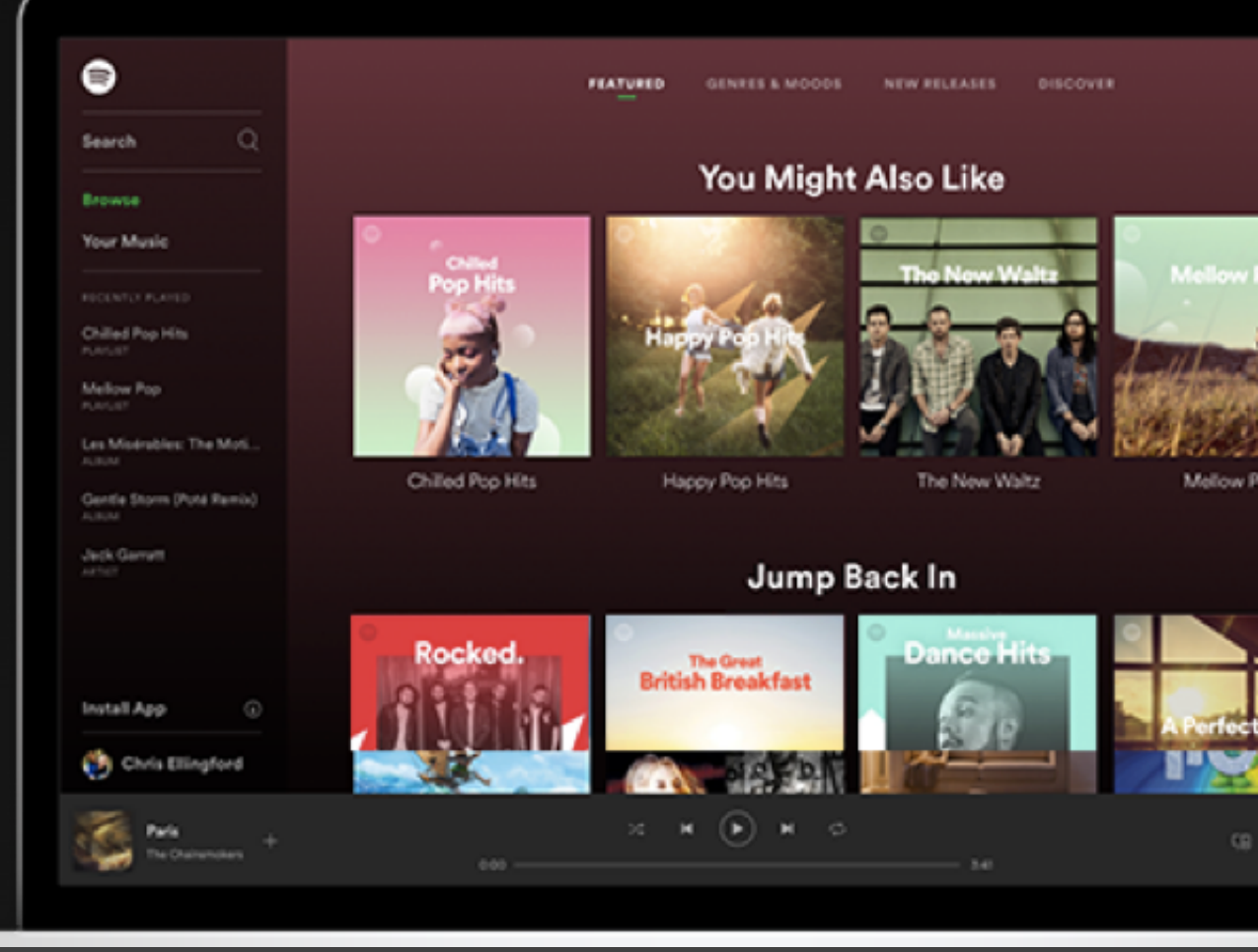
Spotify®

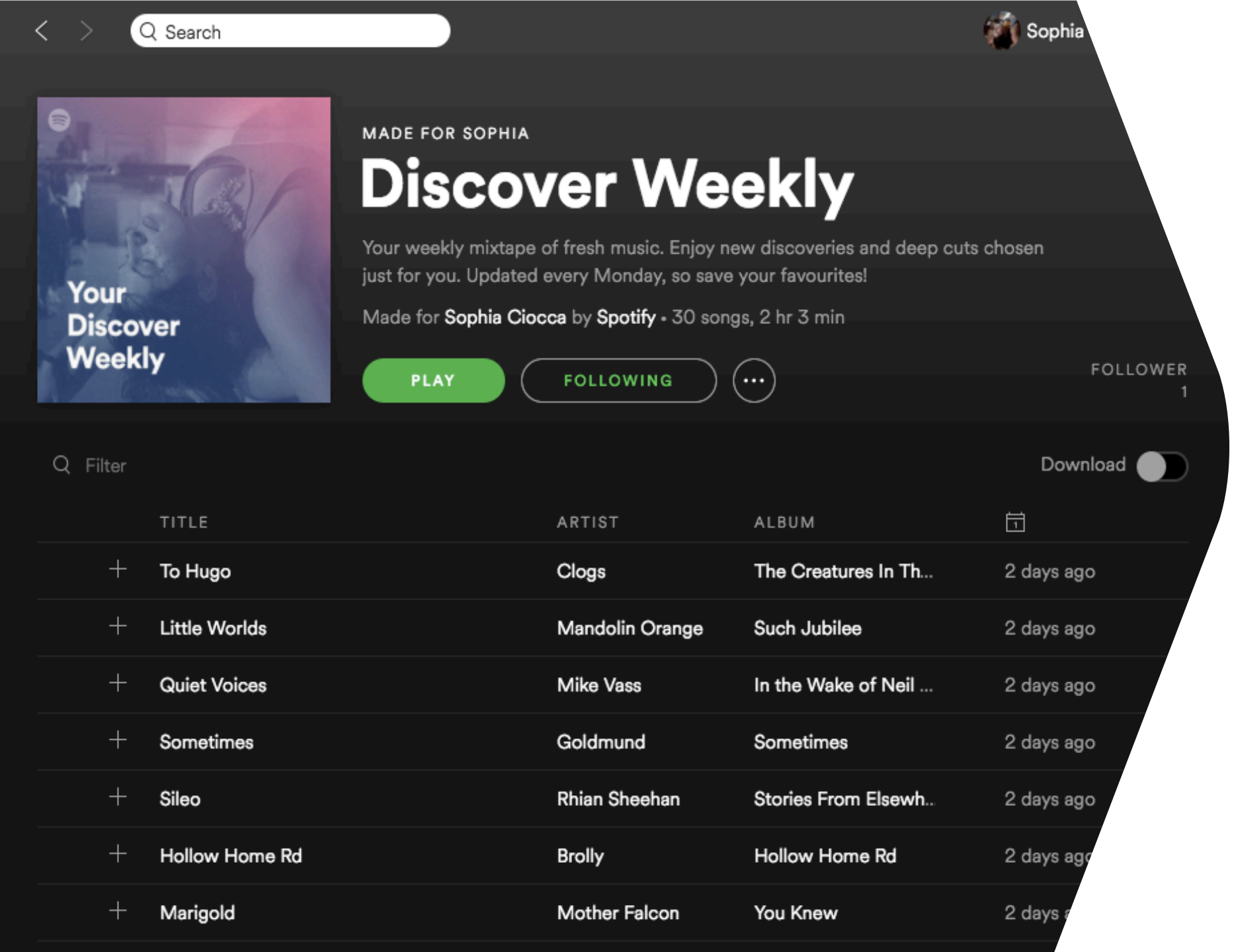
Music for everyone.



Music Recommender

Chun Wu

Dongrui Lu

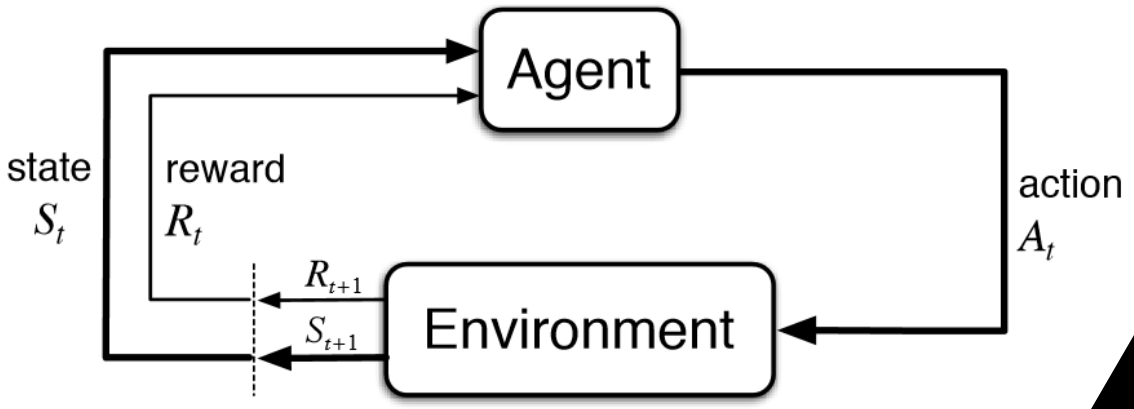


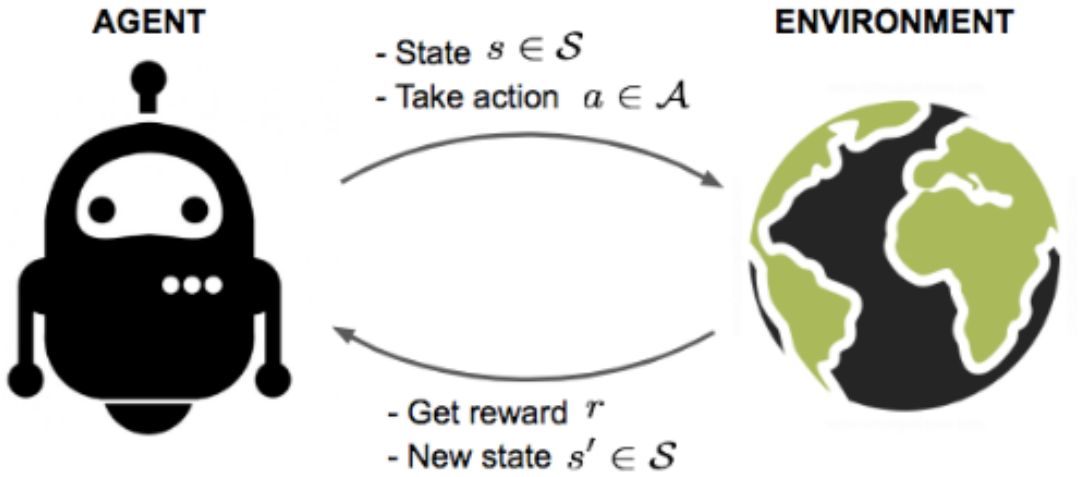
Challenge: 30M songs, 50M subscribers…
How do we recommend music to users?

- Discover
- Radio
- Related Artist
- Now Playing

We focus on **Discover Weekly**.

Discover Weekly reaches one billon tracks streamed in 10 weeks.





Overview

I. Environment

II. Agents & Performance

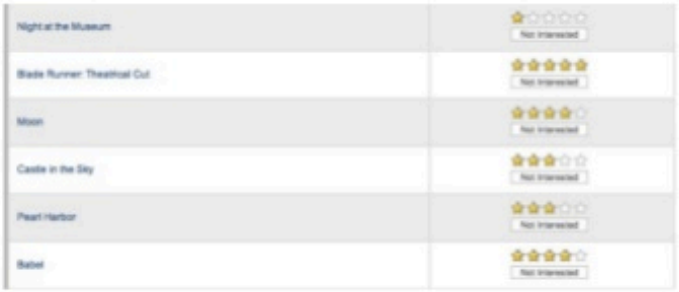
I. Environment

Explicit vs Implicit

- Let's face it, **explicit** feedback is hard to collect as they require additional input from the users. The users give explicit feedback only when they choose to do so.
- On the other hand, **implicit** data is easy to collect in large quantities without any effort from the users. The goal is to convert user behavior into user preferences.

The Netflix Problem Vs The Spotify Problem

- Netflix: Users explicitly "rate" movies



- Spotify: Feedback is implicit through streaming behavior

“Implicit” --> “Event Strength”

	user_id	item_id	listening_time_in_scale_0-5	title	duration	loudness	tempo	key	event_type	event_strength
0	196	242	2	Atmosphere Station	191.81669	-38.525	90.084	9	[Listen it frequently, View Song Information, ...	7.0

Event Type			
Listening operation	Strength	Subsequent operation	Strength
Have listened it completely	+1	Like	+1
Listen it frequently	+3	View Song Information	+1
Skip it	-1	Add it to your music	+2
Skip it frequently	-3	Download	+3
		Don’t recommend it anymore	-2

For example, Strength ("Have listened it completely” ”Add it to your music” "Download”) = 6.
Instead of representing an explicit rating, the event_strength can represent a “confidence” in terms of how strong the interaction was. Articles with a larger number of event_strength by a person can carry more weight in the score.

Reward

We want to recommend a playlist of n_arms songs which have never been heard by the user.

Reward = style_matching_score +
estimated_listening_time_bonus +
estimated_event_bonus

Song style characterized by **loudness, tempo, key**

Style matching score: Dot product of item embedding matrix (song style) & user embedding matrix (user's style preference)

Estimated listening time bonus and **estimated event bonus** are determined by

1. The **similarity** between all the songs he has listened to and the candidate song for recommendation
2. His **listening time** spent on the songs with weights decided by the song similarity
3. The previously mentioned **event strength** he had for each song with weights decided by the song similarity .

3 parts are of equal importance and are rescaled to have the same range (between 0 and 1)

II. Agents & Performance

- Random Agent (Select a song randomly among the available ones)
- UCB Agent
- ALS Agent (Collaborative Filtering)
- Linear UCB Agent

Algorithm LinUCB with disjoint linear models:

- Inputs: α
- For t = 1,2,3,...,T do
  - Observe features of all arms a $\in A_t : x_{t,a} \in \mathbb{R}^d$
  - For all $a \in A_t$ do
    - if a is new then
      - $A_a \leftarrow I_d$ (d-dimensional identity matrix)
      - $b_a \leftarrow 0_{dx1}$ (d-dimensional zero matrix)
    - end if
    - Compute $\hat{\theta}_a \leftarrow A_a^{-1} b_a$
    - Compute $p_{t,a} \leftarrow \hat{\theta}_a^{\ T} x_{t,a} + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a})}$
  - end for
  - Choose arm $a_t = argmax_{a\$\in A_t} p_{t,a}$ with ties broken arbitrarity, and observe a real-value
  - Update $A_{a_t} \leftarrow A_{a_t} + x_{t,a_t} x_{t,a_t}^T$
  - Update $b_{a_t} \leftarrow b_{a_t} + r_t x_{t,a_t}$

end for

Using Sherman–Morrison formula:

To compute the inverse of the sum of an invertible matrix A and the outer product, $uv^\top$, of vectors u and v.

$$(A + uv^\top)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}$$

So that we only need to inverse the covariance matrix during initialization.

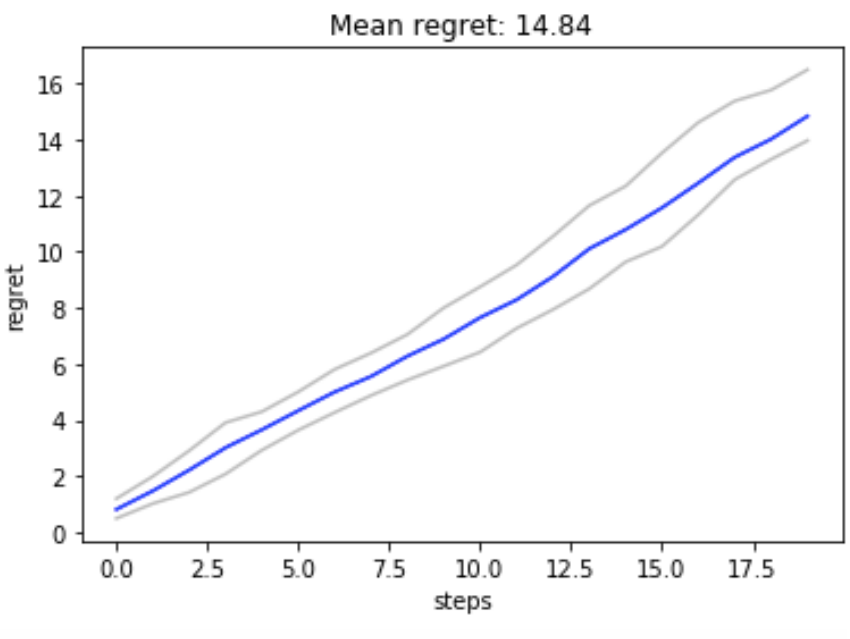
LinUCB is a generic contextual bandit algorithm.

Experiments & Results

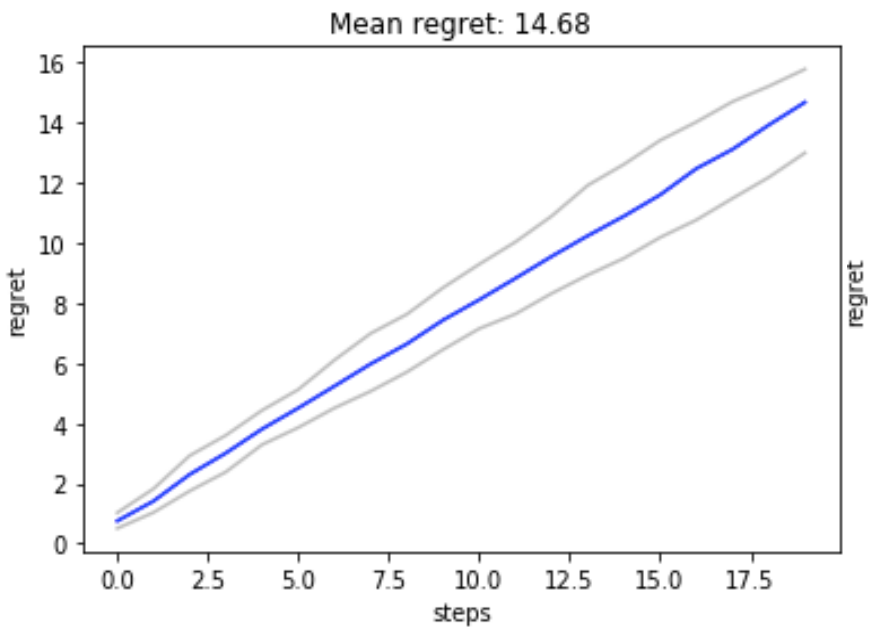
Parameters:

- 943 users
- 1682 songs
- 100000 rows
- 5 keywords per song: title, duration, loudness, tempo, key
- 9 types of event, event strength and listening time per user per song

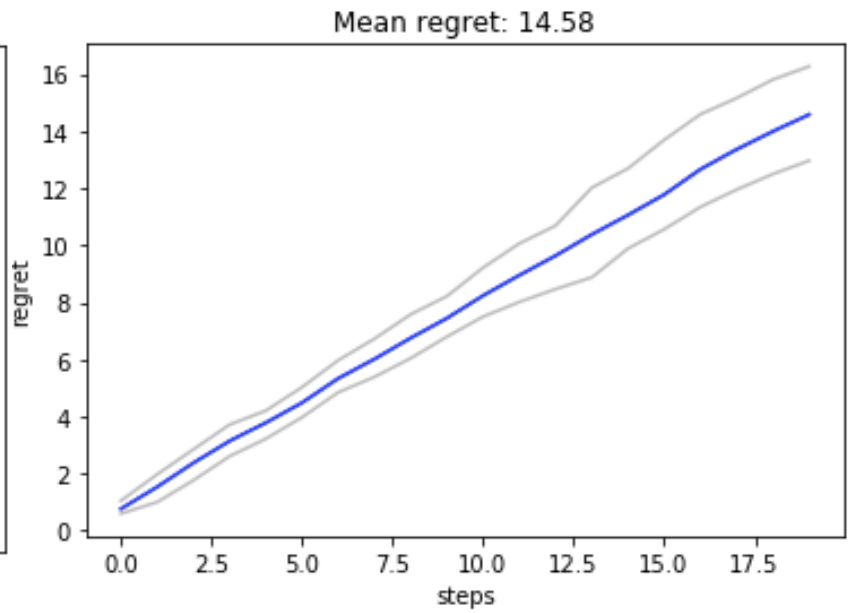
Regrets



Random Agent



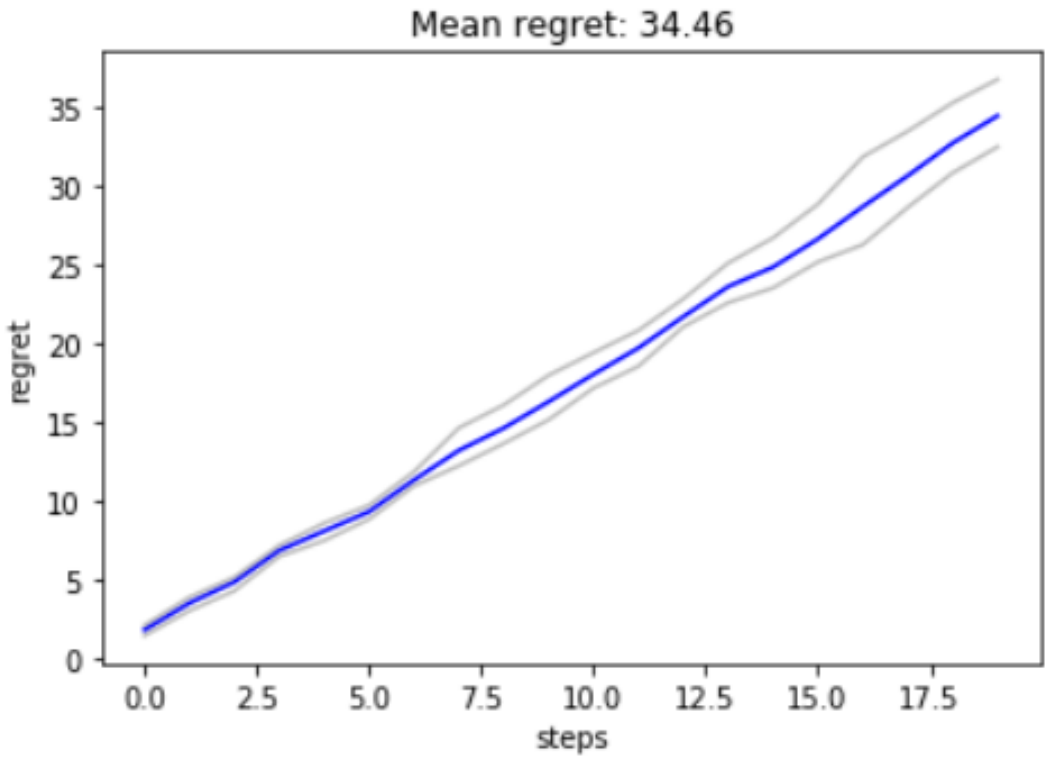
UCB Agent



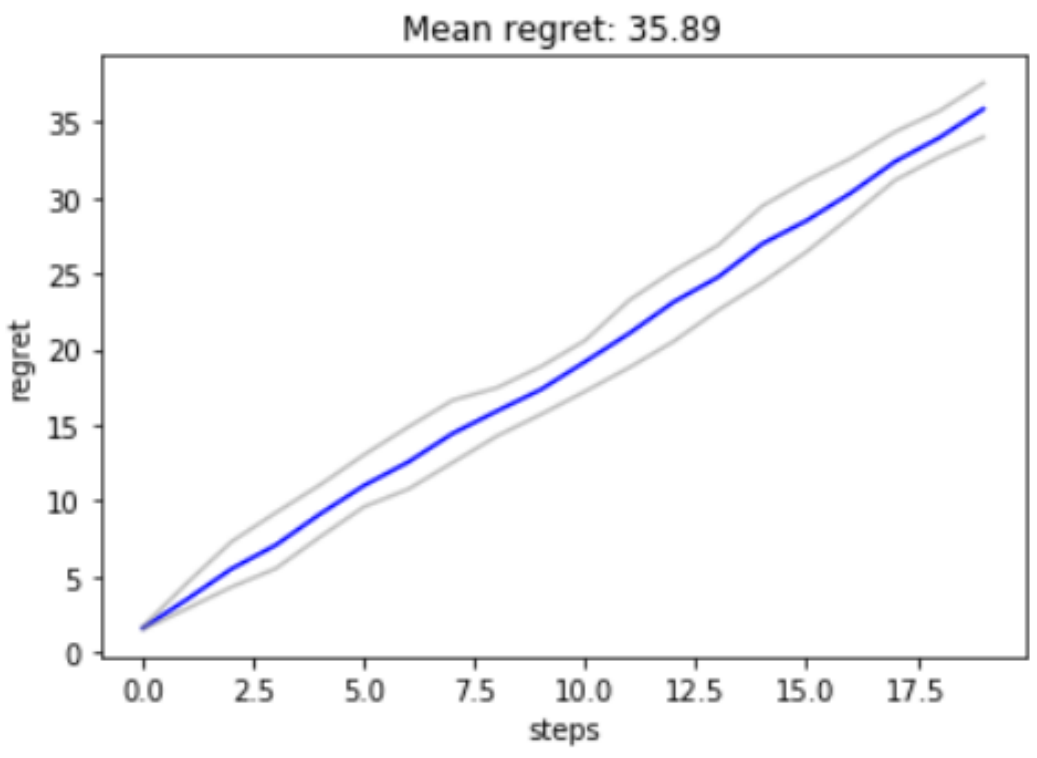
Linear UCB Agent

Due to the long running time and limited time , we only performed a limited number of tests

Random Agent



ALS Agent



Since we recommend only the songs totally new to a user, we generated random scores in the rating matrix for the 'user-item' pair in the dataset. There are fewer real history scores in the rating matrix than the fake random scores, therefore ALS agent performs almost the same as a random agent (bad quality input). What's more, we added a noise term in the environment’s reward calculation. This change increases the regret.

Thank you!