setting should satisfy several desirable properties:

(1) *Valid:* A recourse when executed should lead to an increased exposure for the concerned item among the users of the target group, and therefore should have an improved rank after the recourse.
(2) *Sparse Changes:* A recourse should not change many features of the item. Being close to the original features makes the recourse more easily achievable [15].
(3) *Minimal side-effect* : A recourse should ideally only move the concerned item to an improved rank and have minimal side-effect on the ranks of the other items [19] (specially near the top ranks).

3 RecRec's ALGORITHM TO GENERATE RECOURSE

This section formulates the recourse from the perspective of a content provider wanting to change the features of an item so that it gets more exposure for users in a target group, i.e., within the top-k recommendations for these users.

Given an item's original features, r, the goal is to find updated features r', such that the item, *item*, is recommended within the *top-k* ranks for a group of target users In content filtering based systems, the features of an item are its attributes that are used to measure the similarity between different items, e.g. genres of a movie or book. Table 1 lists the notation used in this section.

Table 1: Notations for RecRec's Algorithm

Notation	Description
U:	Set of users
I:	Set of items
$\mathcal{I}$:	$\{v_i : i \in I, v_i \in \mathbb{R}^f\}$, v_i is the item features for item i.
$\mathcal{R}_j \in \mathbb{R}^{\lvert I_j \rvert}$:	$\mathcal{R}_j[k]$ denotes the rating given by user j to item $I_j[k]$
S:	Set of target users
a:	The item for which recourse is being sought

ALGORITHM 1: Generate recourse to move an item to the *top-k* recommendations for a target group of users.

```
Input  : Item features $\mathcal{I}$, the target user group S, ratings given by each user $\mathcal{R}$, the
         concerned item a, the desired rank of the item (e.g. top-10), hyperparameter $\lambda$,
         hyperparameter LearningRate
Output : New item feature for concerned item a: $v_a^*$
Function Compute_Recourse($\mathcal{I}, S, \mathcal{R}, a$, DesiredRank, LearningRate, $\lambda$)
    iterations ← 0
    $v_a, v'_a \leftarrow \mathcal{I}[a]$
    $S' \leftarrow S$
    while iterations < maxiterations do
        loss ← $-\left(\sum_{j \in S'} v_a \mathcal{I}_j^\top \odot \mathcal{R}_j\right) + \lambda * \lVert v'_a - v_a \rVert_1$
        // In each iteration, perform a gradient descent step for the loss
        $v'_a$ ← gradient_descent(LearningRate, loss)
        // Get the updated rank of the item for each user in S
        newrank ← get_updated_ranks($\mathcal{I}, v'_a, \mathcal{R}, S$) // Remove users from S for
            whom the concerned item is within the desired rank
        for $u \in S$ do
            if newrank[$u$] ∈ DesiredRank then
                $S' \leftarrow S \setminus u$
    return $v'_a$
Function get_updated_ranks($\mathcal{I}, v'_a, \mathcal{R}, S$)
    // Compute the new score of each item for each user
    for $u \in S$ do
        newscore ← $\emptyset$
        for $z \in \mathcal{I}$ do
            newscore[$z$] ← $\sum v_z \mathcal{I}_j^\top \odot \mathcal{R}_j$
        // sort the newscore in descending order to generate new ranks
        newrank[$u$] ← $\arg\text{sort}_{z \in \mathcal{I}}$ newscore[$z$]
    return newrank
```

Equation (1) shows the objective function that we need to optimize to generate a recourse that achieves the desired change in the rank of the item while not changing its feature too much.

$$v_a^* = \arg\max_{v'_a} \sum_{j \in S} v'_a \mathcal{I}_j^\top \odot \mathcal{R}_j \qquad s.t. \left\lVert v'_a - v_a \right\rVert_0 \leq \epsilon \tag{1}$$

We use gradient descent to optimize the objective function for which we need encode the twofold goal as a constrained optimization problem with sparsity inducing L1 norm as the constraint (with λ as the hyperparameter).

$$\arg\min_{v'_a} -\Big(\sum_{j \in S} v_a \mathcal{I}_j^\top \odot \mathcal{R}_j\Big) \; + \lambda * \left\| v'_a - v_a \right\|_1 \qquad (2)$$

Algorithm 1 provides the algorithm to generate recourses for content filtering based recommender systems. The algorithm takes as input the features of all item $\mathcal{I}$, the target user group S, the ratings given by the users $\mathcal{R}$, the item for which a recourse is desired a, and the desired rank for the item *DesiredRank*. It runs the gradient descent algorithm until convergence. In each iteration, the algorithm computes the loss given in eq. (2) and updates the features of the item using the gradient descent algorithm. The first term of the loss function is only computed for the subset of users in the target group S for whom the concerned item has not yet been ranked within the desired rank (line 9). This helps in two ways: a) encourages the algorithm to change the item features to move the item within the desired rank for a larger number of users from the target group, and b) limits the change in the users' recommendation lists (which is another desirable property of a recourse).

Iterative Hard Thresholding We use L1 norm to induce sparsity in the change between the original and the recourse item features. However, this might not be sufficient to ensure that the recourse is sparse. Therefore, after the convergence of algorithm 1, we iteratively set the values of the features with the smallest absolute difference with the original features to the original feature value at those indices, a process termed as iterative hard thresholding [4]. This leads to a tradeoff between the number of users for whom the item is moved within the desired rank (*success rate*) and the sparsity of the recourse. We continue iteratively hard thresholding until we start losing more than a certain percentage of the success rate (in our experiments we set this threshold to 20%).

4 EVALUATION

We performed experiments using three real-world recommender systems to measure RecRec's efficacy, efficiency, and side-effect when generating recourses for items at various ranks:

(1) *MovieLens-100K [9]:* This movie recommendation dataset has about 1000 movies and 1700 users who gave a total of 100K ratings. Each movie has information such as its summary, actors, directors, and genre. We consider the movie summary as mutable and others as immutable features. Using standard NLP data processing we featurize each summary in about 9500 dimensions. Dot product between the feature vectors of two movies determine their similarity. If two movies are very similar and a user has liked one of them, the recommender system will recommend the other. MovieLens also provides the rating a user has given to certain movies which we use to weigh similarity between movies for providing more accurate recommendations. The dataset also provides user metadata, like age and occupation. In experiments, we use age ranges to group users into 5 equi-sized groups that content providers want to target.

(2) *AliEC Ads [22]:* This dataset is an ad click prediction dataset provided by Alibaba. It has about 5600 ads and 13200 users who interacted with 1.4 million ads. For each ad, it provides information like its category, brand, and price. All features are considered mutable. We one-hot encode all categorical features and use price after normalization to feature each ad in about 2300 dimensions. Again, the dot product between feature vectors of two ads determine their similarity. The dataset also provides user metadata such as age level and gender. In experiments, we use age level to group users into five equi-sized groups that content providers want to target.

(3) *Goodreads [25, 26]:* This book recommendation dataset has about 4300 books and 11200 users who have a total of 1.3 million ratings. Each book has features like its short description, genre, number of reviews, hardcover or ebook format. All features are considered mutable. Using standard NLP data processing, we featurize each movie description and other features in about 17400 dimensions. The dot product measures the similarity between two books, and user ratings are used to weigh similarity. The dataset does not provide user metadata, and therefore we group users into five equi-sized groups based on the number of ratings they provided.

4.1 Experimental Methodology

For all three recommender systems, we group the users into 5 equi-sized group either based on available metadata (like age) or based on the number of ratings they provided. Our experimental setup portrays a scenario where a content provider wants to increase the exposure of their item to a group of users, the *target group*. RecRec provides a recourse to the content provider and after its execution, if the rank of their item improves to being within the top-k recommendations for users in the target group, we would say that the item's exposure has increased and the provided recourse was valid. In experiments, we consider an item to have increased exposure for a user if after the recourse it is within the top-10 recommendation for that user. Even though the content providers would want to target a large group of users, optimizing the losses in eq. (2) for all users in the target group can be computationally expensive. Therefore, we experiment using a sampling procedure. We randomly sample a small percentage of the users in the target group and minimize the loss only over them. However, we report the percentage of users in the *entire target group* that get an increased exposure to the item. Intuitively, if we sample more users from the target group, a higher percentage of users would see the item within their top-10 recommendations.

Metrics. We report the following metrics for RecRec:

(1) *Success Rate:* For each recommender system, we report the percentage for users from the target group who see the concerned item withing their top-10 recommendations. We report this various sampling sizes. A higher value for this metric is better.

(2) *Number of changes required:* This metric is computed as the L0 norm of the difference between the original features and the new features after recourse. A lower value for this metric is better.

(3) *Side-effect on user recommendations:* To estimate the impact of a recourse on the users' original recommendations, we measure the similarity between the them and the new recommendations. We use rank-biased overlap (RBO) as a measure of similarity between the two ranked lists. Since the change in recommendations at top ranks matter more, we weigh them more when measuring RBO similarity (p = 0.5). A higher value for this metric is better.

All the metrics are reported for items at various original ranks. Specifically, we generate recourses for items whose original ranks

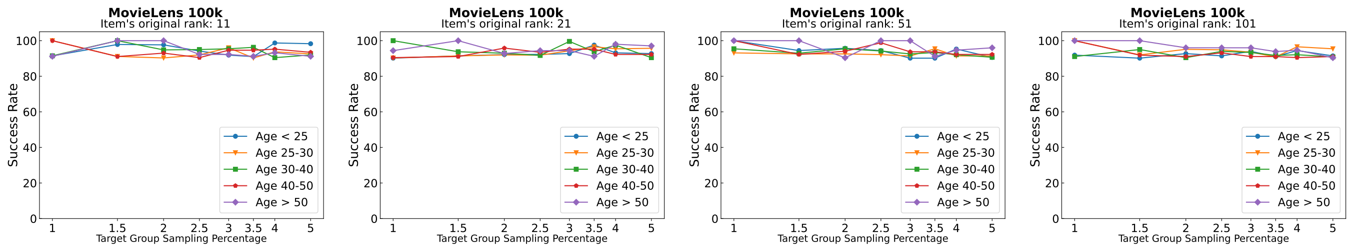

Figure 1: RecRec's success rate for items at original ranks 11, 21, 51, and 101 for the recommender system trained on MovieLens-100K. RecRec gets 100% success rate for all user groups and items at all original ranks with very small sampling size starting from 1% of the user group.

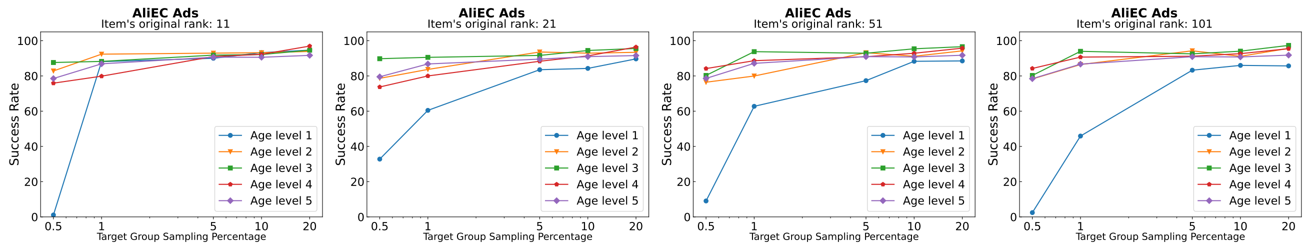

Figure 2: RecRec's success rate for items at original ranks 11, 21, 51, and 101 for the recommender system trained on AliEC Ads. RecRec gets more than 80% success for most user groups and items at all original ranks with very small sampling size starting from 1% of the user group, and increases to 100% for with 20% sampling.

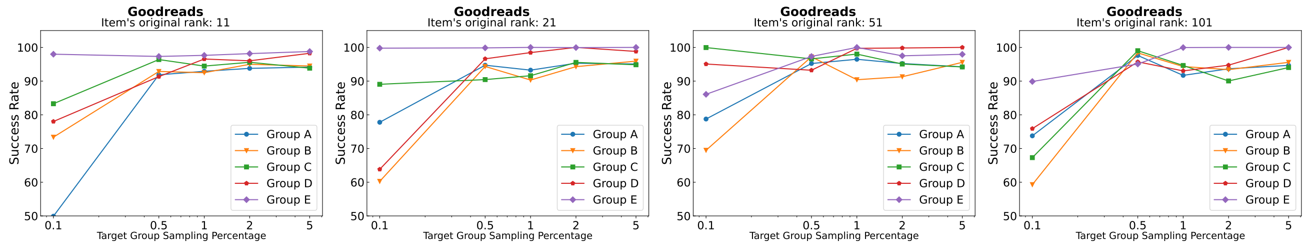

Figure 3: RecRec's success rate for items at original ranks 11, 21, 51, and 101 for the recommender system trained on Goodreads. RecRec gets more than 90% success for all user groups and items at all original ranks with very small sampling size starting from 0.5% of the user group, and increases to 100% for with 2-5% sampling.

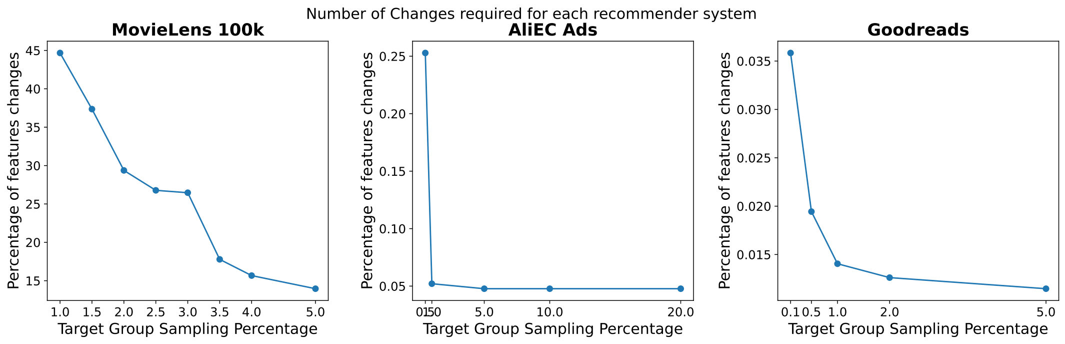

Figure 4: The percentage of item's features that need to be changed to execute the recourse. The plots are for the recommender systems trained on MovieLens-100K, AliEC Ads, and Goodreads. With increasing sampling percentage, the number of changes required to get a recourse decreases, and eventually becomes negligible.

are 11, 21, 51, and 101 (item rank averaged over the users in the target group). For rigorosity, we ensure that an item whose rank is already within top-10 for more than 1% of the users in the target group is not considered for getting a recourse.

Discussion of the results and the conclusions are the full version of the paper available at http://arxiv.org/abs/2308.14916.

References

[1] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Recommender Systems as Multistakeholder Environments. In *Proceedings of the 25th Conference*

on User Modeling, Adaptation and Personalization (Bratislava, Slovakia) *(UMAP '17)*. Association for Computing Machinery, New York, NY, USA, 2 pages. https://doi.org/10.1145/3079628.3079657

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. 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 (jun 2005), 734–749.

[3] Charu C Aggarwal. 2016. Content-based recommender systems. In *Recommender systems*. Springer, 139–166.

[4] Thomas Blumensath and Mike E. Davies. 2009. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis* 27, 3 (2009), 265–274. https://www.sciencedirect.com/science/article/pii/S1063520309000384

[5] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.

[6] Eliane Léontine Bucher, Peter Kalum Schou, and Matthias Waldkirch. 2020. Pacifying the algorithm - Anticipatory compliance in the face of algorithmic management in the gig economy. *Organization* 28 (2020), 44 – 67.

[7] Laurent Candillier, Kris Jack, Françoise Fessant, and Frank Meyer. 2009. State-of-the-art recommender systems. In *Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling*. IGI Global, 1–22.

[8] Motahhare Eslami, Aimee Rickman, Kristen Vaccaro, Amirhossein Aleyasen, Andy Vuong, Karrie Karahalios, Kevin Hamilton, and Christian Sandvig. 2015. "I Always Assumed That I Wasn't Really That Close to [Her]": Reasoning about Invisible Algorithms in News Feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 10 pages. https://doi.org/10.1145/2702123.2702556

[9] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[10] Maya Holikatti, Shagun Jhaver, and Neha Kumar. 2019. Learning to Airbnb by Engaging in Online Communities of Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW (2019), 19 pages. https://doi.org/10.1145/3359330

[11] Mohammad Hossein Jarrahi and Will Sutherland. 2019. Algorithmic Management and Algorithmic Competencies: Understanding and Appropriating Algorithms in Gig Work. In *Information in Contemporary Society*. Springer International Publishing, Cham, 578–589.

[12] Shagun Jhaver, Yoni Karpfen, and Judd Antin. 2018. Algorithmic Anxiety and Coping Strategies of Airbnb Hosts. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 12 pages. https://doi.org/10.1145/3173574.3173995

[13] Min Kyung Lee. 2018. Understanding perception of algorithmic decisions: Fairness, trust, and emotion in response to algorithmic management. *Big Data & Society* 5 (2018).

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

[15] Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence* 267 (2019). https://doi.org/10.1016/j.artint.2018.07.007

[16] Hatim A. Rahman. 2021. The Invisible Cage: Workers' Reactivity to Opaque Algorithmic Evaluations. *Administrative Science Quarterly* 66, 4 (2021), 945–988. https://doi.org/10.1177/00018392211010118

[17] Lubna Razaq, Beth Kolko, and Gary Hsieh. 2022. Making crafting visible while rendering labor invisible on the Etsy platform. In *Designing Interactive Systems Conference 2021* (Virtual Event, USA) *(DIS '22)*. Association for Computing Machinery, New York, NY, USA, 15 pages.

[18] Kunal Shah, Akshaykumar Salunke, Saurabh Dongare, and Kisandas Antala. 2017. Recommender systems: An overview of different approaches to recommendations. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. IEEE, 1–4.

[19] Guy Shani and Asela Gunawardana. 2011. *Evaluating Recommendation Systems*. Springer US, Boston, MA, 257–297. https://doi.org/10.1007/978-0-387-85820-3_8

[20] Özge Sürer, Robin Burke, and Edward C. Malthouse. 2018. Multistakeholder Recommendation with Provider Constraints. In *Proceedings of the 12th ACM Conference on Recommender Systems* (Vancouver, British Columbia, Canada) *(RecSys '18)*. Association for Computing Machinery, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240350

[21] Poonam B Thorat, RM Goudar, and Sunita Barve. 2015. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications* 110, 4 (2015), 31–36.

[22] Tianchi. 2018. Ad Display/Click Data on Taobao.com. https://tianchi.aliyun.com/dataset/dataDetail?dataId=56

[23] Sahil Verma, John Dickerson, and Keegan Hines. 2020. Counterfactual Explanations for Machine Learning: A Review. arXiv:2010.10596 [cs.LG]

[24] Sandra Wachter, Brent Mittelstadt, and Chris Russell. 2018. Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR. arXiv:1711.00399 [cs.AI]

[25] Mengting Wan and Julian J. McAuley. 2018. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. ACM, 86–94. https://doi.org/10.1145/3240323.3240369

[26] Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian J. McAuley. 2019. Fine-Grained Spoiler Detection from Large-Scale Review Corpora. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2605–2610. https://doi.org/10.18653/v1/p19-1248

[27] Yongfeng Zhang and Xu Chen. 2020. Explainable Recommendation: A Survey and New Perspectives. *Found. Trends Inf. Retr.* 14 (2020), 1–101.

[28] Yong Zheng. 2019. Multi-Stakeholder Recommendations: Case Studies, Methods and Challenges *(RecSys '19)*. Association for Computing Machinery, New York, NY, USA, 2 pages. https://doi.org/10.1145/3298689.3346951