Algorithmic Collective Action in Recommender Systems: Promoting Songs by Reordering Playlists

Joachim Baumann[*]
University of Zurich
baumann@ifi.uzh.ch

Celestine Mendler-Dünner
ELLIS Institute, Tübingen
MPI for Intelligent Systems, Tübingen
Tübingen AI Center
celestine@tue.ellis.eu

Abstract

We investigate algorithmic collective action in transformer-based recommender systems. Our use case is a collective of fans aiming to promote the visibility of an underrepresented artist by strategically placing one of their songs in the existing playlists they control. We introduce two easily implementable strategies to select the position at which to insert the song and boost recommendations at test time. The strategies exploit statistical properties of the learner to leverage discontinuities in the recommendations, and the long-tail nature of song distributions. We evaluate the efficacy of our strategies using a publicly available recommender system model released by a major music streaming platform. Our findings reveal that even small collectives (controlling less than 0.01% of the training data) can achieve up to 40× more test time recommendations than songs with similar training set occurrences, on average. Focusing on the externalities of the strategy, we find that the recommendations of other songs are largely preserved, and the newly gained recommendations are distributed across various artists. Together, our findings demonstrate how carefully designed collective action strategies can be effective while not necessarily being adversarial.

1 Introduction

In the ever-evolving landscape of music discovery, the challenge of accessing and sifting through the overwhelming number of tracks released daily has become increasingly difficult. This has resulted in a strong dependence on platforms like Spotify, Deezer, and Apple Music, which distribute and promote music through algorithmic song recommendations. These systems rely on historical data to learn user preferences and predict future content consumption [21, 51, 34, 7, 6].

It has been widely documented that music recommendation systems suffer from popularity bias as they tend to concentrate recommendation exposure on a limited fraction of artists, often overlooking new and emerging talent [35, 4, 2, 15, 9, 28]. As the success and visibility of artists are deeply influenced by the algorithms of these platforms, this can lead to a considerable imbalance in the music industry [1, 41] and reinforce existing inequalities [50]. Thus, artists have started to fight for more transparency and fairer payments from online streaming services. The “Justice at Spotify” campaign, launched by the Union of Musicians and Allied Workers [54], has been signed by more than 28,000 artists. At the same time, the International Society for Music Information Retrieval has been arguing for promoting the discovery of less popular artists by recommending ‘long-tail’ items [3], as have other researchers [11, 53, 16, 40].

[*] Work completed while at the Max-Planck Institute for Intelligent Systems, Tübingen.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

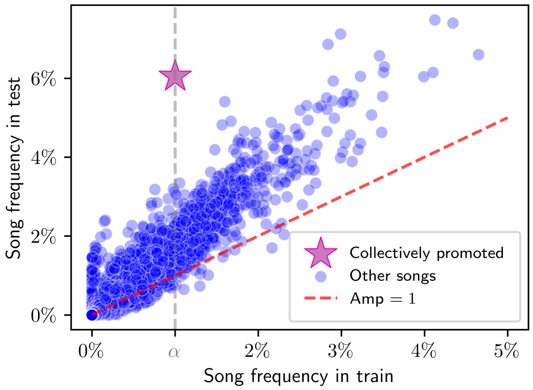


Figure 1: By strategically choosing the position at which to insert the target song in a playlist, collectives can achieve a disproportionally high recommendation frequency relative to training set occurrences, compared to naturally occurring songs. Amplification of one corresponds to matching frequencies in train and test.

In this work, we explore algorithmic collective action as an alternative means for emerging artists to gain exposure in machine learning-powered recommender systems by mobilizing their fan base. Algorithmic collective action [24] refers to the coordinated effort of a group of platform participants who strategically report the part of the training data they control to influence prediction outcomes. Our work is situated in an emerging literature that recognizes data as a powerful lever for users to promote their interests on digital platforms [57, 24].

1.1 Our work

We study algorithmic collective action in transformer-based recommender systems. As a case study, we consider the task of automatic playlist continuation (APC), which is at the heart of many music streaming services. APC models take a seed playlist (an ordered list of unique songs) as input and recommend songs to follow. They are trained on the universe of playlists stored on the platform. The collective consists of platform users who can modify the subset of playlists they own. The goal of the collective is to promote a less popular artist by increasing the recommendations of their songs at test time. To this end, we consider collective action strategies where participants of the collective agree on a target song s^* to strategically place in their playlists.

We motivate and discuss two strategies to choose the position of s^* within any given playlist. Both strategies are derived from a statistical optimality assumption on the recommender and do not require knowledge of the specifics of the model architecture or the model weights. Instead, they use that the model is trained to fit sequential patterns in existing data and build on aggregate song statistics that are feasible to gather from public information. We empirically test our strategies using an industry-scale APC model that has been deployed to provide recommendations for millions of users on Deezer—one of the biggest streaming platforms in the world. To train the model, we use the Spotify Million Playlist Dataset, treating each playlist as a user and randomly sampling a fraction to compose the collective.

We find that by strategically choosing the position of the target song, collectives can achieve significant over-representation at test time, see Figure 1 for a teaser. We experiment with collectives composed of a random sample of users owning between 0.001% and 2% of the training data instances. Interestingly, even tiny user collectives, controlling as few as 60 playlists, can achieve an amplification of up to $25\times$, referring to the song's recommendation frequency relative to the training frequency. This is $40\times$ more than an average song occurring at the same frequency in the training data. In contrast, placing the song in a fixed position in every playlist is largely ineffective.

Our strategy satisfies a strict authenticity constraint and thus preserves user experience at training time by design. Interestingly, we find that also at test time recommendations are largely preserved; not only on aggregate but also for members of the collective. As a consequence, the strategies come with small externalities for users, and at the same time, they also have a relatively small effect on model performance. For large collectives controlling $> 3\%$ of the playlists, the effect corresponds to every target song recommendation replacing an otherwise relevant song in less than 15% of the cases, leaving other recommendations unaltered. Thus, in the hypothetical case where the promoted song is indeed relevant, this could lead to an overall gain in more than 85% of the cases, even though the total number of test-time recommendations is fixed. Lastly, we show that the newly gained recommendations are taken from artists of diverse popularity without any indication that a specific artist suffers disproportionally.

Taken together, our work demonstrates a first example of collective action in sequential recommender systems. We show how collective action goals can be achieved while largely preserving service quality and user experience. The feasibility of such strategies raises many interesting questions, challenges, and opportunities for future work.

1.2 Related work

The fairness of recommendation systems on online platforms remains a pressing issue for both content consumers and producers [10, 32, 19, 26]—see Zehlike et al. [61] for a detailed overview. Several recent works study individual strategic users attempting to influence their own recommendations [5, 25, 12, 13]. Other works consider adding antidote data to fight polarization and unfairness [42, 18].

Beyond recommender systems, a related line of work centers the users in the study of machine learning systems. Vincent and Hecht [55] call for *conscious data contribution*, Vincent et al. [56] discuss data strikes, and Vincent et al. [57] emphasize the potential of data levers as a means to gain back power over platforms. Hardt et al. [24] introduce the framework of *algorithmic collective action* for formally studying coordinated strategies of users against algorithmic systems. They empirically demonstrate the effectiveness of collective action in correlating a signal function with a target label. Sigg et al. [46] inspect collective action at inference time in combinatorial systems. Complementing these findings, we demonstrate that collective action can be effective even without control over samples at inference time. We highlight a so far understudied dimension of algorithmic collective action by discussing and illuminating the externalities of algorithmic collective action strategies.

At a technical level, our findings most closely relate to *shilling attacks*, or more broadly, *data poisoning attacks* [c.f., 49]. Shilling attacks are usually realized by injecting fake user profiles and ratings in order to push the predictions of some targeted items [45, 47]. Due to the fraudulent nature of these attacks, there are little design restrictions on the profiles, and they often come with considerable negative effects for the firm [38, 20]. Data poisoning attacks in recommender systems predominantly focus on collaborative filtering-based models, with a few exceptions; Zhang et al. [62] propose a reinforcement learning-based framework to promote a target item, Yue et al. [58] provide a solution to extract a black-box model's weights through API queries to then generate fake users for promoting an item, and Yue et al. [59] propose injecting fake items into seemingly real item sequences (at inference time and without retraining) with a gradient-guided algorithm, requiring full access to the model weights. Taking the perspective of collective action, we focus on easy-to-implement strategies that require minimal knowledge of the model and operate under an authenticity constraint to preserve the utility of altered playlists while seamlessly integrating into natural interaction with the platform.

Further, our work pertains to a broader scholarly literature interested in improving labor conditions for gig workers on digital platforms [e.g., 29, 52], optimizing long-term social welfare in online systems [33], and understanding dynamics in digital marketplaces [27]. The type of strategic data modification we consider falls under the umbrella of *adversarial feature feedback loops* [39]. Taking advantage of collective strategies to change model outcomes more broadly has been studied in tabular data [17], computer vision [43], and recently in generative AI [44].

2 Preliminaries on automatic playlist continuation

We use automatic playlist continuation (APC) as a running example of a sequential recommendation task. APC forms the backbone of major streaming platforms, such as Spotify and Deezer. To formally define the recommendation task, let $\mathcal{S} = \{s_1, ..., s_n\}$ denote the universe of songs, where $n \geq 1$ denotes the number of unique songs. A playlist p is composed of an ordered list of songs selected from $\mathcal{S}$ without replacement. Given a seed playlist p, the firm's goal is to predict follow-up songs that the user likely listens to. We consider a top-K recommender system that outputs a personalized ordered list of $K \geq 1$ songs. We write $\mathrm{Rec}_K(p)$ for the set of K songs recommended for a seed playlist p.

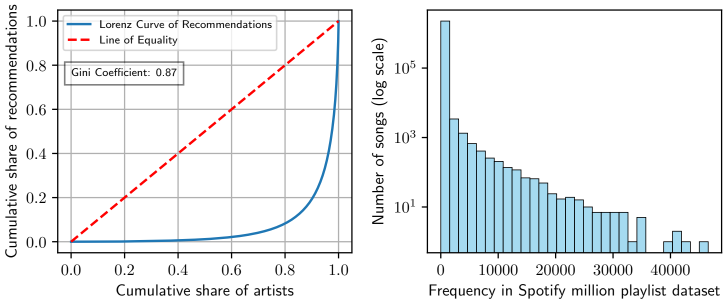


Figure 2: **Imbalance in recommendation distribution**. (left) The Lorenz curve shows that 80% of all recommendations are concentrated among just 10% of artists. (right) The Spotify track frequency distribution shows the long tail of song frequencies in user-generated playlists: close to 50% of tracks in playlists occur only once.

2.1 Transformer-based recommender

Over the past years, most large platforms have shifted from relying on collaborative filtering-based models for APC to building deep learning-based recommenders that account for sequential and session-based information [21, 51, 34]. In this work, we focus on transformer-based recommender systems that posit the following structure: Each song s is mapped to a song embedding vector $h_s = \phi(s)$, where ϕ denotes the embedding function. Each playlist $p = [s_1, s_2, ..., s_L]$ is mapped to an embedding vector h_p by aggregating the embeddings of the songs contained in the playlist as $h_p = g(h_{s_1}, h_{s_2}, ..., h_{s_L})$, where g is a sequence-aware attention function. We assume all playlists have length L smaller than the attention window for the purpose of exposition. At inference time, the recommendation of the next K songs for a given playlist p is determined by evaluating the similarity between the playlist seed p and all potential follow up songs $s \in \mathcal{S} \setminus p$ as

$$\text{SIM}(s, p) := \langle h_s, h_p \rangle. \tag{1}$$

Then, the K songs with the largest similarity value are recommended in descending order of similarity. We denote the set of recommended songs as

$$\text{Rec}_K(p) = \underset{S' \subseteq \mathcal{S} \setminus p : |S'| = K}{\arg\max} \sum_{s \in S'} \text{SIM}(s, p). \tag{2}$$

The embeddings ϕ and the attention function g are parameterized by neural networks. They are trained from existing user-generated playlists in a self-supervised manner by repeatedly splitting playlists into a seed context and a target sequence and employing a contrastive loss function for training.

Statistical abstraction. We do not assume the collective has knowledge of the parameters of either ϕ or g. Instead, the design of the strategy builds on the assumption that sequential, transformer-based models are trained such that $\text{SIM}(s, p)$ is large for songs s that frequently follow context p in the training data, and small otherwise. This approximately is robust to nuances in hyperparameter choices or architecture design and applies to any sufficiently expressive and well trained model.

2.2 Typical imbalances in recommendations

On today's music streaming platforms, a small number of artists receive the vast majority of recommendations, while the majority receive few or none. This imbalance is illustrated by the Lorenz curve in Figure 2, which is based on recommendations derived from the Deezer model on the Spotify MPD dataset (see Section 4). The Gini coefficient measuring inequality corresponds to 0.87. Streaming and radio statistics reveal an even more severe imbalance: the top 1% of newly released songs receive 99.99% of radio plays and 90% of streams go to just 1% of artists [9].

Considering Figure 1 we can also see that songs with high prevalence in the training data are recommended disproportionately often at test time compared to their training set frequency (referring to the slope of $\sim$1.8 of the blue point cloud). This gain in exposure through the recommender comes at the expense of many low-frequency songs that receive no recommendations at test time, further amplifying existing imbalances. Considering Spotify's substantial power to influence song consumption among platform users [1], withholding initial exposure for these songs limits their potential to reach a broader audience, significantly impacting an artist's career. In this work, we focus on collective efforts to boost recommendations for one of these underrepresented songs.

3 Algorithmic collective action for promoting songs

We build on the framework of Hardt et al. [24] and consider collectives that are composed of a fraction $\alpha \in (0, 1]$ of randomly sampled users on the platform. We assume each user controls a single playlist. Members of the collective can strategically manipulate their playlists. We use $\mu(\cdot)$ to describe the strategy of mapping an original playlist to a modified playlist.

Success and amplification. Let $\mathcal{P}_0$ denote the distribution over playlists. The goal of the collective is to increase the exposure of a target song s^* for a randomly sampled playlist from $\mathcal{P}_0$ at test time. We measure the success of collective action as

$$S(\alpha) := \mathrm{E}_{p \sim \mathcal{P}_0} \, 1 \left[s^* \in \mathrm{Rec}_K(p) \right]. \tag{3}$$

The recommender system Rec_K is trained on a partially manipulated training dataset $\mathcal{D}$, composed of N samples from $\mathcal{P}_0$, among which αN have been transformed under μ.

We are particularly interested in measuring the effectiveness of a strategy relative to the effort of the collective. Therefore, we define *amplification* (Amp) as the fraction of newly gained target recommendations at test time divided by the fraction of manipulated playlists in the training set:

$$\mathrm{Amp}(\alpha) = \frac{1}{\alpha}(S(\alpha) - S(0)) \tag{4}$$

An amplification of 0 means that the strategy is ineffective, an amplification of 1 means that the song frequency in the training set is proportionally represented in the model's predictions, and an amplification larger than 1 means that collective action achieves a disproportionate influence on the recommender. In the following, we choose a song s^* that does not currently appear in the training data, hence $S(0) = 0$.

3.1 Authenticity constraint

Participants of the collective are users of the platform. We design collective action strategies under the following authenticity constraint, not to compromise user experience:

Definition 1 (Authenticity constraint)**.** We say a strategy $\mu : p \to p'$ is authentic iff the Levenshtein distance between p and $p' = \mu(p)$ satisfies $\mathrm{Lev}(p, p') \leq 1$ for any p.

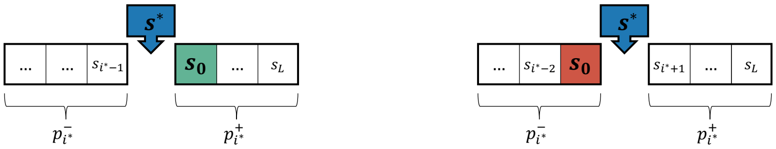
The Levhenstein distance [30], also known as edit distance in information theory, counts the number of operations needed to transform one sequence into another. The song insertion strategy we propose in this work is one concrete instantiation of μ that satisfies this constraint. More specifically, our strategy consists of inserting an agreed-upon target song s^* at a specific position in every playlist p. In contrast, existing adversarial strategies typically perform larger modifications to playlists and would not satisfy this constraint [62, 58].

3.2 Algorithmic lever

The algorithmic lever of the collective is to strategically choose, for each playlist, the position i^* at which to insert the target song. Under our probabilistic assumption about the learner, strategically placing s^* after p means that the similarity between p and s^* is increased. Thus, by choosing the index i^*, the collective targets context $p_{i^*}^-$ referring to the sequence of songs s_j in p up to index $j = i$. Recall that the collective aims to be among the top K songs with high frequency over a randomly sampled context p at test time. Our strategies exploit two different algorithmic levers towards this goal: Indirectly targeting Clusters of similar contexts (`InClust`) or Directly targeting Low-Frequency contexts (`DirLoF`). Pseudocode for the two strategies can be found in Figure 3.

Strategies: (a) `InClust` and (b) `DirLoF`

```
Input: $s^*$, collectively owned playlists $D^* = \{p_1, p_2, ..., p_n\} \subseteq D$
Coordination step:
(a) $r^s \leftarrow$ for every $s$ in $D^*$ pool information to count song frequencies in $D^*$.
(b) $q^s \leftarrow$ for every $s$ in $D^*$ estimate training set song frequency by gathering side information.
for all playlists $p \in D^*$ do
    Define anchor $s_0$: find song $s_0 \in p$ such that (a) $r^{s_0} \geq r^s \; \forall s \in p$ or (b) $q^{s_0} \leq q^s \; \forall s \in p$
    Insert target song: insert $s^*$ (a) before $s_0$ or (b) after $s_0$
    Store modified playlist
end for
```



(a) `InClust` (b) `DirLoF`

Figure 3: Song insertion strategies, pseudocode and illustration.

Concentrating effort. Inclusion in the set $\text{Rec}_K(p)$ leads to a song's recommendation for context p at test time. In turn, being ranked in position $K + 1$ does not yield any recommendations. Instead, the probability mass in the tails is reallocated to the top K songs at test time. This discontinuity can be exploited by the members of the collective to target specific contexts in a coordinated fashion to increase the likelihood of inclusion. Compared to random song placement, the collective can increase the mass on a particular context by a factor of L. The `InClust` strategy implements a way for selecting contexts to target, projecting this intuition from the non-parametric setting to the embedding space of the recommender. Namely, it systematically places s^* directly *before* each occurrence of a popular song s_0. In that way, it targets the region in the context embedding space around h_{s_0} in a coordinated fashion. To implement this strategy, the collective repeatedly determines the most frequent song in their playlists, places s^* before every occurrence of this song, and then repeats this with the remaining playlists until all of them are used.

Strategically exploiting overrepresentation. An alternative lever the collective has available is to strategically target contexts that are overrepresented among the playlists the collective controls. Meaning that the frequency of the context among the playlists owned by the collective is larger than the overall frequency in the training data due to finite sample artifacts. The `DirLoF` strategy aims to identify such contexts by targeting infrequently occurring songs and exploiting the long-tail nature of the overall song frequency distribution (see Figure 2). The core intuition is that if they manage to target low-frequency contexts, a single song placement might be sufficient to overpower existing signals. To identify low-frequency contexts, the collective uses the frequency of the last song as a proxy. For each playlist, it selects the anchor songs s_0 with the smallest overall song frequency and places s^* right after s_0.

3.3 Obtaining song statistics

The `InClust` strategy targets high-frequency contexts, whereas the `DirLoF` strategy targets low-frequency contexts. However, there is an important difference when implementing the strategies. `InClust` can be implemented from only statistics obtained from the songs in playlists the collective owns; all it requires is participants to set up infrastructure for pooling this information, either through an app, an online service, or other means. In contrast, to effectively implement the `DirLoF` strategy, the collective needs statistics about the full training data to identify the songs that are least popular overall. However, they typically do not have direct access to this information. Instead, as a proxy, they can leverage publicly available user-generated playlists, which are often accessible through official APIs (e.g., Spotify). Additionally, scraping external data sources can provide supplementary information. We evaluate the use of scraped song streams in Section 4.2.

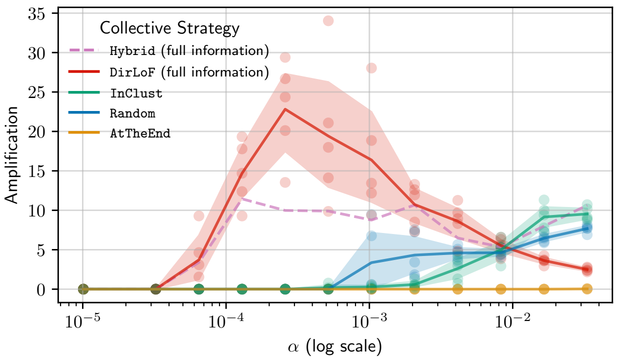


Figure 4: **Success of our collective action strategies.** For tiny collectives `DirLoF` achieves an amplification of up to 25× while uncoordinated strategies (`Random`, `AtTheEnd`) are mostly ineffective. For larger collectives, `InClust` outperforms `DirLoF`. Amplification significantly exceeds 1, implying a disproportional test-time effect due to targeted song placement.

4 Empirical evaluation

We evaluate our collective action strategies against a public version of Deezer's transformers-based APC solution that "has been deployed to all users" [7, p. 472]. To train the model, we use the Spotify Million Playlist Dataset (MPD), which is currently the largest public dataset for APC [14]. It contains one million playlists generated by US Spotify users between 2010 and 2017, with an average length of 66.35 tracks from nearly 300,000 unique artists.

Model training and evaluation. We use the standard methodologies used in APC for model training and testing.[2] We start by randomly selecting 20,000 playlists to build a test and validation set of equal sizes. The remaining 980,000 playlists are used for training the model. The collective intervenes by strategically modifying an α fraction of the playlists composing the training and validation set. We consider collectives of size $\alpha \in [0.00001, 0.02]$ which corresponds to 10 to 20000 playlists. For evaluation on the test set, every playlist p is split into a seed context and a masked end. The length of the seed context is chosen randomly in $[1, 10]$ for each playlist and models are evaluated by comparing the model's recommendations based on the seed playlist to the masked ground truth. We employ five-fold cross-validation, using different random seeds for sampling the playlists designated for training, validation, and testing, as well as for selecting the subset controlled by the collective. We use bootstrapped 95% confidence intervals (CI) over folds when reporting results.

Baselines. We consider four baseline strategies to compare with our collective strategies, each performing the same number of target song insertions. The `Random` strategy inserts s^* at a `Random` position in the playlist, `Insert@`i inserts the song always at position i in every playlist, the `AtTheEnd` strategy places s^* as the last song of the playlist, and `Random@`i-j inserts s^* at a random position between indices i and j. Unlike our collective strategies, these baselines do not require coordination among participants beyond the shared goal of promoting s^*.

4.1 Success of collective action

We start by evaluating the success of the proposed strategies, assuming full information about song frequencies in the training set to illustrate the potential. In Figure 4, we plot the amplification for different α. In particular, we observe that strategic song placement allows very small collectives ($\alpha \leq 0.1\%$) to be successful, whereas `Random` or fixed placement of s^* is ineffective.

For $\alpha = 0.025\%$, the `DirLoF` strategy achieves amplification of up to 25. In contrast to an average song that naturally occurs in 0.025% of the playlists, the number of recommendations is 40× larger, as these low-frequency songs are typically ignored by the recommender. This suggests that collective action could make a tremendous difference for these artists: suppose an artist's song is streamed

[2] The code is available at `https://github.com/joebaumann/recsys-collectiveaction`.

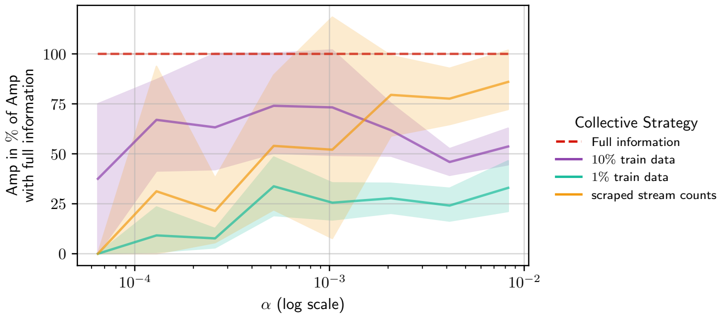


Figure 5: **Information bottleneck.** The empirical amplification of the DirLoF strategy decreases with worse song statistics but scraped song streaming counts can serve as a practical solution.

10,000 times, yielding a revenue of \$40 at a royalty rate of $\$0.004$ per stream [31]; an amplification of 25 would hypothetically increase this revenue to $\$1,000$. While this example is purely illustrative (as actual royalties depend on the platform and payment model used), it emphasizes the link between recommendations and potential revenue.

For collective sizes of $\alpha \geq 0.1\%$ the InClust starts being effective, as it has enough mass to effectively compete with existing signals associated with a cluster of similar context embeddings. As the strategy can target several such clusters at the same time, amplification increases with α though with diminishing returns, achieving $\mathrm{Amp} = 10$ for $\alpha \approx 2\%$. From Figure 1, we can see that in the regime of 2% training data frequency, a typical song enjoys an amplification of 1.8.

We also observe that the success of the random strategy increases with the collective size. This implies that even minimal coordination, in which members agree to all insert the same song s^*, *independent* of the playlist they own, can already lead to significant amplification. Amplification values for the other baselines inserting s^* at a fixed position are all close to 0 (see Table 1 in Appendix C.4).

Robustness to hyperparameters. Our strategies are designed based on a statistical intuition of sequential generation and should not be sensitive to specifics of the model architecture. We demonstrate the robustness with additional experiments where we vary the hyperparameters of the model (see Table 2 in Appendix C.5). However, the design of our strategies relies on the assumption that the model approximates the conditional probabilities in the training data sufficiently well. Accordingly, the effectiveness of the strategy decreases if model training is stopped early (see Table 3).

Hybrid **strategy.** Building on these observations, we construct a hybrid strategy that interleaves the two approaches by first using InClust to target indirect anchors that appear at least λ times in the collective and then deviates to DirLoF for playlists where no such anchor is present (we use $\lambda = 10$). This corresponds to the dashed line in Figure 4. We come back to this strategy in Section 4.3.

4.2 DirLoF strategy with approximate song statistics

The DirLoF strategy critically relies on training data song frequency estimates to determine the low-frequency anchor songs. We investigate the strategy's success with partially available song information in Figure 5. We find that if a collective of size $\alpha = 1\%$ has access to 1% of the remaining training data they do not control, they can already achieve $\approx 30\%$ of the amplification in the full information setting, with 10% of the data, it is $> 50\%$ of the achievable amplification.

By default, user-generated playlists on streaming platforms are often publicly accessible, enabling researchers to gather song frequency data through API calls. However, the amount of training data that can be aggregated is limited by the platform's API rate limits. Alternatively, proxy statistics can be used to increase the fraction of songs for which estimates are available. To illustrate the feasibility of this approach, we implemented a scraper to obtain current stream counts from Spotify. Although these counts are visible in the Spotify browser version, they are not accessible through the Spotify API. The scraped data reflects song popularity as of 2024, which is not ideal given our experiment relies on the much older Spotify MPD dataset (collected between 2010 and 2017). Nonetheless, these counts serve as effective proxies, as Figure 5 impressively shows.

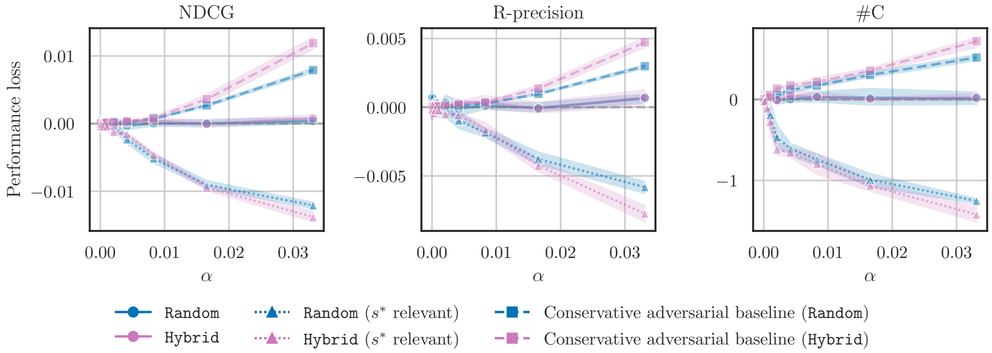


Figure 6: Effect of algorithmic collective action on recommendation performance. Performance loss relative to training on clean data for the `hybrid` / `random` strategies (solid lines), a conservative adversarial baseline (dashed lines), and an optimistic scenario where s^* is treated as relevant (dotted lines).

Despite the temporal gap, a collective of size $\alpha = 1\%$ can achieve over 85% of the amplification achievable in a full-information setting simply by using 2024 stream data to approximate past popularity levels. Even a smaller collective of $\alpha = 0.1\%$ can reach about 50% of the amplification seen in the full-information scenario. In practice, scraped stream counts are likely to be more accurate proxies, as models are typically trained on more recent data. However, within the scope of our study, it remains impossible to access historical stream counts that would reflect popularity as of the time the playlists were originally generated. Thus our proof of concept should be seen as a lower bound.

4.3 Internalities and externalities of algorithmic collective action

We now inspect the effect of our strategies on other participants in the system, including the firm, other artists, and the members of the collective. For this investigation, we focus on the hybrid strategy.

First, we gauge the impact of collective action on the firm. This helps us understand the overall quality degradation of the service and the incentives of the firm to protect against collective action. We compare the performance under a recommender trained on the clean data and a recommender trained on the manipulated data. Figure 6 shows the corresponding loss in performance due to collective action for three different evaluation metrics. We find that our strategy (solid lines) only affects the recommender's performance marginally. We also show a conservative adversarial baseline (dashed lines), which simulates a scenario where successful collective action results in the first relevant item in playlist recommendations being replaced by the target song while leaving other recommendations unaltered. The considerably larger performance loss of this baseline indicates that our strategy only rarely affects relevant songs. Finally, as a thought experiment, consider s^* as a relevant recommendation (dotted lines). Then, collective action even enhances the system's performance. This reference is meant to illustrate an optimistic scenario where collective action helps the recommender detect underrepresented but emerging and popular artists.

Second, we inspect the effect of collective action on other artists. To this end, Figure 7 depicts the change in recommendations for individual songs of different popularity. Songs are binned by frequency and the bars indicate variation across songs. The star shows the target song s^*, and the corresponding increase in recommendations. We see that recommendations replaced by the target song seem to span songs of all popularity levels. In particular, our strategy does not harm specific songs or artists disproportionally and, as intended, has by far the largest effect on the targeted song s^*.

Finally, we focus on the experience for participants who listen to the playlists. At training time our strategies are designed to only ask for minimal modifications with the goal to preserve user experience for members of the collective. We envision this to be an important factor for incentivizing participation in practice. Non-participating individuals are not affected at this stage. At test time, we find that user recommendations are largely preserved for both participating and non-participating individuals. More precisely, participating in collective action does not deteriorate the fraction of relevant songs participants get recommended, i.e., performance remains equivalent across all three recommendation quality metrics (see Figure 13 in Appendix C.6).

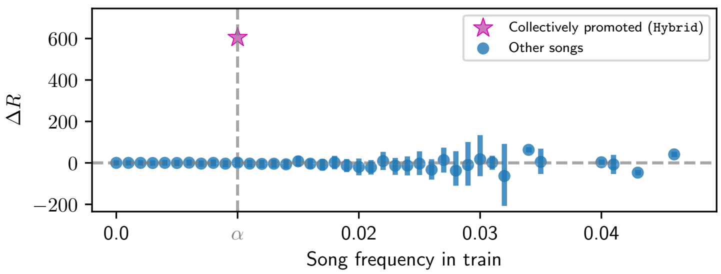


Figure 7: Impact of collective action on other songs. We use $\alpha = 1\%$ to obtain an upper bound on the effect. ΔR denotes the change in the number of recommendations for a song due to collective action. Songs are sorted by their training set frequency and aggregated into 50 evenly spaced bins, whose means are represented by the blue dots with 95% CI.

5 Conclusion

This work studies how collective action strategies can empower participants to exert a targeted influence on platform-deployed algorithms. By experimenting with an industry-scale transformer-based APC model, we demonstrate how strategically inserting a single song within randomly sampled playlists in the training data, can effectively increase recommendations of that song. Intriguingly, we find that the strategy only minimally interferes with service quality, and the recommendations for other users on the platform are largely preserved.

The proposed concept of participating in collective action to steer recommender system outcomes is grounded in the idea that users on online platforms should leave their digital traces more consciously. Thereby, their consumption behavior functions as a lever to reclaim some control over the data that platforms use to predict and recommend future content. Our emphasis on authenticity stands in clear contrast to adversarial machine learning techniques, which are often artificially designed and sometimes malicious in intent.

While altering a single playlist alone has little impact, the true power of algorithmic collective action lies in mobilizing a sufficiently large number of participants around a shared objective. This allows underrepresented artists to gain visibility through coordination. In our case, coordination corresponds to agreeing on a target song and an insertion procedure. The actual implementation of the strategy is possible with very limited technical skills and knowledge of the algorithm. We demonstrate how information for setting the parameters of the strategy can effectively be gathered using web scraping techniques. What we leave for future work is the actual implementation of an app to orchestrate collective action and share all the relevant information with the participants.

Our work suggests a widely unexplored design space for effective collective action strategies that differ from typical adversarial data poisoning attacks [c.f. 49, 62, 58, 59]. They can offer a powerful data lever to counter existing power imbalances [56, 57], and a community-centric approach to participatory AI [8]. Thus, understanding the role of economic power [23, 22], formalizing incentives [37], as well as quantifying long-term payoffs, dynamics, and equilibria, under collective action promises to be a fruitful direction for future work.

6 Limitations and potential for misuse

Grounding algorithmic collective action means identifying both its opportunities and challenges. The power that arises from gaining control over the learning algorithm through collective action can also be abused by individuals controlling a substantial number of playlists. Instead of collective goals, these individuals could leverage similar methods to pursue individualistic goals, creating a different incentive structure and potentially posing a risk to the system. Similarly, popular artists could use our strategy to gain additional exposure and reinforce inequalities among artists. Thus, incentive structures will crucially determine the desirability of the resulting market outcome. Designing larger-scale collective action strategies that promote fairness and equity on online platforms as well as mechanisms that disincentivize malicious use remains a crucial open question.

Acknowledgements

We would like to thank Moritz Hardt for many insightful and formative discussions throughout the course of this work. We would also like to thank Mila Gorecki, Ricardo Dominguez-Olmedo, Ana-Andreea Stoica and André Cruz for invaluable feedback on the manuscript, and Olawale Salaudeen, Florian Dorner, Stefania Ionescu and Tijana Zrnic for helpful feedback on earlier versions of this work. Celestine Mendler-Dünner acknowledges financial support from the Hector foundation.

References

[1] L. Aguiar and J. Waldfogel. Platforms, power, and promotion: Evidence from spotify playlists. *The Journal of Industrial Economics*, 69(3):653–691, 2021.

[2] C. Bauer. Allowing for equal opportunities for artists in music recommendation. In *Proceedings of the 1st Workshop on Human-Centric Music Information Research Systems*, pages 16–18, 2019.

[3] C. Bauer. Report on the ISMIR 2020 special session: how do we help artists? *ACM SIGIR Forum*, 54(2), 2020.

[4] C. Bauer, M. Kholodylo, and C. Strauss. Music recommender systems: challenges and opportunities for non-superstar artists. In *30th Bled eConference*, pages 21–32, 2017.

[5] O. Ben-Porat and M. Tennenholtz. A game-theoretic approach to recommendation systems with strategic content providers. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[6] W. Bendada, T. Bontempelli, M. Morlon, B. Chapus, T. Cador, T. Bouabça, and G. Salha-Galvan. Track Mix Generation on Music Streaming Services Using Transformers. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 112–115, 2023.

[7] W. Bendada, G. Salha-Galvan, T. Bouabça, and T. Cazenave. A Scalable Framework for Automatic Playlist Continuation on Music Streaming Services. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 464–474, 2023.

[8] A. Birhane, W. Isaac, V. Prabhakaran, M. Diaz, M. C. Elish, I. Gabriel, and S. Mohamed. Power to the people? opportunities and challenges for participatory ai. In *ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, 2022.

[9] E. Blake. Data shows 90 percent of streams go to the top 1 percent of artists, 2020. `https://www.rollingstone.com/pro/news/top-1-percent-streaming-1055005`.

[10] R. Burke. Multisided fairness for recommendation. *ArXiv preprint arXiv:1707.00093*, 2017.

[11] Ò. Celma. *Music Recommendation and Discovery: The Long Tail, Long Fail, and Long Play in the Digital Music Space*. Springer Berlin, Heidelberg, 2010.

[12] S. H. Cen, A. Ilyas, J. Allen, H. Li, D. Rand, and A. Madry. Measuring strategization in recommendation: Users adapt their behavior to shape future content. *Arxiv preprint arXiv:2405.05596*, 2023.

[13] S. H. Cen, A. Ilyas, and A. Madry. User strategization and trustworthy algorithms. *ArXiv preprint arXiv:2312.17666*, 2023.

[14] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani. Recsys Challenge 2018: Automatic Music Playlist Continuation. In *ACM Conference on Recommender Systems*, pages 527–528, 2018.

[15] M. P. Coelho and J. Z. Mendes. Digital music and the "death of the long tail". *Journal of Business Research*, 101:454–460, 2019.

[16] S. Craw, B. Horsburgh, and S. Massie. Music recommendation: Audio neighbourhoods to discover music in the long tail. In *Case-Based Reasoning Research and Development*, pages 73–87. Springer International Publishing, 2015.

[17] E. Creager and R. Zemel. Online algorithmic recourse by collective action. *ICML Workshop on Algorithmic Recourse*, 2023.

[18] M. Fang, J. Liu, M. Momma, and Y. Sun. Fairroad: Achieving fairness for recommender systems with optimized antidote data. In *ACM Symposium on Access Control Models and Technologies*, page 173–184, 2022.

[19] A. Ferraro, X. Serra, and C. Bauer. What is fair? exploring the artists' perspective on the fairness of music streaming platforms. In *Human-Computer Interaction*, volume 12933, pages 562–584, 2021.

[20] I. Gunes, C. Kaleli, A. Bilge, and H. Polat. Shilling attacks against recommender systems: a comprehensive survey. *Artificial Intelligence Review*, 42:767–799, 2014.

[21] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas. Contextual and sequential user embeddings for large-scale music recommendation. In *ACM Conference on Recommender Systems*, page 53–62, 2020.

[22] M. Hardt and C. Mendler-Dünner. Performative prediction: Past and future. *ArXiv preprint arXiv:2310.16608*, 2023.

[23] M. Hardt, M. Jagadeesan, and C. Mendler-Dünner. Performative Power. In *Advances in Neural Information Processing Systems*, 2022.

[24] M. Hardt, E. Mazumdar, C. Mendler-Dünner, and T. Zrnic. Algorithmic Collective Action in Machine Learning. In *International Conference on Machine Learning*, volume 202, pages 12570–12586, 2023.

[25] A. Haupt, D. Hadfield-Menell, and C. Podimata. Recommending to strategic users. *ArXiv preprint arXiv:2302.06559*, 2023.

[26] S. Ionescu, A. Hannak, and N. Pagan. Group fairness for content creators: the role of human and algorithmic biases under popularity-based recommendations. In *ACM Conference on Recommender Systems*, page 863–870, 2023.

[27] M. Jagadeesan, M. I. Jordan, and N. Haghtalab. Competition, alignment, and equilibria in digital marketplaces. *AAAI Conference on Artificial Intelligence*, 37(5):5689–5696, 2023.

[28] D. Jannach, L. Lerche, F. Gedikli, and G. Bonnin. What recommenders recommend – an analysis of accuracy, popularity, and sales diversity effects. In *User Modeling, Adaptation, and Personalization*, pages 25–37. Springer Berlin Heidelberg, 2013.

[29] M. H. Jarrahi and W. Sutherland. Algorithmic management and algorithmic competencies: Understanding and appropriating algorithms in gig work. In *Information in Contemporary Society*, pages 578–589, 2019.

[30] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.

[31] L. Marshall. 'let's keep music special. f—spotify': on-demand streaming and the controversy over artist royalties. *Creative Industries Journal*, 8(2):177–189, 2015.

[32] R. Mehrotra, J. McInerney, H. Bouchard, M. Lalmas, and F. Diaz. Towards a fair marketplace: Counterfactual evaluation of the trade-off between relevance, fairness & satisfaction in recommendation systems. In *ACM International Conference on Information and Knowledge Management*, page 2243–2251, 2018.

[33] M. Mladenov, E. Creager, O. Ben-Porat, K. Swersky, R. Zemel, and C. Boutilier. Optimizing long-term social welfare in recommender systems: a constrained matching approach. In *International Conference on Machine Learning*, 2020.

[34] D. Moor, Y. Yuan, R. Mehrotra, Z. Dai, and M. Lalmas. Exploiting sequential music preferences via optimisation-based sequencing. In *ACM International Conference on Information and Knowledge Management*, page 4759–4765, 2023.

[35] P. M. Napoli. Requiem for the long tail: Towards a political economy of content aggregation and fragmentation. *International Journal of Media & Cultural Politics*, 12(3):341–356, 2016.

[36] S. Oh, B. Ustun, J. McAuley, and S. Kumar. Rank list sensitivity of recommender systems to interaction perturbations. In *ACM International Conference on Information & Knowledge Management*, page 1584–1594, 2022.

[37] M. Olson. *The logic of collective action: public goods and the theory of groups.* Harvard University Press, 1965.

[38] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Trans. Internet Technol.*, 4(4):344–377, 2004.

[39] N. Pagan, J. Baumann, E. Elokda, G. De Pasquale, S. Bolognani, and A. Hannák. A classification of feedback loops and their relation to biases in automated decision-making systems. In *ACM Conference on Equity and Access in Algorithms, Mechanisms, and Optimization*, 2023.

[40] L. Porcaro, E. Gómez, and C. Castillo. Assessing the impact of music recommendation diversity on listeners: A longitudinal study. *ACM Trans. Recomm. Syst.*, 2(1), 2024.

[41] R. Prey, M. Esteve Del Valle, and L. Zwerwer. Platform pop: disentangling spotify's intermediary role in the music industry. *Information, Communication & Society*, 25(1):74–92, 2022.

[42] B. Rastegarpanah, K. P. Gummadi, and M. Crovella. Fighting fire with fire: Using antidote data to improve polarization and fairness of recommender systems. In *ACM International Conference on Web Search and Data Mining*, page 231–239, 2019.

[43] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[44] S. Shan, J. Cryan, E. Wenger, H. Zheng, R. Hanocka, and B. Y. Zhao. Glaze: Protecting artists from style mimicry by Text-to-Image models. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2187–2204, 2023.

[45] M. Si and Q. Li. Shilling attacks against collaborative recommender systems: a review. *Artificial Intelligence Review*, 53:291–319, 2020.

[46] D. Sigg, M. Hardt, and C. Mendler-Dünner. Decline now: A combinatorial model for algorithmic collective action. *ArXiv preprint arXiv:2410.12633*, 2024.

[47] A. P. Sundar, F. Li, X. Zou, T. Gao, and E. D. Russomanno. Understanding shilling attacks and their detection traits: A comprehensive survey. *IEEE Access*, 8:171703–171715, 2020.

[48] D. Thain, T. Tannenbaum, and M. Livny. Distributed computing in practice: the condor experience. *Concurrency and Computation: Practice and Experience*, 17(2-4):323–356, 2005.

[49] Z. Tian, L. Cui, J. Liang, and S. Yu. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Comput. Surv.*, 55(8), 2022.

[50] T. Tofalvy and J. Koltai. "Splendid Isolation": The reproduction of music industry inequalities in Spotify's recommendation system. *New Media & Society*, 25(7):1580–1604, 2023.

[51] F. Tomasi, J. Cauteruccio, S. Kanoria, K. Ciosek, M. Rinaldi, and Z. Dai. Automatic music playlist generation via simulation-based reinforcement learning. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 4948–4957, 2023.

[52] C. Toxtli and S. Savage. *Designing AI Tools to Address Power Imbalances in Digital Labor Platforms*, pages 121–137. Springer International Publishing, 2023.

[53] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. In *ISMIR*, 2008.

[54] Union of Musicians and Allied Workers. Justice at Spotify, mar 2021. URL `https://weareumaw.org/justice-at-spotify`.

[55] N. Vincent and B. Hecht. Can "Conscious Data Contribution" Help Users to Exert "Data Leverage" Against Technology Companies? *Proc. ACM Hum.-Comput. Interact.*, 5, 2021.

[56] N. Vincent, B. Hecht, and S. Sen. “Data Strikes”: Evaluating the Effectiveness of a New Form of Collective Action Against Technology Companies. In *The World Wide Web Conference*, pages 1931–1943, 2019.

[57] N. Vincent, H. Li, N. Tilly, S. Chancellor, and B. Hecht. Data Leverage: A Framework for Empowering the Public in Its Relationship with Technology Companies. In *ACM Conference on Fairness, Accountability, and Transparency*, pages 215–227, 2021.

[58] Z. Yue, Z. He, H. Zeng, and J. McAuley. Black-box attacks on sequential recommenders via data-free model extraction. In *ACM Conference on Recommender Systems*, page 44–54, 2021.

[59] Z. Yue, H. Zeng, Z. Kou, L. Shang, and D. Wang. Defending substitution-based profile pollution attacks on sequential recommenders. In *ACM Conference on Recommender Systems*, page 59–70, 2022.

[60] H. Zamani, M. Schedl, P. Lamere, and C.-W. Chen. An Analysis of Approaches Taken in the ACM RecSys Challenge 2018 for Automatic Music Playlist Continuation. *ACM Trans. Intell. Syst. Technol.*, 10(5), 2019.

[61] M. Zehlike, K. Yang, and J. Stoyanovich. Fairness in ranking, part II: Learning-to-rank and recommender systems. *ACM Comput. Surv.*, 55(6), 2022.

[62] H. Zhang, Y. Li, B. Ding, and J. Gao. Practical data poisoning attack against next-item recommendation. In *The Web Conference 2020*, page 2458–2464, 2020.

A Song recommendation inequality

Figure 8 visualizes the track-level distribution of algorithmic exposure with the cumulative share of recommendations (y-axis) plotted against the percentiles of tracks (x-axis). The recommendations are derived from the Deezer model [7] on the Spotify MPD dataset [14]. More precisely, they are based on the outputs generated for a random selection of 10,000 seed playlists for testing, produced by a model that has been trained on the remainder of the dataset, without any collective action—see Section 4 for more details. Similar to the artist-based Lorenz curve in Figure 2, we observe a very high level of inequality with a Gini coefficient of 0.8 (measuring the gap between the line of equality and the Lorenz curve).

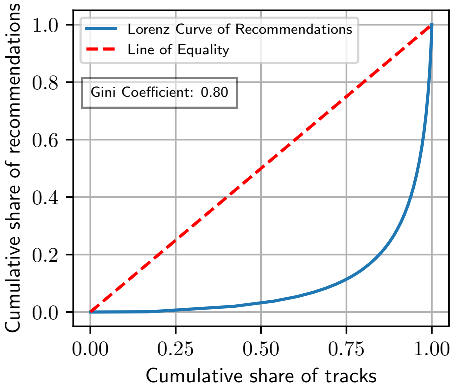


Figure 8: The Lorenz curve shows the unequal distribution of recommendations across tracks: 80% of all recommendations are concentrated among just 15% of tracks.

B Experimental details

All experiments were run as jobs submitted to a centralized cluster, using the open-source `HTCondor` job scheduler [48]. All jobs utilized the same computing resources: For data preprocessing and performing the data modifications as per a strategic collective action, 1 CPU was used with an allocated 100GB of RAM. In a subsequent step, transformer models were trained using a single NVIDIA A100-SXM4-80GB GPU. For each job, data preprocessing takes roughly 1-2 hours to complete (with coordinated strategies taking longer than uncoordinated ones). Models are trained for 18 epochs, using the optimal hyperparameters provided by Bendada et al. [7], which takes roughly 6 hours.

A total of 1195 experiments were run: We investigated 10 strategies (including 6 `DirLoF` strategies with varying levels of song statistics knowledge and 8 simple baselines, as well as 6 different hyperparameter configurations), across 12 collective sizes (α), and an additional baseline without collective action ($\alpha = 0$) as a reference for the main experiments. For the ablation study, 6 strategies were tested over 13 α values. Each experiment was conducted with five folds using different random seeds. This resulted in approximately 2390 CPU hours and 7170 GPU hours of total compute usage. The complete code is available at `https://github.com/joebaumann/recsys-collectiveaction`.

C Additional experiments

C.1 Ablation study for `InClust` strategy

We perform an additional empirical investigation to provide insights into the inner workings of the strategy. In particular, the effect of strategic positioning using indirect anchors on the attention function. Recall that the recommender is trained such that $\mathrm{SIM}(s, p) = \langle h_{s_0}, g(p) \rangle$ is large for songs s that frequently follow context p in the training data, and small otherwise. The embedding function ϕ (described in Section 2.1) is insensitive to the song ordering within playlists, and strategic positioning will only surface on the attention function g.

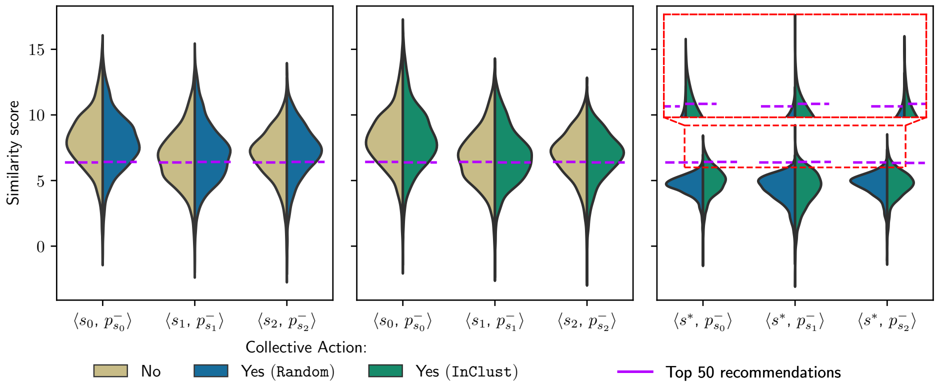


Figure 9: Similarities of context embeddings for indirect anchor songs (s_0, s_1, and s_2) and the target song (s^*) for $\alpha = 1.1\%$. Dashed purple lines represent average thresholds to be among the top 50 most similar songs.

For illustration, we use three indirect anchors corresponding to the three songs that occur most frequently in the playlists of the collective, denoted s_0, s_1, and s_2. We compare `InClust`, where we insert the target song s^* before these songs, with `Random`, where we add s^* in the same playlists but at random positions. This ensures we have the same pretrained song embeddings across the two strategies. Playlists that do not contain any of those three songs are not manipulated. We use $p_{s_i^-}$ to denote the context targeted by using s_i as an indirect anchor.

In Figure 9, we visualize the similarity scores of the songs (s^*, s_0, s_1, and s_2) with the three context clusters that have been targeted. More specifically, for every anchor song s_i, we use all seed contexts $p_{s_i^-}$ that have been targeted in the training data and compare the similarity score of these contexts with (yes) and without (no) collective action. This results in a distribution over scores, as visualized by the violin plot. The left and the middle panel show that the similarity scores of the anchor songs and their associated contexts are not altered for either of the two strategies (`Random` and `InClust`). Furthermore, as can be seen in the right panel, the `InCLust` strategy is very effective in getting the target song to be among the top 50 most similar tracks for the targeted contexts (corresponding to the probability mass above the purple threshold). In contrast, for the random placement that does not specifically target these contexts, s^* generally fails to achieve a ranking in the top 50.

Targetting multiple context clusters. The `InCLust` strategy is effective on all three distinct context clusters that are targeted simultaneously with a single target song s^*. We confirm the effectiveness of this strategy by assessing its success with respect to a test set, which contains a randomly drawn set of playlists (each split into a seed context and a ground truth) that have not been seen during training. Figure 10 shows that for any number of indirect anchors, the `InClust` strategy significantly outperforms the `Random` placement strategy. Furthermore, it also clearly shows that it is possible to effectively target multiple context clusters using the same target song. In conclusion, it is possible to compete with several context clusters simultaneously.

Overall, this ablation indicates that inserting a single song in a subset of playlists can be effective in associating s^* with specific contexts while preserving recommendations for songs $\mathcal{S} \backslash s^*$.

C.2 Information bottleneck

Figure 11 displays the 100 targeted direct anchors for $\alpha = 0.01\%$ under different levels of knowledge about the song frequencies in the training set. Less information results in the selection of more frequent songs, as the gap between the estimated probability (relative occurrences of songs in the known fraction of the dataset or estimated using external information) and the true probability (relative occurrences of songs in the entire training data) widens. This difference provides insight into the reduced amplification observed in Figure 5 for a specific value of α.

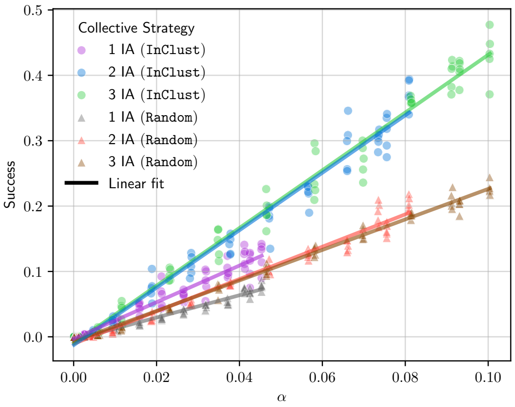


Figure 10: Success with respect to the number of used indirect anchors (IA), in random or coordinated fashion. Each dot or triangle corresponds to a separate training run.

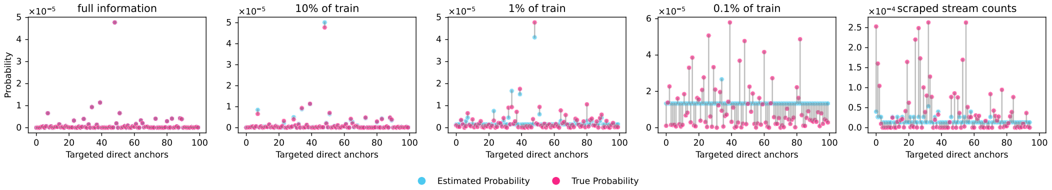


Figure 11: Estimated and true probabilities of targeted direct anchors with limited information about training set frequencies ($\alpha = 0.01\%$).

C.3 Songs targeted by different strategies

Figure 12 illustrates the anchor song selection for three distinct strategies. The `InClust` strategy identifies anchors based on their prevalence within the collective, targeting songs frequently listened to by its members. Highlighted in red in the left panel of Figure 12, this method repeatedly employs indirect anchors, relying solely on internal playlist statistics without needing broader song frequency insights. Conversely, the `DirLoF` strategy targets a specific cluster of direct anchors (resulting in the red-colored cluster of anchors in the bottom left corner of the middle panel in Figure 12) targeting each only once. This method requires external data to ensure the disproportionately represented anchors in the collective match those less prevalent in the training data, as shown on the x-axis. The `Hybrid` strategy, illustrated in the right panel of Figure 12, combines these approaches, targeting

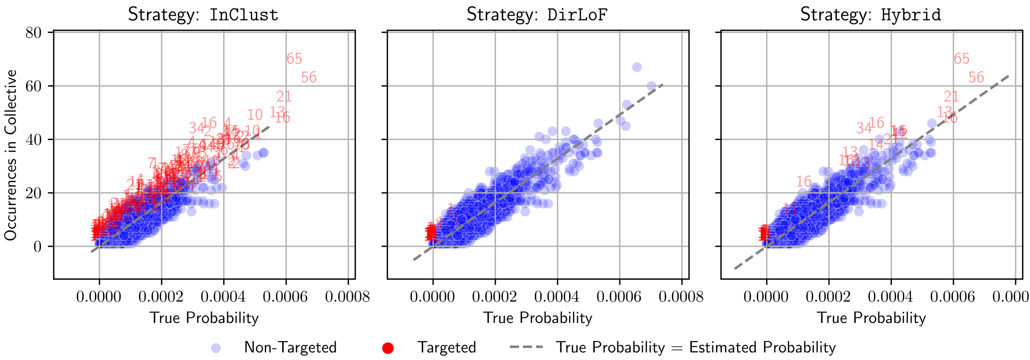


Figure 12: Songs in playlists controlled by a collective composed of 0.07% of the training data. Blue dots are songs that are not targeted. Red integers indicate a used anchor song and the number of times it is targeted.

Table 1: Mean Amplification (Std Dev) for additional baseline strategies that do not require coordination. Insert@i denotes the insertion of s^* at index i and Random@i-j denotes the insertion of s^* at a random index between i and j. The best and second-best performing strategies are highlighted in bold and underlined, respectively.

	α		
Strategy	0.0002	0.001	0.002
Insert@0	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Insert@1	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Insert@3	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Insert@5	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Insert@7	0.00 (0.00)	0.02 (0.04)	0.07 (0.08)
Random@1-10	0.00 (0.00)	0.00 (0.00)	0.03 (0.03)
Random	0.00 (0.00)	3.36 (4.45)	4.32 (2.90)
DirLoF	**22.82 (6.21)**	**16.38 (7.37)**	**10.73 (2.66)**

Table 2: Mean Amplification (Std Dev) under different hp configurations ($\alpha = 0.001$). hp* denotes the optimal set of hp reported by Bendada et al. [7] with 8 attention heads (n_heads), a learning rate (lr) of 1.0, a dropout rate (drop_p) of 0.13, a weight decay (wd) of 1.53e-05, and 18 epochs. We experiment with five alternative hp configurations that are equivalent to hp* except for the following changes: hp1 sets n_heads=4, hp2 sets lr=0.5, hp3 sets drop_p=0.2, hp4 sets wd=5e-05, and hp5 makes all four of these changes. The highest amplification values among hp configurations are highlighted in bold.

	Hyperparameter configuration					
Strategy	hp*	hp1	hp2	hp3	hp4	hp5
Random	3.36 (4.45)	1.12 (2.15)	0.23 (0.52)	2.63 (5.67)	2.69 (2.96)	2.69 (3.41)
DirLoF	**16.38 (7.37)**	**18.47 (9.03)**	**21.77 (1.67)**	**16.13 (10.07)**	**26.32 (4.38)**	**27.92 (5.53)**

both frequently occurring songs within the collective and low-frequency anchors. The parameter λ governs the fraction of the collective targeting indirect anchors (as in the left panel) versus targeting direct anchors (as in the middle panel). Larger λ values result in a larger share of direct anchors being targeted, and smaller values result in keeping more of the frequently targeted indirect anchors, i.e., the large red integers visualized toward the top of the left panel. Note that all strategies target several anchors, ensuring all playlists can be used effectively.

C.4 Additional baselines

Table 1 demonstrates that simple baselines, which insert s^* at a fixed position, are much less effective than a random insertion. Additionally, inserting s^* at random within the first 10 positions of any playlist is much worse than a random insertion at any position.

C.5 Robustness of collective strategies

Table 2 shows that the DirLoF strategy is robust against hyperparameter (hp) changes. It consistently outperforms the Random strategy across all configurations. Finally, Table 3 illustrates the effectiveness of DirLoF and Random across different numbers of training epochs for the recommender. Notice that the effectiveness of the DirLoF strategy decreases if model training is stopped early.

C.6 Internalities and externalities of algorithmic collective action

Here we provide more details on the results presented in Section 4.3.

Metrics for model accuracy. To assess the quality of recommendations we follow Chen et al. [14] and Bendada et al. [7] in using the following three popular performance metrics: **R-precision** measures the fraction of recommended items present in the masked ground truth, augmented by

Table 3: Mean Amplification (Std Dev) for relative to trained epochs ($\alpha = 0.001$). The best-performing strategies are highlighted in bold.

	Strategy	
# of epochs	`DirLoF`	`Random`
1	0.02 (0.04)	**4.10 (9.17)**
2	**7.97 (5.86)**	0.08 (0.17)
4	**9.88 (16.10)**	2.17 (1.81)
8	**19.45 (12.82)**	2.51 (4.82)
12	**25.79 (10.30)**	4.23 (3.13)
16	**14.95 (6.00)**	3.07 (4.85)
18	**16.38 (7.37)**	3.36 (4.45)

Table 4: Mean (± 95% CI) recommendations without collective action (R_0), total gained recommendations (ΔR), and gained recommendations in % of R_0 (considering songs that are recommended at least once without collective action) for direct anchors, indirect anchors and others over five folds.

Metric	Direct anchors	Indirect anchors	Other songs
R_0	0.09 ± 0.02	360.94 ± 11.29	0.29 ± 0.00
ΔR	-0.00 ± 0.01	2.68 ± 3.74	-0.00 ± 0.00
ΔR in % of R_0	-0.00 ± 0.00	0.02 ± 0.01	-0.00 ± 0.00
Song counts (per fold)	2784	157	2246756

artist matches. The Normalized Discounted Cumulative Gain (**NDCG**) measures the ranking quality by rewarding relevant tracks placed higher in the recommendation list. The number of clicks (**#C**) quantifies how many batches of ten song recommendations (starting with top candidates) are needed to find one relevant track. The Deezer model trained on the unmanipulated data achieves comparable results to the winning solutions of the RecSys 2018 APC challenge along these metrics [60].

Impact on other artists. Unlike the partially aligned interests of the firm and the collective, the dynamics among artists differ, since boosting recommendations for one artist inevitably reduces the visibility of others. We are interested in understanding who is affected by our strategy. For the purpose of this analysis, we hold the total number of recommendations at inference time constant, making it a zero-sum game.

To complement Figure 7, in Table 4, we show the effect of collective action (`hybrid` strategy with $\alpha = 1\%$) on other songs. In line with the result presented in Section 4.3 (showing bins of songs with similar frequencies for just one fold), we do not find any evidence that any other songs experience a systematic change in exposure due to collective action, not even the targeted (in)direct anchor songs.

User experience of collective participants Collective participants are platform users who continue consuming content on the platform during and after performing strategic actions. To understand the price of collective action, we investigate the downstream effect on their own user experience. After all, users are likely to engage in collective action only if it does not result in significant detriment to their own content consumption experience. While a perfect measure of user satisfaction is outside the scope of this work, we utilize participant's known preferences as a basis for the experienced quality of recommendations. More precisely, for any collective participant that manipulated their playlist with $h(p) = (p^-_{i^*}, s^*, p^+_{i^*})$, we use the targeted context $(p^-_{i^*})$ as a seed for the recommender and evaluate the output recommendations using the user-generated continuation of the playlist $(p^+_{i^*})$ as the ground truth.

We revisit the collective strategy outlined in the ablation study (Section C.1) to scrutinize the internalities of collective action. We find that the recommendations for collective participants are stable and robust under authentic strategic playlist manipulations. Figure 13 illustrates the precision score—measuring the similarity of recommendations for different strategies—across attacked contexts. It is around 80% on average, with little variation across strategies. This is likely attributed to inherent instabilities in the top K recommendations rather than the specifics of the strategies [36].

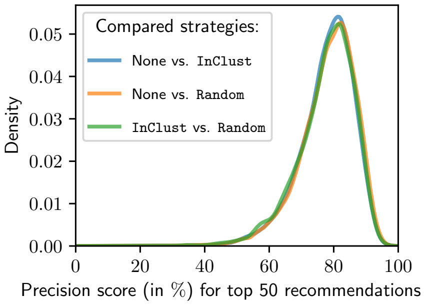


Figure 13: Variation in top 50 song predictions for attacked contexts: Predictions are stable under different collective action strategies.

Table 5: Average model performance: Predictions are robust under different collective action strategies.

Strategies	NDCG	R-precision	#C
None	0.35 ± 0.0	0.28 ± 0.0	1.43 ± 0.1
`Random`	0.34 ± 0.0	0.28 ± 0.0	1.50 ± 0.1
`Inclust`	0.35 ± 0.0	0.28 ± 0.0	1.42 ± 0.1

Furthermore, by participating in collective action, users do not affect the variety of songs they get recommended. This stability of model predictions despite the coordinated collective action is shown in Table 5: Inserting s^* between $p_{i^*}^-$ and $p_{i^*}^+$ does not significantly distort the recommenders ability to predict $p_{i^*}^+$ from $p_{i^*}^-$. Interestingly, random placement of songs reduces the performance slightly more (especially for #C), as in this case, s^* can be inserted within the context $p_{i^*}^-$ by pure chance.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and in the introduction are shown in detail in Section 4. The scope outlined in the abstract and in the introduction corresponds to the framework described in Section 3. Furthermore, Section 1.1 accurately reflects the paper's contributions.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of our work are discussed in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experimental methodology is clearly described in Section 4 and all steps of the experiments are clearly outlined, ensuring the reproducibility of the presented results. Furthermore, the experimental setup is described in detail in Section B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide open access to the code here: `https://github.com/joebaumann/recsys-collectiveaction`. The data used for the experiments is publicly available and referenced in the paper. Furthermore, the hardware setup and the detailed experimental setup are described in Section B.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All the training and test details (e.g., data splits, hyperparameters, etc.) are described in Sections 4 and B as well as in the open access code repository.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All experiments were run several times with different seeds and we report error bars (where appropriate) along with a description throughout the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed information on the used computer resources in Section B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The underlying goal of this research is a positive societal impact as clearly described in the introduction as well as in Sections 2.2 and 3. Broader impacts (such as unintended uses) are additionally discussed in Section 6.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used for this research are properly credited and the licenses are respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.