Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Sister Conferences Best Papers Track

Rewiring What-to-Watch-Next Recommendations to Reduce Radicalization Pathways (Extended Abstract)*

Francesco Fabbri[1], **Yanhao Wang**[2], **Francesco Bonchi**[3,4], **Carlos Castillo**[5,6] and **Michael Mathioudakis**[7]
[1]Spotify, Barcelona, Spain
[2]East China Normal University, Shanghai, China
[3]CENTAI Institute, Turin, Italy
[4]Eurecat, Barcelona, Spain
[5]ICREA, Barcelona, Spain
[6]Universitat Pompeu Fabra, Barcelona, Spain
[7]University of Helsinki, Helsinki, Finland
francescof@spotify.com, yhwang@dase.ecnu.edu.cn, bonchi@centai.eu, chato@icrea.cat, michael.mathioudakis@helsinki.fi

Abstract

Recommender systems typically suggest to users content similar to what they consumed in the past. A user, if happening to be exposed to strongly polarized content, might be steered towards more and more radicalized content by subsequent recommendations, eventually being trapped in what we call a "radicalization pathway". In this paper, we investigate how to mitigate radicalization pathways using a graph-based approach. We model the set of recommendations in a "what-to-watch-next" (W2W) recommender as a directed graph, where nodes correspond to content items, links to recommendations, and paths to possible user sessions. We measure the segregation score of a node representing radicalized content as the expected length of a random walk from that node to any node representing non-radicalized content. A high segregation score thus implies a larger chance of getting users trapped in radicalization pathways. We aim to reduce the prevalence of radicalization pathways by selecting a small number of edges to "rewire", so as to minimize the maximum of segregation scores among all radicalized nodes while maintaining the relevance of recommendations. We propose an efficient yet effective greedy heuristic based on the absorbing random walk theory for the rewiring problem. Our experiments on real-world datasets confirm the effectiveness of our proposal.

1 Introduction

"*What-to-watch-next*" (W2W) recommenders are a key feature of video sharing platforms [Zhao *et al.*, 2019], as they sustain user engagement, thus increasing content views and driving advertisement and monetization. However, recent studies have raised serious concerns about the potential role played by W2W recommenders, especially in driving users towards undesired or polarizing content [Ledwich and Zaitsev, 2020; Cinus *et al.*, 2022]. Specifically, radicalized communities on social networks and content-sharing platforms have been recognized as keys to news consumption and opinion formation around politics and related subjects [Lewis, 2018; Weiss and Winter, 2018; Roose, 2019]. Recent work highlighted the role of recommender systems, which may steer users towards radicalized content, eventually building "*radicalization pathways*" [Lewis, 2018; Ribeiro *et al.*, 2020; McCauley and Moskalenko, 2008], i.e., a user might be further driven towards radicalized content even when this was not her initial intent. Therefore, we study how to reduce the prevalence of radicalization pathways in W2W recommenders while maintaining the relevance of recommendations in our WWW '22 paper [Fabbri *et al.*, 2022].

Formally, we model a W2W recommender system as a directed labeled graph where nodes correspond to videos (or other types of content), and directed edges represent recommendation links from one node to another. In this scenario, each video has the same number d of recommendation links; thus, every node in the graph has the same out-degree d. Moreover, each node has a binary label such as "harmful" (e.g., radicalized) or "neutral" (e.g., non-radicalized). A user's browsing activity through the W2W recommendations is modeled as a *random walk* on the graph. After visiting a node (e.g., watching a video), the user moves to one of the d recommended videos with a probability that depends on its visibility or ranking in the recommendation list. In this setting, for each harmful node v, we measure the expected number of consecutive harmful nodes visited in a random walk before reaching any neutral node. We call this measure the "*segregation*" score of node v: Intuitively, it quantifies how easy it is to get "stuck" in radicalization pathways starting from a given node. Our goal is to reduce the segregation of the graph while guaranteeing that the quality of recom-

*This is an extended abstract of [Fabbri *et al.*, 2022] that won the best paper award at WWW '22.

mendations is maintained, where the quality is measured by the *normalized discount cumulative gain* [Biega *et al.*, 2018; Järvelin and Kekäläinen, 2002] (nDCG) of each node. An important challenge is that the underlying recommendation graph has some level of homophily intrinsically because, given that the W2W seeks to recommend relevant videos, it is likely to link harmful nodes to other harmful nodes.

We formulate the problem of reducing the segregation of the graph as picking k rewiring operations on edges (w.r.t. modifications in the lists of recommended videos for some nodes) so as to minimize the maximum of segregation scores among all harmful nodes while maintaining recommendation quality measured by nDCG above a given threshold for all nodes. We prove that our k-REWIRING problem is NP-hard and NP-hard to approximate within any factor. We, therefore, turn our attention to designing efficient and effective heuristics. Our proposed algorithm is based on the *absorbing random walk theory* [Mavroforakis *et al.*, 2015], thanks to which we can efficiently compute the segregation score of each node and update it after every rewiring operation. Our algorithm finds a set of k rewiring operations by greedily choosing the optimal rewiring for the special case of $k = 1$ – i.e., the 1-REWIRING problem, then updates the segregation score of each node. We further design a sorting and pruning strategy to avoid unnecessary attempts and thus improve the efficiency of searching for optimal rewiring. Finally, we present experiments on real-world datasets in the context of video sharing. We compare our proposed algorithm against several baselines, including an algorithm for suggesting new edges to reduce radicalization in Web graphs [Haddadan *et al.*, 2021]. The results show that our algorithm outperforms existing solutions in mitigating radicalization pathways. Our code and data are publicly available at https://github.com/FraFabbri/rewiring-what-to-watch.

2 Preliminaries

Let us consider a set V of n items and a matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where each entry $s_{uv} \in [0, 1]$ at position (u, v) denotes the relevance score of an item v given that a user has browsed an item u. This expresses the likelihood that a user who has just watched u would be interested in watching v. Typically, a recommender system selects the d most relevant items to compose the recommendation list $\Gamma^+(u)$ of u, where the number of recommendations d is a design constraint (e.g., given by the size of the app window). We assume that the system selects the top-d items v w.r.t. s_{uv} and that their relevance score uniquely determines the ranking of the d items in $\Gamma^+(u)$. For each $v \in \Gamma^+(u)$, we use $i_u(v)$ to denote its ranking in $\Gamma^+(u)$. After a user has seen u, the user will find the next item to see from $\Gamma^+(u)$, and the probability p_{uv} of selecting $v \in \Gamma^+(u)$ depends on the ranking $i_u(v)$ of v in $\Gamma^+(u)$. More formally, $p_{uv} = f(i_u(v))$, where f is a non-increasing function that maps from $i_u(v)$ to p_{uv} with $\sum_{v \in \Gamma^+(u)} p_{uv} = 1$.

This setting is modeled as a directed probabilistic d-regular graph $G = (V, E, \mathbf{M})$, where the node set V corresponds to the set of all n items, the edge set E comprises $n \cdot d$ edges where each node $u \in V$ has d out-edges connected to nodes in $\Gamma^+(u)$, and $\mathbf{M}$ is an $n \times n$ transition matrix with a value of p_{uv} for each $(u, v) \in E$ and 0 otherwise. A user's browsing session is modeled as a random walk on G starting from an arbitrary node in V with transition probability p_{uv} for each $(u, v) \in E$. The nodes in V are divided into two disjoint subsets: V_n and V_h w.r.t. "neutral" (e.g., not-radicalized) and "harmful" (e.g., radicalized) items.

The risk we want to mitigate is having users stuck in a long sequence of harmful nodes while performing a random walk. In order to quantify this phenomenon, we define the measure of *segregation* score. Given a set $S \subset V$ of nodes and a node $u \in V \setminus S$, we use a random variable $T_u(S)$ to indicate the first instant when a random walk starting from u reaches (or "hits") any node in S. We define $\mathbb{E}_G[T_u(S)]$ as the *hitting length* of u w.r.t. S, where the expectation is over the space of all possible random walks on G starting from u. In our case, we define the segregation score z_u of node $u \in V_h$ by its expected hitting length $\mathbb{E}_G[T_u(V_n)]$ w.r.t. V_n. The segregation $Z(G)$ of graph G is defined by the maximum of segregation scores among all nodes in V_h – i.e., $Z(G) = \max_{u \in V_h} z_u$. In the following, we omit the argument G from $Z(G)$ when it is clear from the context.

Our main problem is to mitigate the effect of segregation by modifying the structure of G. Specifically, we aim to find a set O of rewiring operations on G, each of which removes an existing edge $(u, v) \in E$ and inserts a new one $(u, w) \notin E$ instead, such that $Z(G^O)$ is minimized, where G^O is the new graph after performing O on G. For simplicity, we require that $u, v \in V_h$, $w \in V_n$, and $p_{uv} = p_{uw}$. In other words, each rewiring operation changes the recommendation list $\Gamma^+(u)$ of u by replacing one (harmful) item $v \in \Gamma^+(u)$ with another (neutral) item $w \notin \Gamma^+(u)$ and keeping the ranking $i_u(w)$ of w the same as the ranking $i_u(v)$ of v in $\Gamma^+(u)$. Another goal, which is often conflicting, is to preserve the relevance of recommendations after performing the rewiring operations. Besides requiring only a predefined number k of rewirings, we also consider an additional constraint on the loss in the quality of the recommendations. For this purpose, we adopt the well-known *normalized discounted cumulative gain* (nDCG) [Järvelin and Kekäläinen, 2002], i.e., the ratio of the DCGs between $\Gamma^+(u)$ after rewiring operations and the original (ideal) recommendation list $\Gamma_0^+(u)$, to evaluate the quality loss $L(\Gamma^+(u))$. Let $o = (u, v, w)$ be a rewiring operation that deletes (u, v) and adds (u, w) and O be a set of rewiring operations. For ease of presentation, we define a function $\Delta(O) \triangleq Z(G) - Z(G^O)$ to denote the decrease in the segregation after performing the rewiring operations in O and updating G to G^O. Based on all the above notions, we formally give the following problem definition.

Problem 1 (k-REWIRING)**.** *Given a directed probabilistic graph $G = (V, E, \mathbf{M})$, a positive integer $k \in \mathbb{Z}^+$, and a threshold $\tau \in (0, 1)$, find a set O of k rewiring operations that maximizes $\Delta(O)$, under the constraint that $L(\Gamma^+(u)) \geq \tau$ for each node $u \in V$.*

The hardness of the k-REWIRING problem is analyzed in the following theorem.

Theorem 1. *The k-REWIRING problem is NP-hard and NP-hard to approximate within any factor.*

See [Fabbri *et al.*, 2022, Appendix A] for the proof.

Algorithm 1: OPTIMAL 1-REWIRING

Input : Graph $G = (V, E, \mathbf{M})$, fundamental matrix $\mathbf{F}$, segregation vector $\mathbf{z}$, threshold τ
Output: Optimal rewiring operation o^*

1 Initialize $\Omega \leftarrow \emptyset$, $o^* \leftarrow$ NULL, $\Delta^* \leftarrow 0$;
2 **foreach** *node* $u \in V_h$ **do**
3 Find $w \in V_n$ s.t. $(u, w) \notin E$ with the largest s_{uw};
4 **foreach** $v \in V_h$ *with* $(u, v) \in E$ **do**
5 Add $o = (u, v, w)$ to Ω if $L(\Gamma^+(u)) \geq \tau$ after replacing (u, v) with (u, w);
6 Sort V_h as $\langle h_1, \ldots, h_{n_h} \rangle$ in descending order of z_h;
7 **foreach** $o \in \Omega$ **do**
8 Compute $\Delta(h_1, o)$ using Eq. 1;
9 **if** $z'_{h_1} > z_{h_2}$ **then**
10 $\Delta(o) \leftarrow \Delta(h_1, o)$;
11 **else**
12 Find the largest $j > 1$ such that $z'_{h_1} < z_{h_j}$;
13 Compute $\Delta(h_i, o)$ for each $i = 2, \ldots, j$;
14 $\Delta(o) \leftarrow z_{h_1} - \max_{i \in [1,j]} z'_{h_i}$;
15 **if** $\Delta(o) > \Delta^*$ **then**
16 $o^* \leftarrow o$ and $\Delta^* \leftarrow \Delta(o)$;
17 **return** o^*;

Algorithm 2: HEURISTIC k-REWIRING

Input : Graph $G = (V, E, \mathbf{M})$, threshold τ, size k
Output: A set O of k rewiring operations

1 Compute the initial $\mathbf{F}$ and $\mathbf{z}$ based on $\mathbf{M}$;
2 Acquire Ω using Lines 2–5 of Alg. 1;
3 Initialize $O \leftarrow \emptyset$;
4 **for** $i \leftarrow 1, 2, \ldots, k$ **do**
5 Run Lines 6–16 of Alg. 1 to get $o^* = (u^*, v^*, w^*)$;
6 $O \leftarrow O \cup \{o^*\}$ and update G, $\mathbf{M}$, $\mathbf{F}$, and $\mathbf{z}$ for o^*;
7 Delete the existing rewiring operations of u^* from Ω and add new possible operations of u^* to Ω;
8 **if** $\Omega = \emptyset$ **then break**;
9 **return** O;

3 Algorithms

Due to the NP-hardness of k-REWIRING, we propose an efficient heuristic for the problem. The heuristic is motivated by the following observation: the special case of k-REWIRING when $k = 1$, which we call 1-REWIRING, is solvable in polynomial time. Given an optimal 1-REWIRING algorithm, k-REWIRING can be addressed by running it k times. Next, we first present an optimal 1-REWIRING algorithm (Section 3.1) and then a greedy k-REWIRING algorithm (Section 3.2).

3.1 Optimal 1-REWIRING Algorithm

We now introduce our method to find the optimal solution o^* of 1-REWIRING, i.e., the rewiring operation that maximizes $\Delta(o)$ among all $o \in \Omega$. The detailed procedure is presented in Algorithm 1, to which the fundamental matrix $\mathbf{F} = (\mathbf{I} - \mathbf{M}_{hh})^{-1}$, where $\mathbf{M}_{hh}$ is the sub-matrix of $\mathbf{M}$ w.r.t. all harmful nodes, and the vector $\mathbf{z}$ consisting of the segregation scores of all nodes before rewiring are given as input. The algorithm proceeds in two steps: (1) candidate generation, as described in Lines 2–5, which returns a set Ω of possible rewiring operations that include the optimal 1-REWIRING, and (2) optimal rewiring search, as described in Lines 6–16, which computes the objective value for each candidate rewiring to identify the optimal one.

In the first step, we should exclude all rewiring operations that violate the quality constraint for candidate generation. Towards this end, we do not consider any rewiring operation for any node u that will lead to the *normalized discount cumulative gain* (nDCG) of u below the threshold τ. Since $\Delta(h, o)$ of node h w.r.t. $o = (u, v, w)$ is independent of (u, w), for a specific node u, we fix w to the neutral node with the highest relevance score s_{uw} and $(u, w) \notin E$ so that as many rewiring operations as possible are feasible. Then, we should select the node v where $(u, v) \in E$ will be replaced. We should guarantee that $L(\Gamma^+(u)) \geq \tau$ after (u, v) is replaced by (u, w). For each node $v \in \Gamma^+(u)$, we take s_{uv} and s_{uw} into the formula of nDCG calculation [Järvelin and Kekäläinen, 2002]. If $L(\Gamma^+(u)) \geq \tau$, we will list $o = (u, v, w)$ as a candidate. After considering each node $u \in V_h$, we generate the set Ω of all candidate rewiring operations.

The second step is to search for the optimal rewiring operation o^* from Ω. We first sort all harmful nodes in descending order of their segregation scores as $\langle h_1, h_2, \ldots, h_{n_h} \rangle$, where h_i is the node with the i-th largest segregation score. Since we want to minimize the maximum segregation, we can focus on the first few nodes with the largest segregation scores and ignore the remaining ones. We need to compute $\Delta(o)$ for each $o \in \Omega$ and always keep the maximum of $\Delta(o)$. In particular, for any node h and operation $o = (u, v, w)$, we calculate $\Delta(h, o)$ as:

$$\Delta(h, o) = z_h - z'_h = \frac{f_{hu} z_v}{1/p_o + f_{vu}}. \tag{1}$$

After evaluating every $o \in \Omega$, the one maximizing $\Delta(o)$ is o^*. Furthermore, to compute $\Delta(o)$ for operation o, we perform the following steps: (1) compute $\Delta(h_1, o)$ using Eq. 1; (2) if $z'_{h_1} > z_{h_2}$, then $\Delta(o) = \Delta(h_1, o)$; (3) otherwise, find the largest j such that $z'_{h_1} < z_{h_j}$, compute $\Delta(h_i, o)$ for each $i = 2, \ldots, j$; in this case, we have $\Delta(o) = z_{h_1} - \max_{i \in [1,j]} z'_{h_i}$. The above steps guarantee finding the optimal rewiring operation, as all rewiring operations that might be optimal have been considered.

3.2 Heuristic k-REWIRING Algorithm

Our k-REWIRING algorithm based on the 1-REWIRING algorithm is presented in Algorithm 2. Its basic idea is to find the k rewiring operations by running the 1-REWIRING algorithm k times. The first step is initializing the fundamental matrix $\mathbf{F}$ and the segregation vector $\mathbf{z}$. In our implementation, $\mathbf{F}$ and $\mathbf{z}$ are approximately computed through the power iteration method in [Mavroforakis *et al.*, 2015]. Then, the candidate generation procedure is the same as in Algorithm 1. Next, it runs k iterations for getting k rewiring operations. At each iteration, it also searches for the optimal rewiring operation $o^* = (u^*, v^*, w^*)$ among Ω as Algorithm 1. After

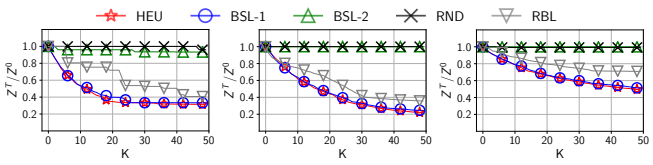
that, G, $\mathbf{M}$, $\mathbf{F}$, and $\mathbf{z}$ are updated according to o^* (see [Fabbri *et al.*, 2022] for the update procedure). Since the existing rewiring operations of u^* are not feasible anymore, it will regenerate new possible operations of u^* based on the updated $\Gamma^+(u^*)$ and the threshold τ to replace the old ones. Finally, the algorithm terminates when k rewiring operations have been found, or there is no feasible operation.

4 Experiments

We conduct experiments to show the effectiveness of our algorithm on mitigating radicalization pathways compared to existing algorithms. Note that we only present a small fraction of experiments here due to space limitations. Please refer to [Fabbri *et al.*, 2022] for more comprehensive results.

Datasets. Our experiments are conducted on the YouTube dataset [Ribeiro *et al.*, 2020] in the context of video sharing, which contains 330,925 videos and 2,474,044 recommendations. The dataset includes node labels such as "*alt-right*", "*alt-lite*", "*intellectual dark web*", and "*neutral*". We categorize the first three classes as "radicalized" or "harmful" and the last class as "neutral" following the analysis done by this dataset's curators [Ribeiro *et al.*, 2020], in which these three classes are shown to be overlapping in terms of audience and content. When generating the recommendation graphs, we only include videos having a minimum of 100k views. We consider the video-to-video recommendations collected via simulations as implicit feedback interactions, where the video-to-video interactions are formatted as a square matrix, with position (u, v) containing the number of times the user jumped from video u to video v. Using alternating least squares [Hu *et al.*, 2008] (ALS), we first derive the latent dimensions of the matrix, generate the scores (normalized to $[0, 1]$), and build the recommendation lists for each video. We eventually create different d-regular graphs with $d \in \{5, 10, 20\}$. Finally, we have three recommendation graphs, namely, **YT-D5-S**, **YT-D10-S**, and **YT-D20-S**.

Algorithms. We compare our proposed heuristic (**HEU**) algorithm for k-REWIRING with three baselines and one existing algorithm. The first baseline (**BSL-1**) selects the set of k rewiring operations by running Algorithm 1. Instead of picking only one rewiring operation, it picks the k operations with the largest values of Δ all at once. The second baseline (**BSL-2**) considers the best possible k rewiring operations by looking at the initial values of the vector $\mathbf{z}$. It firsts selects the k nodes with the largest z values, then among the possible rewiring operations from those nodes, it returns the k operations with the largest values of Δ. The third baseline (**RND**) picks k random rewiring operations from all the candidates. Finally, the existing method we compare with is the *RePBubLik* algorithm [Haddadan *et al.*, 2021; Haddadan *et al.*, 2022] (**RBL**). We adapt *RePBubLik* to our k-REWIRING problem as follows: (1) we run it to return a list of potential edges to be added for reducing the structural bias of the harmful nodes; (2) for each potential insertion, in order to generate a rewiring operation, we check among the existing edges to find the one edge that meets the quality constraint τ after being replaced by the new edge; (3) we finally select a set of k rewiring operations from the previous step.



(a) YT-D5-S (b) YT-D10-S (c) YT-D20-S

Figure 1: Performance comparison on the YouTube dataset.

Hardware and Implementation. All the experiments were conducted on a server running Ubuntu 16.04 with an Intel Broadwell 2.40GHz CPU and 29GB of memory. Our algorithms were implemented in Python 3. For *RePBubLik*, we used the implementation published by original authors.

Experimental Results. In Figure 1, we present the results on the YouTube recommendation graphs. On each graph, we evaluate the performance of each algorithm along 50 rewiring operations with the threshold of quality constraint fixed to $\tau = 0.9$. We keep track of the relative decrease in the segregation Z^T/Z^0 after each rewiring operation, where Z^0 is the initial segregation and Z^T is the segregation after T rewiring operations. On all the graphs, it is clear that our greedy heuristic (**HEU**) outperforms all the competitors. On the graph with the smallest out-degree ($d = 5$), it decreases Z by over 40% within only 10 rewiring operations (i.e., $Z^{10}/Z^0 \leq 0.6$). In this case, it stops decreasing Z after 30 rewiring operations, which implies that only after a few rewiring operations it has found the best possible operations constrained by the threshold τ. On the graph with $d = 10$, our heuristic algorithm is able to decrease Z by nearly 80%, which is even larger than the case of $d = 5$. On the graph with the largest out-degree ($d = 20$), the algorithm is still effective but, as expected, achieves a comparable reduction in Z after 50 operations. The first baseline (**BSL-1**) shows almost the same solution quality as **HEU** since most of the operations found by both algorithms are the same. Although the rewiring operations provided by *RePBubLik* (**RBL**) also decrease the original Z_0 significantly, they are less effective than the ones given by our algorithm. When $d = 5$, it also reaches some steady states along the iterations, where the new rewiring operations do not decrease the Z value at all. The other baseline (**BSL-2**) and the random solution (**RND**) do not produce substantial decreases over the initial Z_0.

5 Conclusion

In this paper, we studied the problem of reducing the risk of radicalization pathways in *what-to-watch-next* recommenders via edge rewiring on the recommendation graph. We formally defined the segregation score of a radicalized node to measure its potential to trap users into radicalization pathways. We formulated the k-REWIRING problem to minimize the maximum segregation score among all radicalized nodes while maintaining the quality of the recommendations. We proposed an efficient yet effective greedy algorithm for k-REWIRING based on the absorbing random walk theory. Our experimental results in the context of video recommendations confirmed the effectiveness of our proposed algorithm.

Ethical Statement

In this work, we aim to reduce the exposure to radicalized content generated by W2W recommender systems. Our approach does not include any form of censorship and instead limits algorithmic-induced over-exposure, which is stimulated by biased organic interactions (e.g., the spread of radicalized content through user-user interactions). Our work contributes to raising awareness on the importance of devising policies to reduce harmful algorithmic side effects. Generally, we do not foresee any immediate or direct harmful impacts from this work.

Acknowledgements

Francesco Fabbri was a fellow of Eurecat's *Vicente López* Ph.D. grant program when he finished this research; his work was partially financially supported by the Catalan Government through the funding grant *ACCIÓ-Eurecat* (Project *PRIVany-nom*). Yanhao Wang has been supported by the National Natural Science Foundation of China under grant number 62202169. Francesco Bonchi acknowledges support from Intesa Sanpaolo Innovation Center. Carlos Castillo has been partially supported by the *HUMAINT* programme (Human Behaviour and Machine Intelligence), European Commission, and by *"la Caixa"* Foundation (ID 100010434), under agreement LCF/PR/PR16/51110009. Michael Mathioudakis has been supported by the University of Helsinki and Academy of Finland Projects MLDB (322046) and HPC-HD (347747).

The funders had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[Biega *et al.*, 2018] Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. Equity of attention: Amortizing individual fairness in rankings. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 405–414, 2018.

[Cinus *et al.*, 2022] Federico Cinus, Marco Minici, Corrado Monti, and Francesco Bonchi. The effect of people recommenders on echo chambers and polarization. *Proceedings of the International AAAI Conference on Web and Social Media*, 16(1):90–101, 2022.

[Fabbri *et al.*, 2022] Francesco Fabbri, Yanhao Wang, Francesco Bonchi, Carlos Castillo, and Michael Mathioudakis. Rewiring what-to-watch-next recommendations to reduce radicalization pathways. In *Proceedings of the ACM Web Conference 2022*, pages 2719–2728, 2022.

[Haddadan *et al.*, 2021] Shahrzad Haddadan, Cristina Menghini, Matteo Riondato, and Eli Upfal. RePBubLik: Reducing polarized bubble radius with link insertions. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 139–147, 2021.

[Haddadan *et al.*, 2022] Shahrzad Haddadan, Cristina Menghini, Matteo Riondato, and Eli Upfal. Reducing polarization and increasing diverse navigability in graphs by inserting edges and swapping edge weights. *Data Min. Knowl. Discov.*, 36(6):2334–2378, 2022.

[Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, 2008.

[Järvelin and Kekäläinen, 2002] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, 2002.

[Ledwich and Zaitsev, 2020] Mark Ledwich and Anna Zaitsev. Algorithmic extremism: Examining YouTube's rabbit hole of radicalization. *First Monday*, 25(3), 2020.

[Lewis, 2018] Rebecca Lewis. Alternative influence: Broadcasting the reactionary right on YouTube. Technical report, Data & Society Research Institute, September 2018.

[Mavroforakis *et al.*, 2015] Charalampos Mavroforakis, Michael Mathioudakis, and Aristides Gionis. Absorbing random-walk centrality: Theory and algorithms. In *Proceedings of the 2015 IEEE International Conference on Data Mining*, pages 901–906, 2015.

[McCauley and Moskalenko, 2008] Clark McCauley and Sophia Moskalenko. Mechanisms of political radicalization: Pathways toward terrorism. *Terror. Political Violence*, 20(3):415–433, 2008.

[Ribeiro *et al.*, 2020] Manoel Horta Ribeiro, Raphael Ottoni, Robert West, Virgílio A. F. Almeida, and Wagner Meira Jr. Auditing radicalization pathways on YouTube. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 131–141, 2020.

[Roose, 2019] Kevin Roose. The making of a YouTube radical. *The New York Times*, 2019.

[Weiss and Winter, 2018] Bari Weiss and Damon Winter. Meet the renegades of the intellectual dark web. *The New York Times*, 2018.

[Zhao *et al.*, 2019] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed H. Chi. Recommending what video to watch next: A multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 43–51, 2019.