mAtrIx: Exploration of Multi-arm Bandit Algorithms Using a Content-based Simulated Social Media Recommendation System

Kris Frasheri, Ken Jen Lee, Xizi (Lucy) Wang, and Ryan Yen

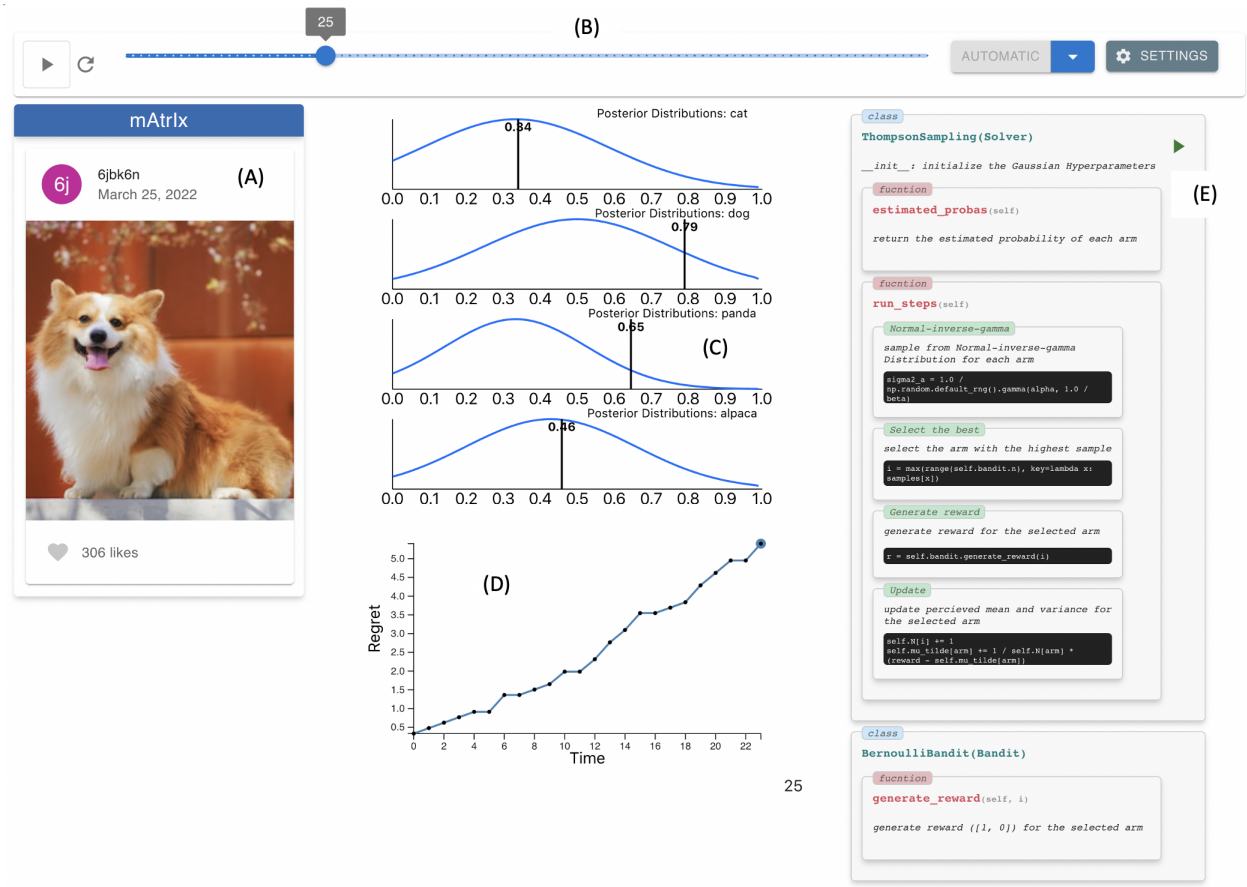


Fig. 1: Interface of the interactive website: (a) simulated social media interface, (b) timeline with buttons for changing mode and other settings, (c) posterior distribution plots, (d) regret plot, (e) major steps in the algorithm demonstrated using psuedocode.

Abstract—Multi-armed bandit (MAB) algorithms are often used to build recommendation systems that provide personalized advertisements on e-commerce websites and social media. However, non-machine-learning (ML) experts may find it difficult to understand or interpret how these complex algorithms lead to personalization. In this paper, we propose a new interactive visualization for explaining the internal workings of various MAB algorithms and their outputs using a common real-world scenario: social media advertisement. It contains a simulated interactive social media application and an explanation dashboard, allowing learners to explore MAB algorithms in a customized recommendation system. Our main contributions are the design and implementation of the interactive visualization that embeds the MAB algorithm learning progress into a real-world scenario as well as three use cases that explain how mAtrIx facilitates MAB algorithm learning.

Index Terms—Machine learning, reinforcement learning, visualization, AI explainability, human-computer interaction

1 Introduction

The multi-armed bandit (MAB) problem is a classic challenge studied across disciplines like computer science, operations research, and statistics [25]. Particularly, given an environment with a finite amount of actions, an agent is tasked with identifying which action will yield the greatest long-term reward. Fundamental to this problem is the exploration versus exploitation trade-off. Since the true reward of every

- *Kris Frasheri is with University of Waterloo. E-mail: kfrasher@uwaterloo.ca*
- *Ken Jen Lee is with University of Waterloo. E-mail: kenjen.lee@uwaterloo.ca.*
- *Xizi (Lucy) Wang is with University of Waterloo. E-mail: l84wang@uwaterloo.ca.*
- *Ryan Yen is with University of Waterloo. E-mail: r4yen@uwaterloo.ca.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

action is unknown to the agent, it must balance its desire to explore different actions to gain information about their rewards, with its need to exploit an action that appears to yield the greatest reward from historical data [25]. The challenge in the multi-armed bandit problem is to identify the right balance between exploration and exploitation of an environment that leads to optimal long-term rewards.

Various algorithms have been created to tackle the MAB problem. Examples include epsilon-greedy, Upper Confidence Bound (UCB), and Thompson Sampling [25, 30]. MABs algorithms have been further applied to a variety of real-world scenarios, including dynamic pricing [18], online advertisement selection [7, 24], and content recommendation [1]. While MAB algorithms have shown great promise in many applications, their widespread use also raises important ethical concerns [4]. Particularly, in the case of content recommendation in social media platforms, MAB algorithms (i.e., agents) are tasked with learning what a user's preferences are in order to recommend pieces of content that the user is most likely to engage with [21]. However, the most significant concern is the potential for these algorithms to reinforce and even amplify existing biases and discrimination, and encourage polarization [10, 11]. For example, if historical data used to train the algorithms is biased or unrepresentative, the recommendations provided by these algorithms can further exacerbate these biases [10], leading to unfair or discriminatory outcomes. As a result, the use of such algorithms might result in users being trapped in filter bubbles, such that only content that is similar to those users were exposed to in the past is recommended to them, isolating them from alternative viewpoints [2]. While this can lead to a more personalized and engaging user experience, it reduces users' exposure to diverse opinions, limiting their access to accurate and trustworthy information, and increasing polarization in online communities [11]. Over time, this could lead to the formation of echo chambers, which are "environments in which the opinion, political leaning, or belief of users about a topic gets reinforced due to repeated interactions with peers or sources having similar tendencies and attitudes" [8].

To address these issues, it is critical to understand and comprehend the recommendation algorithms behind the online social media platform [4, 5]. Users may acquire a better understanding of how the recommendation system works, what data it utilizes to produce suggestions, and how it influences their online experience by visualizing it. This can assist users in being more conscious of their online filter bubble and taking actions to break free from it by searching out different contexts and engaging with opposing ideas. Based on this idea, we aim to build a web application that can provide users with an intuition of how such algorithms could affect the type of content as users interact with a simulated social media via visualization. Furthermore, displaying the recommendation algorithm can assist practitioners and students in identifying biases and limitations in the system and encourage better learning of MAB algorithms through a relatable example.

There are multiple existing tools or codes that provide visualizations for MAB algorithms. However, they suffer from a few limitations. For instance, some tools provide only static or pre-computed visualizations, which may not be informative for users who want to learn how the algorithm works in different scenarios. The majority of existing websites and code only offer an explanation of the MAB algorithms without providing adequate contexts on how these algorithms impact real-world scenarios like online shopping. Furthermore, the prevailing focus of previous work has been on the visualization of results and the examination of how different parameters impact the outcome, rather than providing step-by-step visualization to clarify how the algorithm operates. This makes it difficult for users to comprehend how the final results are generated. Based on these previous visualization works, we built mAtrIx, an interactive website that provides a comprehensive explanation of various MAB algorithms in the context of social media recommendations. It is a unique tool that integrates multiple visualizations, such as the regret plot, cumulative rewards, and arm selection distribution, into a single platform. The mAtrIx allows users to switch between different MAB algorithms, adjust parameter settings, and visualize the performance of the algorithm in real-time. Additionally, mAtrIx offers a step-by-step walkthrough of the code used to implement each algorithm, making it easier for novice learners to understand how different algorithms work.

In the below sections, we start by introducing the related works, followed by the proposed design of our system, named mAtrIx, including its system architecture, visual representation and user interactions. After that, we present three use cases describing three unique user flows, and lastly, a discussion on our design process and possible future work.

Our main contribution is the development of mAtrIx, an interactive website that provides a comprehensive explanation of various MAB algorithms in the context of social media recommendations. Here are our contributions of this work:

- An interactive visualization learning tool, mAtrIx, that demonstrates the inner workings of MAB algorithms. Specifically, mAtrIx:
  - Connects MAB algorithms to real-world scenarios, specifically a social media recommendation system, by providing real-time updates of posts with different tags based on the various MAB algorithms.
  - Provides a step-by-step walkthrough demo of code, allowing users to understand how different algorithms work and how changing various parameters will affect the final outcome.
  - Integrates visualizations of the results and plots, allowing users to follow different algorithms and reason through their choice and results. This enables users to interactively explore the impact of different algorithmic decisions and hyperparameters on recommendation outcomes.
- We provided three use cases that help readers understand how learners can use mAtrIx to learn and explore three different MAB algorithms in manual, automatic, and demo modes.

2 Related Work

2.1 Social Media as a MAB Problem

Many social media used to display content in their feed using chronological order by default, i.e., content created by a user's social network is shown in an order determined by how recent the piece of content was created. However, in an effort to increase user engagement, they (e.g., Facebook [6], Instagram [14], Twitter [16]) have instead begun showing content based on "engagement-based ranking" [13, 14]. While real-world social media might make use of complicated algorithms for the purposes of content recommendation, we use a simpler framing of a simulated social media as a MAB problem [21] to provide a gentle introduction and intuition of how MAB algorithms work under the hood. Particularly, the goal of MAB algorithms in this context is to recommend content with the highest probability of user engagement. This engagement could come in the form of various interactions, including whether the user "liked" a piece of content, commented on it, reposted it etc. [21]. As users interact with a social media interface, a MAB algorithm would be actively trying to select, from a finite pool of possible content, which piece of content to show to users next. As such, each arm in the MAB problem is represented by a type of content that the algorithm or agent can recommend (or pull, using the analogy of having multiple one-armed bandits) [21]. For example, the algorithm might be choosing which tag the next piece of content should belong to (e.g., on an image sharing social media platform, example tags could be selfies, sceneries, food etc.). The amount of reward received by the algorithm for a particular action (of recommending a particular type of content) can be determined by how much a user engages with a piece of content. This reward is then used to update the estimated reward associated with the specific type of content chosen as the previous action, which should lead to more accurate information on users' preferences overtime, and hence more effective recommendations.

2.2 Potential Harms of Filter Bubbles and Associated Mitigation Strategies

While using engagement-based content recommendation systems could lead to higher user engagement, prior research has shown that they

could also have certain disadvantages to users. Particularly, such algorithms often lead to the formation of filter bubbles, where users are only shown content belonging to a very narrow range of information diversity [23]. Often, users are not even aware that they are inside a filter bubble, and have no awareness of the content that exists outside their bubble [23]. Such a phenomenon carries with it various harmful effects on both users and society at large. Filter bubbles are often not transparent, and limit users' freedom of choice and autonomy. Since users inside filter bubbles are not aware of content with different perspectives, their ability to make reasonable decisions through their awareness of the range of opinions and engagement in diverse discussions is compromised [4]. Moreover, overtime, filter bubbles could lead to worsened ideological segregation and polarization, since users are being exposed to an increasingly narrow range of information and other users who subscribe to similar ideologies [4, 11]. At a larger scale, filter bubbles could be a threat to democracy if the algorithms causing these bubbles are designed to serve the interests of certain individuals or groups [4].

To help mitigate these disadvantages, prior works explored different ways of presenting to users a more diverse range of content. For instance, Bhuiyan et al. designed OtherTube, a browser extension for YouTube that displays strangers' personalized YouTube recommendations. They found that OtherTube was effective at allowing users to break their filter bubbles towards developing new interests or rediscovering old interests through showing them videos outside what the default YouTube recommendation algorithm suggests to them [3]. Ookalkar et al., on the other hand, designed a browser extension aimed at breaking users' news filter bubbles on Twitter by changing users' Twitter feed through the inclusion of tweets from agencies with ideological standings that are different from that of the users [22].

Other than being exposed to more diverse ranges of content, prior research also found that higher user awareness of the existence of social media filter bubbles correlated with more actions taken against such bubbles. Specifically, Burbach et al. conducted a survey of Facebook users and found that users who perceived filter bubbles as an issue took significantly more actions against such bubbles, mainly through engaging with a wider range of content [5]. As such, raising users' awareness of the existence of filter bubbles could be an effective way of mitigating the harms of filter bubbles [4]. Aligning with this finding, mAtrIx hopes to benefit users not through exposing them to diverse content, but by unveiling the inner working of MAB algorithms, and how such algorithms could influence what they are shown when using various social media platforms. This is important, since content recommendation algorithms and filters used in real-world social media are seldom known by, or transparent to, the users [4].

2.3 Related Visualizations

The objective of this work is to develop an interactive website that provides a comprehensive explanation of various MAB algorithms in the context of social media recommendations. The MAB problem is a well-known problem, and has become increasingly relevant with the advent of personalized recommendation systems in e-commerce and social media platforms. Prior research has predominantly focused on understanding recommendation systems based on MAB algorithms through the use of visualizations [9, 19, 20, 26]. However, these studies are not primarily designed to provide insight into the internal workings of these algorithms and how they function. Multiple projects and websites have also focused on visualizing the distributions or probabilities over time for various MAB algorithms. For example, Flyon [12] used a simulation to demonstrate how the Thompson Sampling algorithm decides and reacts to changes in the reward signal. Similarly, Stubley [29] developed multiple visualizations within a tutorial article to demonstrate how epsilon-greedy and Thompson Sampling function when distributing website visitors. Also, Keaton and Sabbaghi presented an application for visualizing arm assignments and regret to facilitate users' understanding of the performance of different MAB algorithms as a pedagogical tool [15]. Another relevant website by Yun et al. [17] contains an interactive visualization article of how different algorithms play slot machines automatically, as well as visualizations of each algorithm with varying hyperparameter settings.

However, these animated or interactive visualizations have several limitations. Firstly, they seldom connect MAB algorithms to real-world scenarios, instead focusing solely on explaining the algorithms by adjusting parameters. Our project utilizes a social media recommendation system as an example context, and provide real-time updates of posts with different tags based on the various MAB algorithms. Secondly, a lack of step-by-step walkthrough demos of code has made it difficult for novice learners to understand how different algorithms work. Our project addresses this by implementing a pseudocode step-by-step walkthrough, allowing users to understand the interim results and how changing different parameters will affect the final outcome. Lastly, existing visualizations of results and plots are seldom connected, with some websites only providing visualizations for the regret and distribution plots separately. Our project seeks to address this issue by allowing users to select different algorithms and reason through their choices and results using integrated visualizations.

2.4 Popular MAB Algorithms

The multi-armed bandit problem is a simple yet difficult problem to address. It is simple due to the short description of the problem requiring an agent to optimize rewards in its environment, however it is difficult as there is no known tractable optimal solution. There are several algorithms that have been studied and applied in practice that aim to solve the multi-armed bandit problem that we would be inclined to explore throughout our project.

2.4.1 Epsilon-greedy

The most naive approach is the epsilon-greedy algorithm, where an agent will randomly explore one of the actions in their environment with probability ε, otherwise greedily selecting the action with the highest observable payoff to date.

Algorithm 1 Epsilon Greedy Multi-Armed Bandit Algorithm

Require: $0 \leq \varepsilon \leq 1$
Require: $T \geq 0$
$t \leftarrow 0$
$Q \leftarrow \{0, ..., 0\}$ ▷ Length of Q is N
while $t \leq T$ **do**
  $p \leftarrow$ ... *Generate random number between* 0 *and* 1 ...
  if $p \leq \varepsilon$ **then**
    $a* \leftarrow argmax_a(Q)$ ▷ Select action with highest cumulative reward to date
  else if $p > \varepsilon$ **then**
    $a* \leftarrow random(1, N)$ ▷ Select a random action
  end if
  $t \leftarrow t + 1$
  ... Perform action $a*$ to receive reward r ...
  $Q[a*] \leftarrow Q[a*] + r$
end while

Algorithm 1 presents the epsilon greedy MAB algorithm. N represents the number of total actions in the environment, T represents the number of trials performed, Q is the cumulative rewards of each action $\in [1, N]$. ε is a user-defined variable that controls the nature of exploitation and exploration. We can see for every time step t that we first generate a random integer $0 \leq p \leq 1$ that then gets compared against ε. If the value of p is greater, then our agent will select the action that has the best perceived cumulative reward to date. Otherwise, the agent will select a random action in the action space.

As such, this algorithm uses non-adaptive exploration, as when it chooses to explore is independent of the history of observed rewards [25]. Moreover, when it chooses to explore, it randomly chooses an action without consideration for how much it knows about each action, or how accurate the estimated payoff is for each action [30].

As a result, the epsilon-greedy strategy is prone to converging to local optima states due to its lack of exploration or exploitation dependent on the value chosen for ε.

2.4.2 Upper Confidence Bound

The Upper Confidence Bound (UCB) algorithm uses a confidence interval of the estimated rewards to determine which action to select next.

$$A_t = argmax_a(R_t(a) + c\sqrt{\frac{ln(t)}{N_t(a)}})$$

Specifically, at time step t, the action chosen would be the action that yields the highest value from the equation shown above, which is the sum of the estimated reward (i.e., payoff) of the action at the time and the second term. This term includes c, the confidence level that controls the degree of exploration, and the square root of the natural logarithm of time step t divided by how many times action a has been selected so far [30]. Particularly, the fraction in the second term measures the amount of uncertainty associated with the action: it increases if an action is selected less frequently, and vice versa [30]. With this, the UCB algorithm is able to balance the problem of exploration and exploitation by balancing the information gained by exploring under-utilized actions and exploiting those with the highest estimated payoff.

2.4.3 Thompson Sampling

The Thompson Sampling algorithm is a Bayesian approach that samples from the posterior distribution of the rewards from past actions to decide which action to take every epoch.

Algorithm 2 Thompson Sampling Multi-Armed Bandit Algorithm

Require: $T \geq 0$
$V \leftarrow 0$
while $t \leq T$ **do**
  Sample $R_1(a), ..., R_k(a) \sim Pr(R(a)|r_1^a, ..., r_n^a)\ \forall a$
  $\hat{R}(a) \leftarrow \frac{1}{k}\sum_{i=1}^{k} R_i(a)\ \forall a$
  $a* \leftarrow arg\,max_a(\hat{R}(a))$
  Execute $a*$ and receive reward r
  $V \leftarrow V + R_t$
  Update $Pr(R(a*))$ based on r
  $t \leftarrow t + 1$
end while

Algorithm 2 presents the Thompson Sampling MAB algorithm. The notion is that for every time step t, we first want to sample k potential average rewards from the posterior distribution established for each action a. Initially, the agent will not have any knowledge of the underlying distribution that shapes the true mean reward for each action, however as the number of trials of an action increases, the confidence in the estimated mean reward of the action increases. The next step is to estimate the empirical average reward modelled by $\hat{R}(a)$ for all k samples we took from all actions a. We then select and execute an optimal action $a*$ to execute, based on the estimated empirical averages calculated in the previous step for all a. This will result in a reward r being returned for the selected action $a*$, where we will update the prior distribution with the newly seen reward. Supplying this new reward will allow the agent to gain confidence in their estimation of what the true underlying reward of selecting action $a*$ is. In our application, these priors were the normal inverse-gamma distribution, however it is possible to select other distributions of interest that tailor towards the statistical nature of the MAB problem being addressed.

Over time, exploration decreases and exploitation increases as the confidence in selecting actions with the highest perceived payoff will be chosen. This enables the agent to converge on the action that yields the highest estimated mean reward to exploit, naturally shifting from exploration to exploitation.

3 Proposed Design

Our proposed design includes a content-based simulated social media recommendation system that allows learners to explore multi-armed bandit algorithms in a customized manner. The interface of mAtrIx consists of a few main components (Fig. 1). On the very left of the interface is the simulated social media feed (Fig. 1(A)). At the top is the timeline, and a button to open the settings panel (Fig. 1(B)). The settings panel itself allows users to configure the algorithm used and the reward rates of each content tag (Fig. 6). In the middle, the distribution plots and regret plots are shown (Fig. 1(C-D)). Lastly, the right side of the interface shows the pseudocode steps associated with the currently chosen algorithm (Fig. 1(E)). Below, we will describe the technical implementation and architecture of the system, and detail the specific visual representations and interactions. Overall, our proposed design aims to provide an interactive and accessible educational tool for novices to understand the complex workings of multi-armed bandit algorithms and their impact on our daily lives through social media recommendation.

3.1 System Architecture

The system architecture of our proposed interactive visualization for exploring MAB algorithms is fully implemented using vanilla JavaScript and ReactJS [27] (Fig. 2). We previously built a Python backend server, which was responsible for handling incoming requests and generating personalized recommendations using various algorithms. However, due to the need of real-time updates and the possibility of having an overwhelming amount of API requests, there is a chance that the back-end API may crash or slow down significantly. To address this issue, we implemented all the MAB algorithms in a plain JavaScript utility file instead. This file contains a class called "GenerateNewBandit", which can generate results such as regret, current step, interim result for demo code, reward, posterior distribution, and tag (the arm to pull next) based on the Thompson-sampling, UCB, and epsilon-greedy algorithms. These results are then sent back to the React components, which are responsible for rendering the interactive visualization. To ensure consistent state management across components, all results are stored in a global state object called "BanditInfo," which is managed by the state management library, RecoilJS [28]. Each React component, such as the social media app, timeline, distribution plot, regret plot, and demo code, listens to changes in the global state object and updates its respective views accordingly.

3.2 Visual Representation

3.2.1 Simulated Social Media Interface

The simulated social media interface (Fig. 3a and 3b) is modelled after a typical image scrolling social media on a mobile device (e.g., Instagram). Each random-generated post consists of a username, a profile picture, a date denoting when the photo was posted, a photo representing a specific arm, and a "like" icon that users can click on to like the post. Moreover, we provided the learner with a tutorial post (Fig. 3a) to help the learner get started with the explorations of our simulated social media application.

3.2.2 Plots

To provide an intuition of how an algorithm converges over time as the number of interactions with the simulated social media increases, a regret plot is included, inspired by van den Burg's work [32]. . Particularly, we calculate the regret by the distance between the "arm" an algorithm pulls and the optimal "arm" to pull. As more interactions happen, the regret accumulates. However, as convergence occurs, the regret plot "flattens" out since the regret for the later time steps are smaller than the regret of the earlier time steps (Fig. 4a and 4b). Mathematically, this is expressed as:

$$Regret(T) = \sum_{i=1}^{T} \max_{i \in [1,K]} (\mu_i^* - \mu_i)$$

Where T is the total number of trials, K is the total number of arms, μ_i^* is the true mean reward of the optimal arm at step i and μ_i is the true mean reward of the arm the agent selected at time i. In this equation, regret can never be negative and as the agent converges to the optimal arm our result provided by $Regret(T)$ should become stagnant.

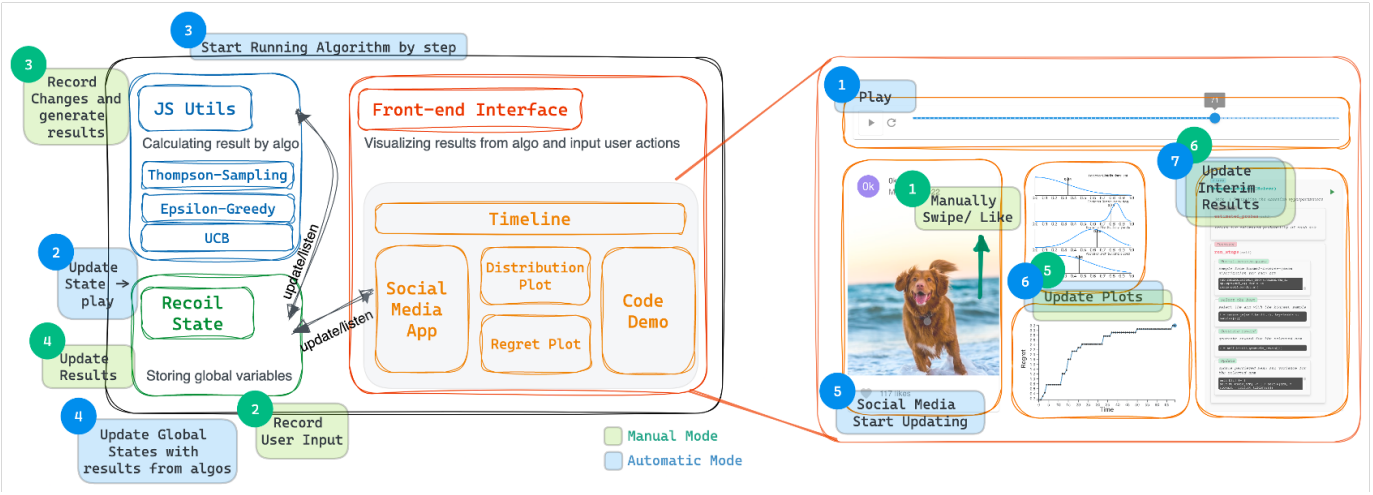
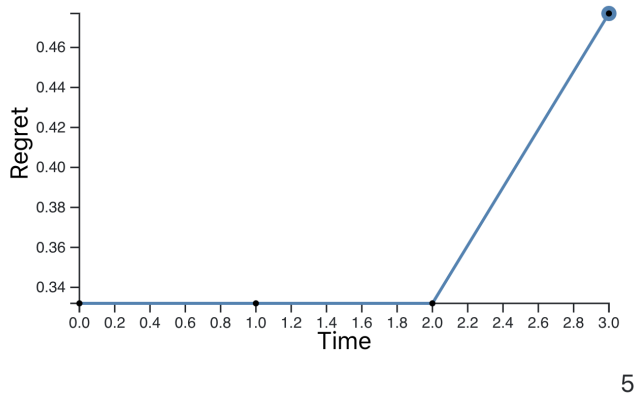


Fig. 2: An overview of the system architecture and the walktrhough of both automatic and manual mode.

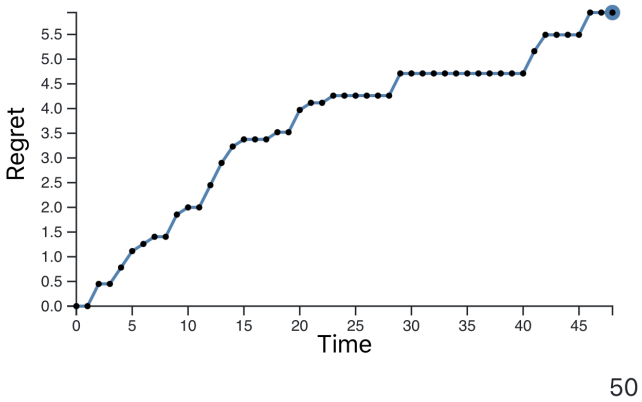
(a) The very first post in the simulated social media is a tutorial picture.

(b) Example cat post (all photos collected using the Unsplash API [31]).

Fig. 3: Example posts in the simulated social media.



(a) Example regret plot at step 5 in automatic mode.



(b) Example regret plot at step 50 in automatic mode.

Fig. 4: Regret plots at step 5 versus step 50 with dynamic X and Y axes scaling.

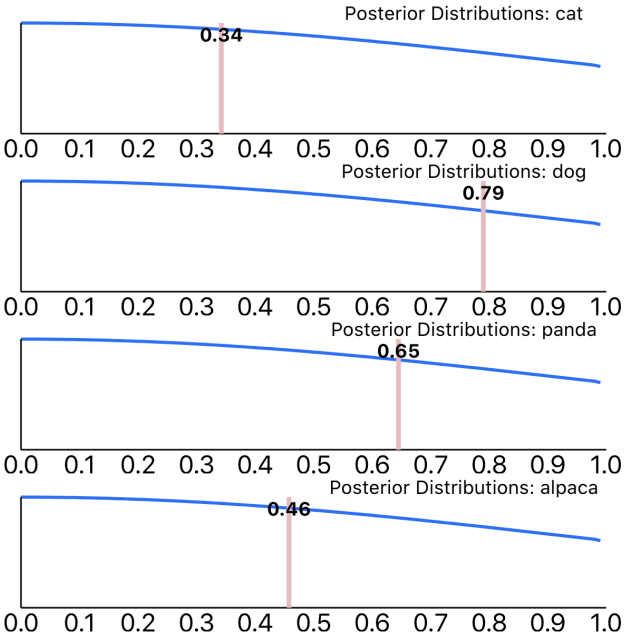
In order to increase the clarity of the regret plot, the X and Y axes scale dynamically. For example, at every time step, the X axis' range is from 0 to whatever the current time step is. This is so that the regret plot will always take up the entire range of the X axis for maximum clarity. Similarly, the Y axis scales at each time step as regret slowly accumulates.

In addition, for the Thompson Sampling algorithm, the posterior distribution for each bandit is also included, inspired by Flyon's work [12]. Each posterior distribution plot shows the distribution from which the Thompson Sampling algorithm samples for that specific bandit. The more the simulated social media is interacted with, the more the distribution of each bandit converges with the real reward rate of that bandit, as seen in Fig. 5a and 5b.

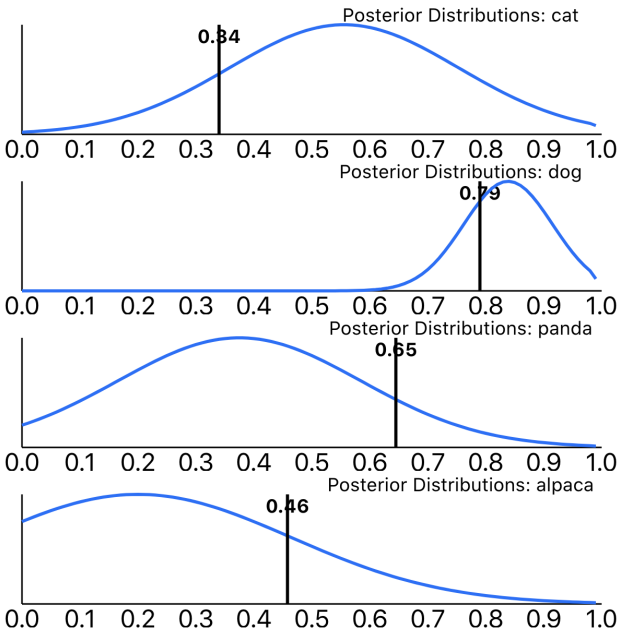
3.3 User interactions

3.3.1 Simulated Social Media Interface

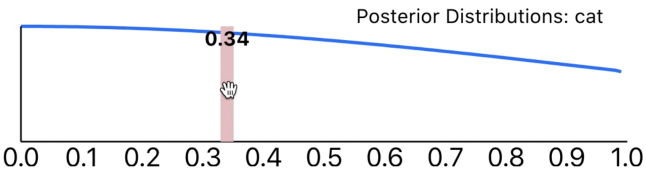
At the beginning of each time step, the simulator generates a post with a recommended image that represents the tag calculated by the MAB algorithm that the learner choose to explore. Users are only allowed to interact with the social media application during the manual simulation. In the manual mode, the learner can choose to like the post by clicking the heart button below the image or dislike the post by doing nothing



(a) Posterior distributions for each of the four bandits (i.e., topic of picture in the simulated social media interface) at step 1 in automatic mode.



(b) Posterior distributions for each of the four bandits (i.e., topic of picture in the simulated social media interface) at step 50 in automatic mode. The algorithm has nearly converged to the most rewarding bandit, dog, in this case.



(c) Users can drag the pink line to change the true reward rate of each bandit ("cat" in this case) in automatic mode.

Fig. 5: Posterior distributions at (a) step 1 versus (b) step 50, and the (c) draggable slider that can be used to configure the true reward rate of the tag while in automatic mode.

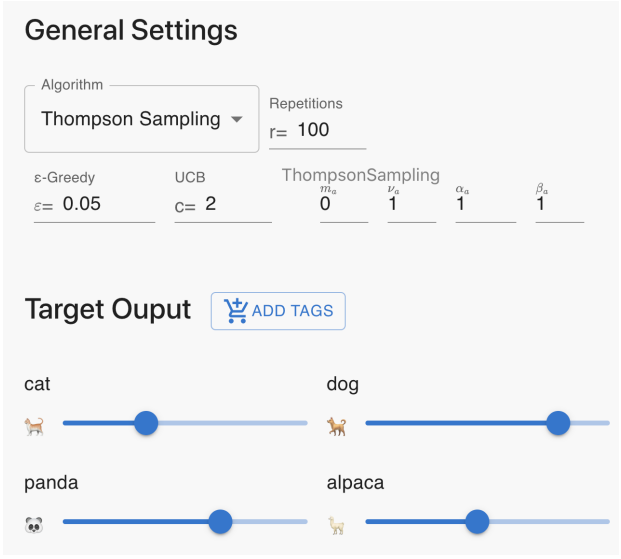


Fig. 6: Settings panel.

based on their own preferences. After this brief decision, they can swipe up the post to continue browsing the next post, like what they might do in other social media applications (e.g., Instagram). In the automatic simulation mode, in each step, the system automatically clicks the heart button for the learner based on the probability of the predefined simulated user liking this specific type of image (or the true reward rate). Then, the system automatically does a swipe-up action for the learner and generates the next post.

3.3.2 Timeline

The learners are able to switch between manual mode and automatic mode, as well as check the progress in the timeline component. The number in the timeline indicates time steps that the learner has experienced. Clicking on the play button plays or replays the automatic simulation, and clicking on the pause button pauses the simulation. The reset button starts the simulation from the beginning.

3.3.3 Settings

Users can open a floating panel (Fig. 6) by clicking on the Settings button beside the timeline (Fig. 1). In this settings panel, users can select which algorithm to use, algorithm-specific parameters, and edit simulated users' preferences of (or the true reward rates for) each type of content. Another method that users can change the true reward rate of each bandit is by sliding the true reward rate line in the posterior distribution plot of that particular bandit, as shown in Fig. 5c.

3.3.4 Pseudocode Walk-through

To facilitate users' understanding of how different algorithms work, we incorporate a Pseudocode Walk-through visual representation into our proposed design (Fig. 7). This visual representation consists of multiple layers of code, including class, function, and code steps (Fig. 7.2). Each layer is depicted using block-based visualization with different colors, and users can hover over each block to view its name, parameters, natural language summarization on how the block of code is responsible for (Fig. 7.3), and detailed code implementation (Fig. 7.5). As the user pressed the play button (Fig. 7.1) to indicate the start of the demo code flow, an arrow will step through each block (Fig. 7.6), highlighting the current block and displaying real-time updated interim results on the arrow (Fig. 7.4). Once the code finishes running, the arrow will link to the social media app, displaying the generated tag from the algorithms, which is the same as the current image tag displaying on the social media app. This visual representation allows users to follow along with the execution flow of code step-by-step and easily navigate between different blocks.

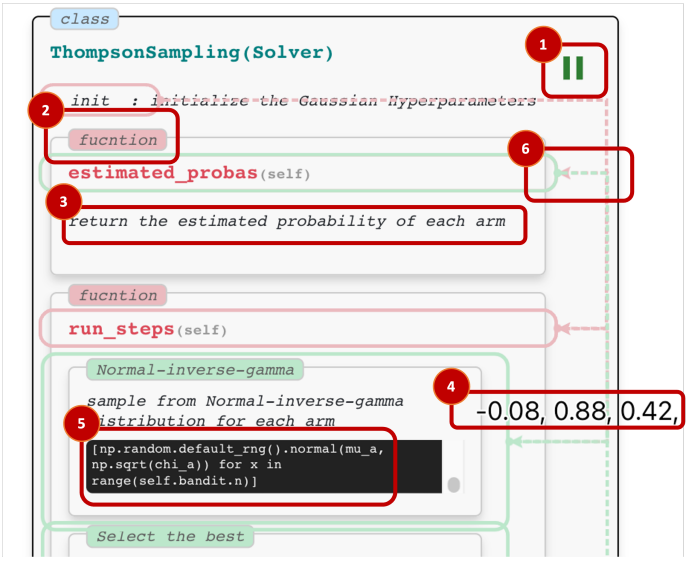


Fig. 7: The visual representation of the Pseudocode Walk-Through component. (1) Button to start and stop the step-by-step demo; (2) Name of the block on the level, consisting of class, function and code step; (3) Natural language summary of what the block does; (4) Real-time update of the intermediate result of the algorithm; (5) The real code implemented in Python, the user can scroll through the code to see how the block is actually implemented; (6) Demo arrow leading from the executed block to the next block with the current intermediate result.

Fig. 8: The pseudocode Walk-Through component for the epsilon-greedy algorithm.

Epsilon-greedy. The pseudocode for the epsilon-greedy algorithm (Fig. 8) consists of the following main steps: 1) if a randomly generated number is less than epsilon (set to *0.05* in our code), then the next tag is randomly chosen, 2) otherwise, the arm with the highest estimated reward at the current time is chosen. 3) Then, the algorithm waits for the reward from the simulated social media (1 if the post was liked when it was swiped up, 0 otherwise), and lastly, 4) the estimated reward for the chosen tag is updated.

UCB. The pseudocode for the UCB algorithm consists of the following main steps: 1) increment the time step, 2) pick the best action using the UCB formula (Section 2.4.2), 3) wait for the reward from the simulated social media, and 4) update the estimated reward for the chosen tag.

Thompson Sampling. The pseudocode for the Thompson Sampling algorithm consists of the following main steps: 1) sampling a point from the Normal-inverse-gamma distribution for each tag, 2) selecting the tag with the highest sampled value, and the same steps 3 and 4 as the two previous algorithms.

4 Example Use Cases

We will highlight the three modes in which the system can be operated by the user in the following section. The automatic mode allows the user to activate a simulated social media user that automatically interacts with the simulated social media interface, allowing the user to observe changes and plots in real-time. On the other hand, the manual mode enables the user to swipe and like the social media posts manually. Lastly, the demo mode provides a step-by-step walkthrough of the pseudocode for a specific algorithm, allowing the user to understand the inner workings of the algorithm and how it generates the output.

4.1 Manual

When the learner opens the website, they notice that the default settings of mAtrIx use the Thomson Sampling algorithm in manual mode. Closing the settings panel, the learner first sees a post with instructions (Fig. 3a) on the left, in the simulated social media interface. Following the tutorial picture, the learner clicks the like button and swipes the tutorial post up. Then, a new post with a randomly chosen image containing a dog shows up. After the learner clicks on the like button and swipes up the dog post, they can observe the normal distribution curve in the posterior distribution graph (Fig. 5b) of the "dog" bandit's arm moving to the right. Then, a cat post appears. When the learner dislikes it, the curve for the "cat" bandit arm moves to the left. The learner decides to always like posts with dog images, like panda posts sometimes, and never like posts with photos of cats or alpacas. After swiping through posts for some time, the learner understands that the more they like dog posts, the higher the expected reward of the "dog" bandit arm is, and the higher the probability that the next post contains a dog photo. Also, the learner notices the changes in the regret plot and understands that when an arm that does not have the highest expected mean reward is selected (i.e., non-dog posts), the cumulative regret value will increase. Also, the learner finds that the regret plot slowly converges over time, with more dog images and fewer images of other animals showing up. They can then open the settings panel, choose another algorithm, for example, UCB, and continue to interact with the simulated social media interface. Similarly, the algorithm converges over time, as shown from the regret plot, albeit possibly slower than

the Thompson Sampling algorithm. At this point, they gain an intuition of how modern social media applications work – an agent behind the system learns from users' previous behaviors, calculates rewards, balances exploitation and exploration, and is encouraged to provide users with the type of images that users liked more frequently.

4.2 Automatic

The manual mode of mAtrIx has a limitation in that it requires users to interact with mAtrIx many times to observe the convergence effect of the MAB algorithm. This can be time-consuming, especially as more labels are added to the simulation. To address this, we added an automatic mode that allows users to visualize how the MAB algorithm will converge over time to the optimal set of images as perceived by the agent. In the automatic mode, users can adjust the true underlying rate of rewards that a simulated user will provide by manipulating sliders on the distributions page before starting the simulation. This can be seen in Figure 5a, where each arm will come with a red bar illustrating the true probability when a user sees the image, they will like the image before swiping. For each of these tags, the user can grab and move the slider to update this true underlying mean reward, as seen in Figure 5c. This allows them to see how changes in the rewards affect the algorithm's behavior.

To tune the hyperparameters of the MAB agent, users can adjust additional settings in the application, as shown in Figure 6. Once they are satisfied with their choice of MAB algorithm and agent hyperparameters, they can press play on the timeline to start the simulation. In automatic mode, images are selected automatically based on the probability that the simulated user will like them. This is determined by the true reward provided by the user and the MAB algorithm's convergence.

Figure 5b shows an example of how the posterior distributions begin to converge to the arm that yields the highest reward in Thompson Sampling. In this example, the label "dog" was provided with a 79% chance of being liked by the simulated user. After 50 iterations, the algorithm converges to this arm, which represents how the MAB algorithm will continually show dog images on mAtrIx to the user. Other arms with lower rewards, such as "alpaca," are discarded by the agent without presenting additional images to the user. The corresponding regret plot in Figure 4b shows how the agent's regret will converge and flatten out over time as the optimal arm is continually being exploited and chosen.

Overall, automatic mode serves as an efficient way for users to interact with the application and observe the behavior of the chosen MAB algorithm.

4.3 Demo Mode

In our example use case of demo mode, a user is observing the changes and plots in the automatic mode but wants to understand how the current dog image was being generated using the Thompson-Sampling algorithm. The user clicks on the pause button on the timeline component, which stops the automatic update. Next, the user clicks on the play button located in the top right corner of the pseudocode walkthrough panel (Fig. 7.1). The system starts to slowly walk through each code block with the arrow, beginning at the code block being executed and ending at the next code block about to be processed (Fig. 7.6). As the code is executed, the interim result will be real-time updated on the arrow. The user can see the final tag, "dog," being generated by the algorithm. Finally, the arrow links back to the social media app, allowing the user to see the generated tag on the image. This demo mode provides users with a step-by-step walkthrough of the algorithm, allowing them to understand the reasoning behind the generated tags while the system is running.

5 Discussion

Our web application was designed with the aim of improving user awareness and agency of personalization systems, while also educating non-experts about how these systems work. To this end, we employed an interactive and relatable metaphor to create an engaging learning experience. In manual mode, users can explore how MAB algorithms work. Automatic mode extends this knowledge, enabling users to gain transparency of the underlying model through visualizing how the algorithm works. Demo mode provides a step-by-step walkthrough of each agent, showing users every action the agent performs when presenting an image on mAtrIx, alongside pseudocode to represent the processes performed. Additionally, we developed a suite of three MAB algorithms, adjustable hyperparameters, and additional visualization features, such as regret plots and posterior distributions in Thompson Sampling, to allow users to learn about these agents through exploration.

Our work can be utilized in future research to develop tools that improve the transparency of algorithms, helping researchers and designers create more transparent and understandable systems. Furthermore, our web application aims to educate users about the prevalent algorithms used in modern social media applications, raising awareness of their effects and promoting critical reflection.

One limitation of our web application lies in the design of the hyperparameters users can adjust before runtime. Currently, all hyperparameters are presented in the settings toolbox, without explaining their meanings or significance to performance. These terminologies and symbols may be foreign to individuals who are not familiar with statistics, which could cause confusion about their purpose or effect on the agent's behavior at runtime. Additionally, during our evaluation, we observed a problem with representing the internal behavior of the UCB and epsilon-greedy algorithms. By default, these algorithms lack native visualization techniques, unlike the posterior distributions seen in Thompson Sampling. Although the user can see what image set the agent converged on, they may not fully understand why the image was selected, causing the internal workings of the model to become obfuscated to the individual. In future development, providing additional or non-traditional means of visualization could aid in adding additional transparency to MAB algorithms that lack traditional visualization.

Overall, our goal was to create an application that enhances user algorithmic literacy of MAB agents through our visualization techniques, promoting an interactive learning experience. We hope that our work inspires others to continue improving algorithmic transparency and educating users about the underlying systems that impact their online experiences.

6 Future Work

Currently, mAtrIx only supports 4 pre-defined bandit arms (including dog, cat, panda, and alpaca), and all image URLs were collected beforehand using the Unsplash API [31]. In the future, we plan to extend the work to allow the learners to edit the number and name of tags and generate post images based on customized tags using a generative adversarial network (GAN). Although an attempt at using GAN for this work was done, it revealed pictures that were of unsatisfactory quality. Moreover, our current system allows learners to explore three popular MAB algorithms. We see future work in supporting more algorithms and "drop in" models in ONNX format. Also, although we allow users to edit parameter values and switch algorithms, the exploration of each model is independent. We can extend the work by supporting a comparison of the performance of different algorithms and models in the same regret plot.

Currently, we only provide beginning users with a simple instruction post image to teach them how to interact with the embedded social media application. In the future, we plan to add a complementary tutorial page or a tutorial flow to guide the user to complete their first exploration step by step. Also, for novices who have limited statistical and computer science backgrounds, we plan to add descriptions for plots and terms, helping learners understand changes in graphs and comprehend algorithms.

Lastly, future work could include some form of evaluation on the effectiveness of mAtrIx. Particularly, such evaluations could include two outcome measurements: i) whether mAtrIx is effective at allowing students to learn more about MAB algorithms, and ii) whether mAtrIx is effective at increasing users' (who might not necessarily have a CS background, or are interested in learning about MAB algorithms) awareness of social media content recommendation.

7 Conclusion

In this work, we developed mAtrIx, an interactive visualization tool to facilitate the exploration of MAB algorithms for novices. To our knowledge, mAtrIx is the first model-agnostic visualization tool that embeds the MAB learning progress into a real-world daily scenario and uses multiple diagrams associated with pseudocodes to explain algorithms and support learning. We expect that this visualization can help novices understand MAB algorithms as well as how these algorithms influence our daily lives.

References

[1] Y. Ban and J. He. Local clustering in contextual multi-armed bandits. In *Proceedings of the Web Conference 2021*, pp. 2335–2346, 2021. 2

[2] M. M. Bhuiyan, C. A. Bautista Isaza, T. Mitra, and S. W. Lee. Othertube: Facilitating content discovery and reflection by exchanging youtube recommendations with strangers. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pp. 1–17, 2022. 2

[3] M. M. Bhuiyan, C. A. B. Isaza, T. Mitra, and S. W. Lee. OtherTube: Facilitating content discovery and reflection by exchanging YouTube recommendations with strangers. In *CHI Conference on Human Factors in Computing Systems*. ACM, apr 2022. doi: 10.1145/3491102.3502028 3

[4] E. Bozdag and J. van den Hoven. Breaking the filter bubble: democracy and design. *Ethics and Information Technology*, 17(4):249–265, dec 2015. doi: 10.1007/s10676-015-9380-y 2, 3

[5] L. Burbach, P. Halbach, M. Ziefle, and A. C. Valdez. Bubble trouble: Strategies against filter bubbles in online social networks. In *Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Healthcare Applications*, pp. 441–456. Springer International Publishing, 2019. doi: 10.1007/978-3-030-22219-2_33 2, 3

[6] J. Chadwick. Hfacebook bows to pressure and brings back the chronological feed so you can see updates from friends in the order they were published – but doubles down on algorithms on the 'home' page, 2022. 2

[7] D. Chakrabarti, R. Kumar, F. Radlinski, and E. Upfal. Mortal multi-armed bandits. *Advances in neural information processing systems*, 21, 2008. 2

[8] M. Cinelli, G. D. F. Morales, A. Galeazzi, W. Quattrociocchi, and M. Starnini. The echo chamber effect on social media. *Proceedings of the National Academy of Sciences*, 118(9), feb 2021. doi: 10.1073/pnas.2023301118 2

[9] M. Eslami, A. Rickman, K. Vaccaro, A. Aleyasen, A. Vuong, K. Karahalios, K. Hamilton, and C. Sandvig. " i always assumed that i wasn't really that close to [her]" reasoning about invisible algorithms in news feeds. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 153–162, 2015. 3

[10] M. Fernández, A. Bellogín, and I. Cantador. Analysing the effect of recommendation algorithms on the amplification of misinformation. *arXiv preprint arXiv:2103.14748*, 2021. 2

[11] S. Flaxman, S. Goel, and J. M. Rao. Filter bubbles, echo chambers, and online news consumption. *Public opinion quarterly*, 80(S1):298–320, 2016. 2, 3

[12] B. Flyon. Exploring bayesian bandits - an online tool. 3, 5

[13] I. A. Hamilton. Facebook whistleblower frances haugen says it's cheaper to run 'hateful' ads on the platform than other kind of adverts. 'we are literally subsidizing hate.', 2022. 2

[14] I. A. Hamilton. How to switch your facebook feed to a chronological timeline, 2022. 2

[15] T. J. Keaton and A. Sabbaghi. Visualizations for interrogations of multi-armed bandits. *Stat*, 8(1):e247, 2019. e247 sta4.247. doi: 10.1002/sta4.247 3

[16] I. Mehta. Twitter makes algorithmic timeline default on ios, 2023. 2

[17] Y. Michelle, P. Deric, and L. Ayaz. The multi-armed bandit problem. 3

[18] A. Moradipari, C. Silva, and M. Alizadeh. Learning to dynamically price electricity demand based on multi-armed bandits. In *2018 IEEE global conference on signal and information processing (GlobalSIP)*, pp. 917–921. IEEE, 2018. 2

[19] M. D. Muralikumar and M. J. Bietz. Visualizing algorithmic selection in social media. In *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing*, pp. 319–323, 2019. 3

[20] S. Nagulendra and J. Vassileva. Understanding and controlling the filter bubble through interactive visualization: a user study. In *Proceedings of the 25th ACM conference on Hypertext and social media*, pp. 107–115, 2014. 3

[21] T. T. Nguyen and H. W. Lauw. Dynamic clustering of contextual multi-armed bandits. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 1959–1962, 2014. 2

[22] R. Ookalkar, K. V. Reddy, and E. Gilbert. Pop: Bursting news filter bubbles on twitter through diverse exposure. In *Conference Companion Publication of the 2019 on Computer Supported Cooperative Work and Social Computing*, CSCW '19, p. 18–22. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3311957.3359513 3

[23] E. Pariser. *The filter bubble: What the Internet is hiding from you*. penguin UK, 2011. 3

[24] E. M. Schwartz, E. T. Bradlow, and P. S. Fader. Customer acquisition via display advertising using multi-armed bandit experiments. *Marketing Science*, 36(4):500–522, 2017. 2

[25] A. Slivkins. Introduction to multi-armed bandits, 2022. 1, 2, 3

[26] M. Song, Z. Bnaya, and W. J. Ma. Sources of suboptimality in a minimalistic explore–exploit task. *Nature human behaviour*, 3(4):361–368, 2019. 3

[27] M. O. Source. React, 2023. 4

[28] M. O. Source. Recoil, 2023. 4

[29] P. Stubley. A visual exploration of multi-armed bandit experiments, 2020. 3

[30] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 2, 3, 4

[31] Unsplash. Unsplash api, 2023. 5, 8

[32] G. van den Burg. Thompson sampling code, 2020. 4