

# Reinforcement Learning for Recommender Systems

From Contextual Bandits to Slate-Q

---

**Christy Bergman, DevAdvocate Ray RLib**

<https://linkedin.com/in/christybergman>

christy@anyscale.com

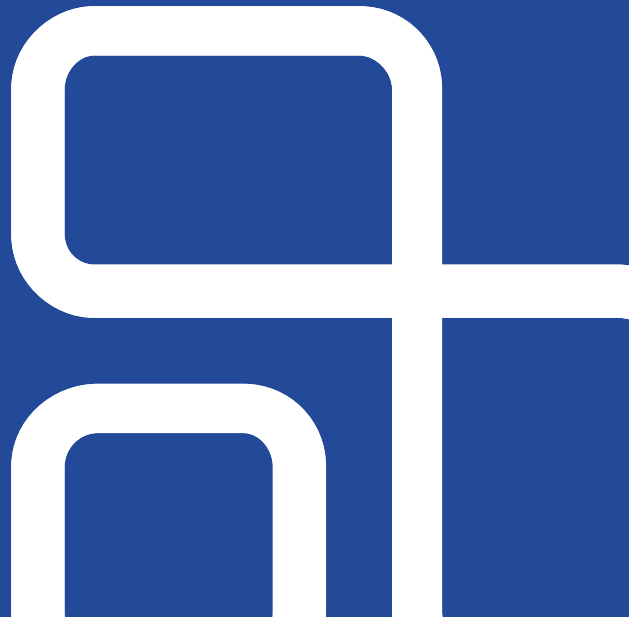
@cbergman on Twitter

**Avnish Narayan, Software Engineer Ray RLib**

<https://linkedin.com/in/avnishn>

avnish@anyscale.com

@narayan\_avnish on Twitter



# Overview of the tutorial

25 min: Intro to RL and why use RL to solve Recommender Systems?

10 min: Break/ Q&A

25 min: RLlib's Contextual Bandits Algorithms Applied To Recommender Systems

10min: Break/ Q&A

10 min: RLlib's SlateQ Applied To Recommender Systems

---

**~1.5h**



# What is Reinforcement Learning (RL)?

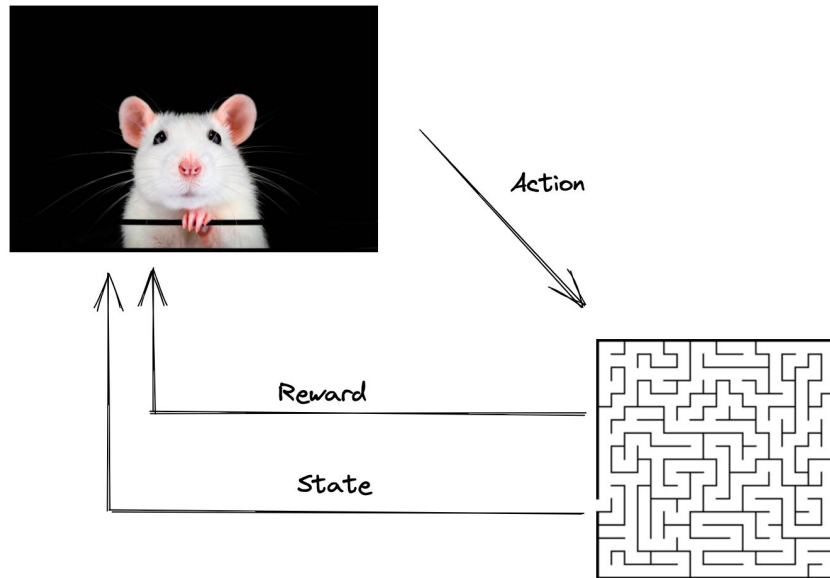
We want to teach a mouse to get to the center of a maze ... how can we do that?

Place bits of cheese in the maze along paths that we **do** want the mouse to take.

Place bits of poison along paths that we **don't** want the mouse to take.

Let the mouse repeatedly explore the maze for a fixed time from different random starting points.

Eventually the mouse learns all paths from everywhere in the maze that maximize cheese and minimize poison.





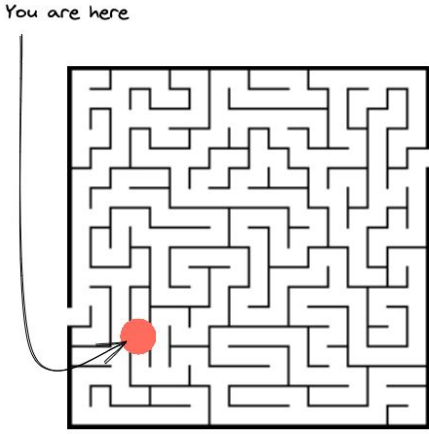
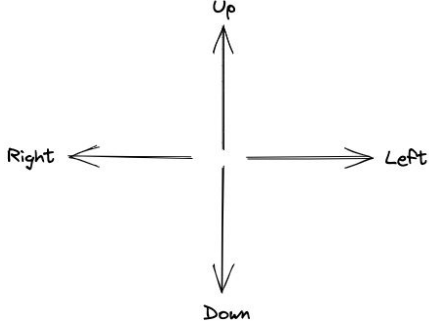
# What is Reinforcement Learning (RL)?

The mouse can be at different positions in the maze. We call this its **state** or observation.

The mouse can move up down left and right in the maze. We call this its **actions**.

There are terminal states in which the mouse no longer needs to act, such as if the mouse enters the center of the maze.



For every action that mouse takes at its current state to transition into a new state we call this a **time step**.

State	Action
	



# What is Reinforcement Learning (RL)?

The mouse gets cheese or poison depending on whether its taking the best action at its state. We call this its **reward**.

Reward	
<p>Cheese :)</p> 	<p>Poison :(</p> 

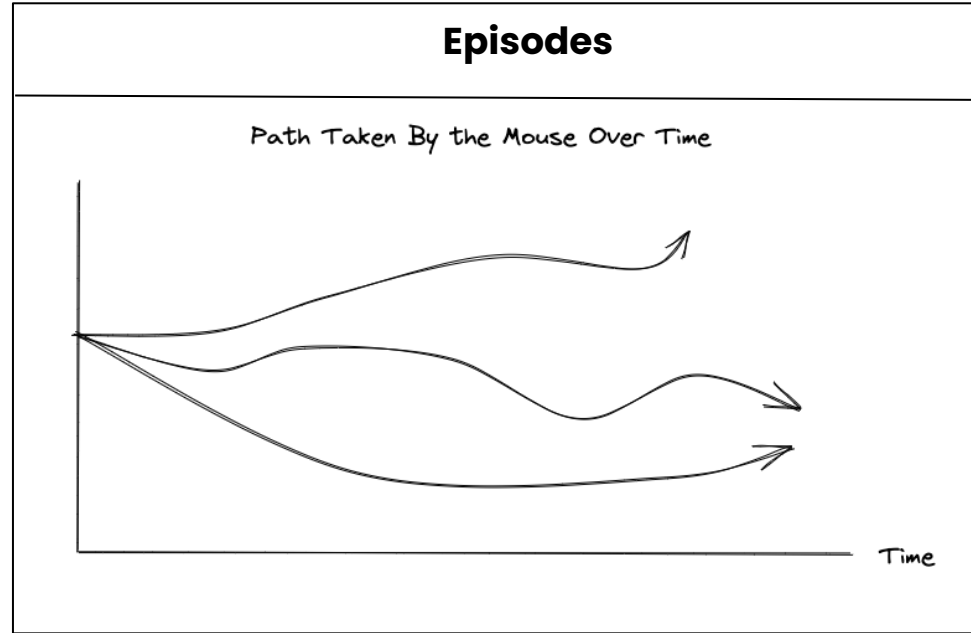


# What is Reinforcement Learning (RL)?

We let the mouse explore and follow the cheese trail for some time but eventually **reset** it to a new random starting state after some time steps.

We call the sequence of time steps that the mouse creates from its starting state till we reset it an **episode**.


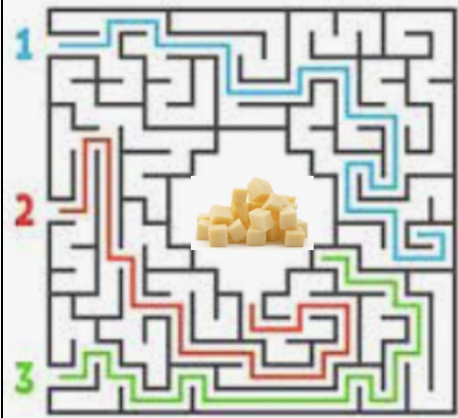
When an episode terminates, say from the mouse reaching its exploration time limit, or by reaching the center of the maze, we say that its **done** signal is true.





# What is Reinforcement Learning (RL)?

We can use **Reinforcement Learning** to teach the mouse, or our **agent**, to take the sequence of actions at the states which it visits so that we maximize the total reward per episode in the maze **environment**.

Agent	Environment
	

# **Can We Use The Framework Of The Toy Problem And Relate It To A Real World Problem?**





# RL and Self Driving

## Autonomous Vehicle vision-based controller

**State:** camera images, lidar sensor readings

**Action:** steering wheel angle, pressure on brake, pressure on gas pedal

**Reward:** +1 driving safely at the current time step, 0 otherwise

**Done:** when the trip ends or if an unsafe driving incident occurs at the current time step.

Using RL, train the car-agent to maximize the amount of safe driving behavior during a trip



# Recommender Systems and RL



# What Are Recommender Systems?

What	Where
Serve up <b>personalized content</b> , based on users <b>interactions</b> , in order to provide an <b>improved experience</b> for each user.	<ul style="list-style-type: none"><li>• Games</li><li>• Rideshare</li><li>• Purchasing apps (B2B, B2C)</li><li>• Websites, web apps</li><li>• Mobile apps</li><li>• Chatbots</li><li>• Call centers</li></ul>




# RL and Recommender Systems

All Seminars Machine learning Tiny Desk Concerts Conversation Live Music San Francisco Symphony Hiking Violins APIs Jazz Wood Nature Rapping Art




 **Jeremy Howard Interviews Kaggle Grandmaster Sanyam Bhutani**  
Jeremy Howard  
2.6K views • 1 day ago




 **Track Sprint Workout: How To**  
Jade Teta  
24K views • 10 years ago




 **AAAI 20 / AAAI 2020 Keynotes Turing Award Winners Event / Geoff Hinton,...**  
ICML IJCAI ECAI 2018 Conference Videos  
37K views • 2 years ago



 **Hiking 60 Miles Alone in Hornstrandir Iceland**  
Kraig Adams  
5M views • 2 years ago



 **Russia invades Ukraine LIVE | DW News livestream | Headline news from around the world**  
DW News  
23K watching  
**LIVE NOW**



 **Phantom Thread - House of Woodcock (Official Audio)**  
Nonesuch Records  
1.3M views • 4 years ago



 **Think Fast, Talk Smart: Communication Techniques**  
Stanford Graduate School of Business  
25M views • 7 years ago



 **Hiking 60 Miles Alone in Hornstrandir Iceland**  
Kraig Adams  
5M views • 2 years ago



# RL and Recommender Systems

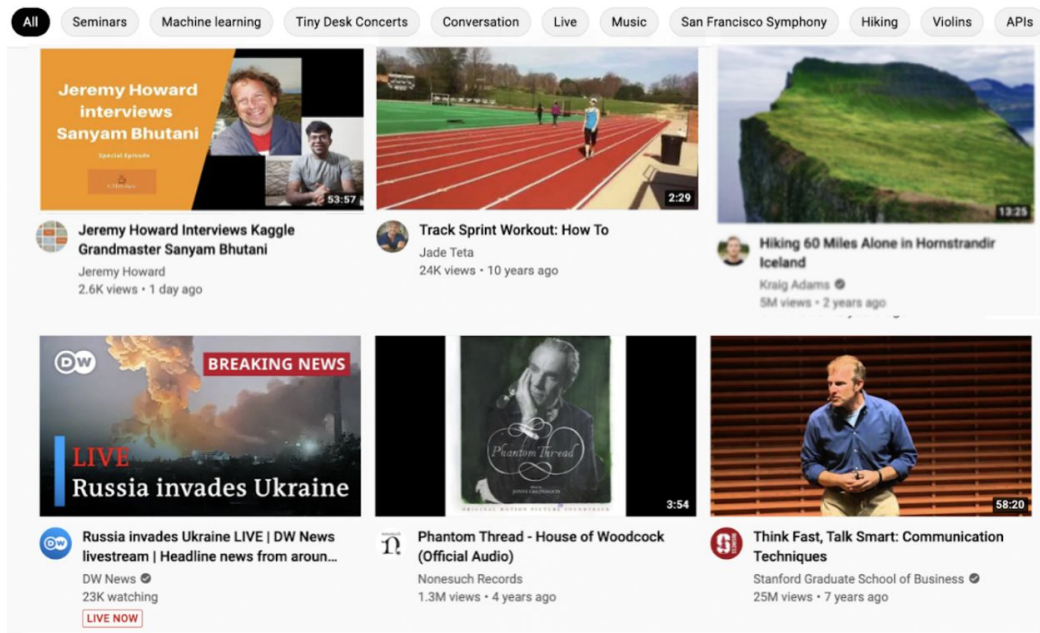
**State:** user actions, features about the user, user watch history

**Action:** recommended videos

**Reward:** +1 if user clicks a piece of content AND long-term satisfaction has increased at the current time step, 0 otherwise

**Done:** True if we've made the maximum amount of recommendations in a user's session on the website.

Using RL, train a recsys agent to maximize the click through rate and long term satisfaction of users.

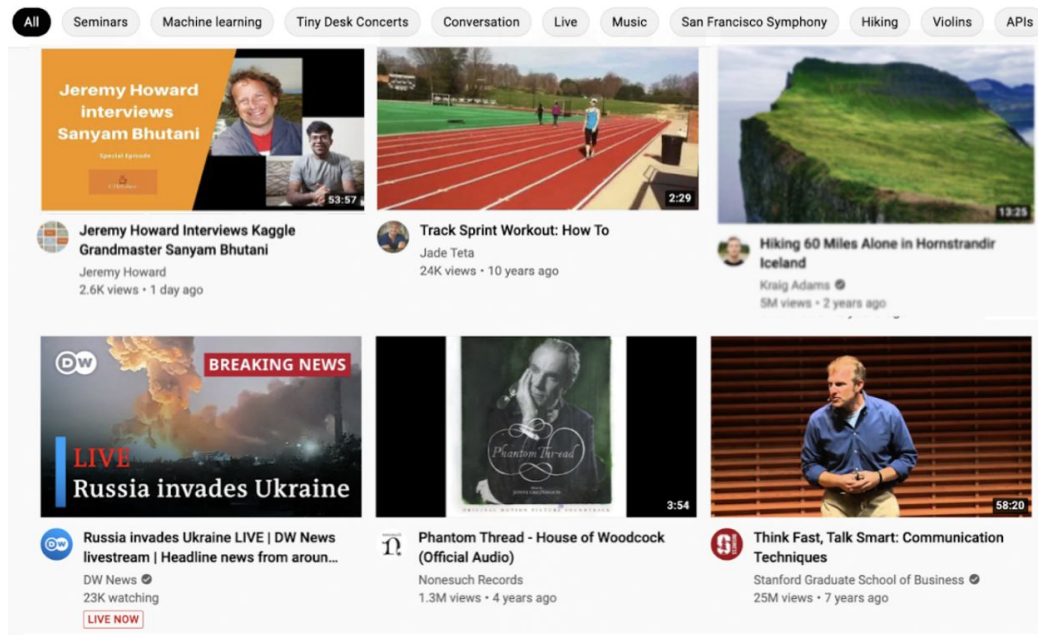




# Supervised Learning (SL) Approach to Recommender Systems

## High-level steps to build a Recommender System using SL

- 1) Train a model using supervised learning that takes as input some features about the user (demographic, interests) and outputs some categories or topics the user would be interested in.
- 2) Use a search ranking algorithm on the categories to produce video recommendations.





# Why use RL over SL for Recommendations?

RL Algorithms will train recsys agents that maximize the quality of all the recommendations that are given. SL trains agents to give pointwise recommendations.

With SL there is no concept of long term satisfaction of a user embedded in to training. This must be hardcoded into the recommender system as a heuristic.

RL can be used to train an end to end recommendation system whereas SL will have brittle hand-engineered features in the system.

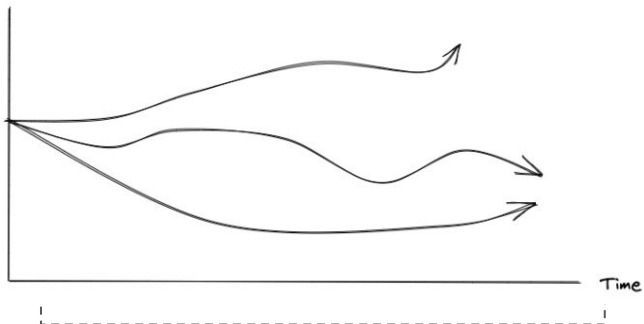


# Deep RL and RLib code abstractions

## Recommender Systems

### Episodes

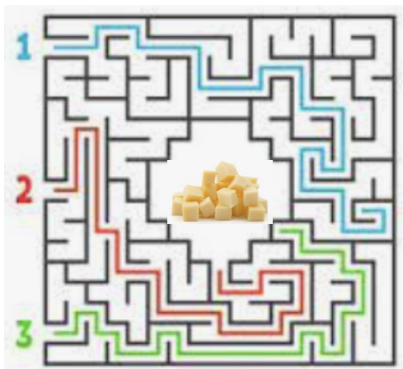
Path Taken By the Mouse Over Time



Agent

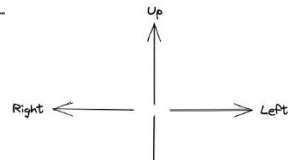


Policy (deep-RL, this is a neural network)



Environment

Actions



State, Rewards



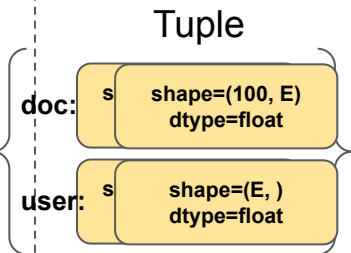


# RL Ingredients and RLib code abstractions

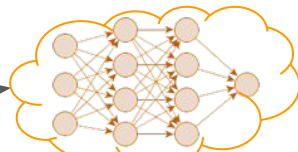
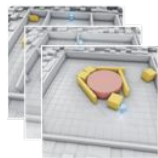
## Recommender Systems

### Episodes

observation



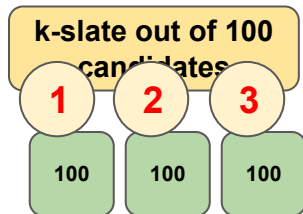
(User features, Doc features)



**Agent** (deep-RL, this is a neural network)

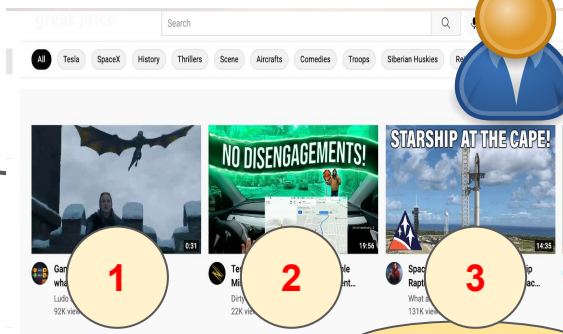


**Actions**

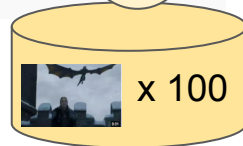


$100 \times 99 \times 98 = \sim 1\text{M}$  unique slates

**State, Rewards**

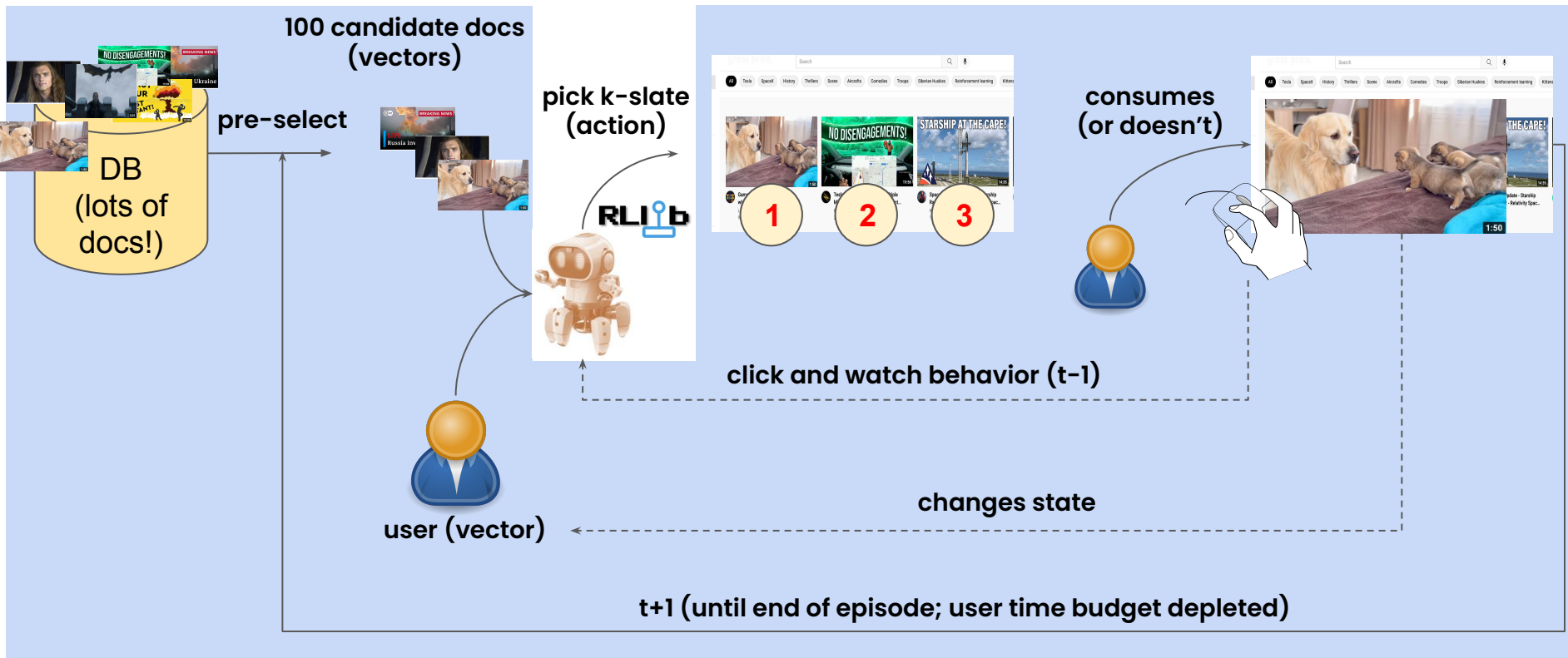


**Environment**





# A Recommender System in Action

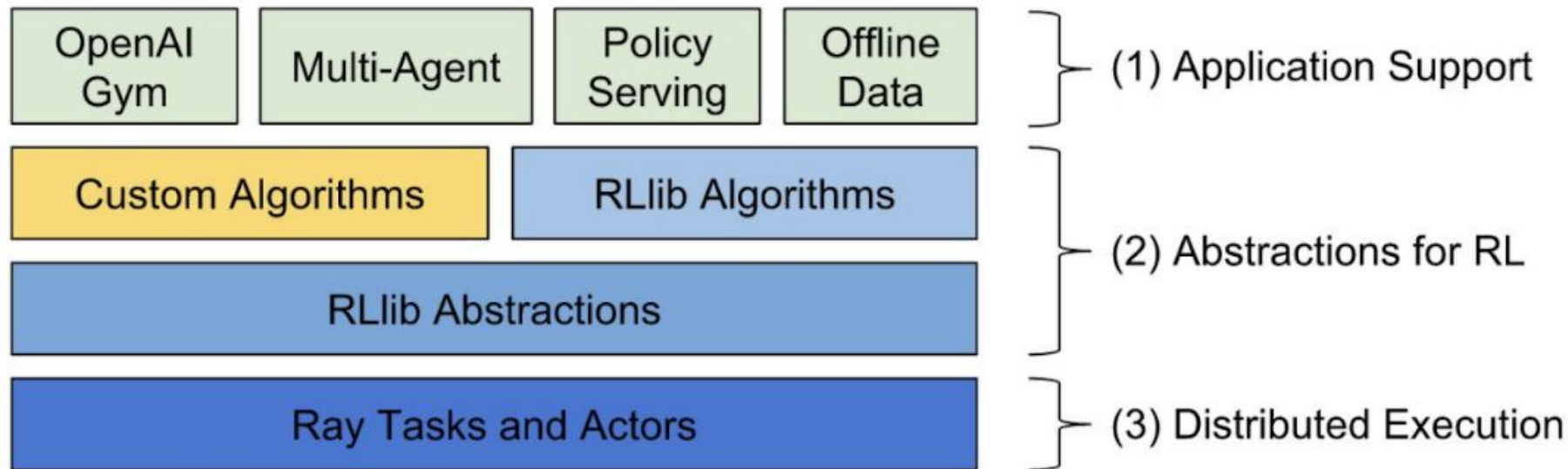


# **What Software Should I Use To Do RL?**



# RLlib Parallelize and Distribute at Scale

Ray provides a common framework for distributed RL algorithms using asynchronous parallel Tasks and Actors.





# Why Should I Use RLlib

- Support for both TF2 and PyTorch



# Why Should I Use RLlib

- Support for both TF2 and PyTorch
- Almost any algo can learn in a multi-agent setting.



# Why Should I Use RLlib

- Support for both TF2 and PyTorch
- Almost any algo can learn in a multi-agent setting.
- 25+ available algorithms: model-free, model-based, offline RL, meta RL, evolutionary strategies.



# Why Should I Use RLlib

- Support for both TF2 and PyTorch
- Almost any algo can learn in a multi-agent setting.
- 25+ available algorithms: model-free, model-based, offline RL, meta RL, evolutionary strategies.
- Benchmarks and tuned examples. Gives you a leg-up, someplace to start with hyperparameter choices.





**And why should I use RLlib?**  
Because these companies here do!  
Thx for presenting at Ray- and RL Summits!



TWO SIGMA



SIEMENS



**K**

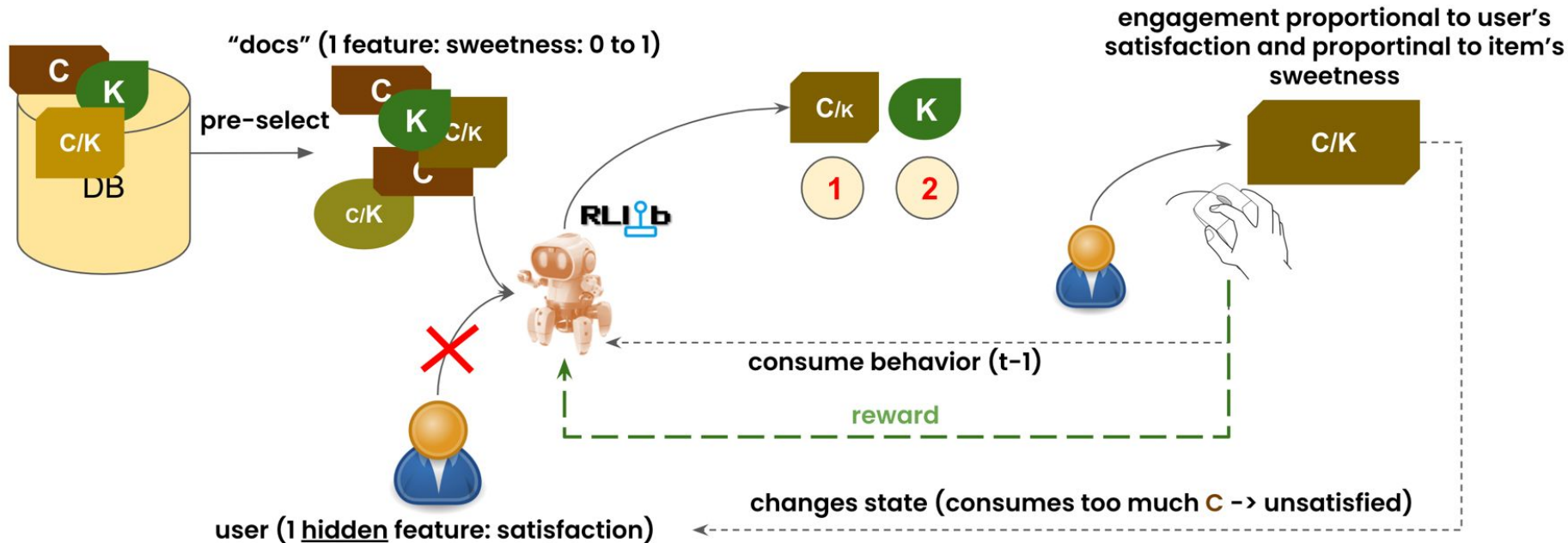
Kale: Sweetness=0.0  
Low engagement; high  
satisfaction

**C**

Chocolate: Sweetness=1.0  
High engagement; low  
satisfaction

# Google RecSim

## The “Long Term Satisfaction” Problem



<https://github.com/google-research/recsim>

```
$ pip install recsim
```

**10 min Break :)**  
**Then ... our Colab Notebook**

**GOOGLE COLAB:**  
**[bit.ly/odsc\\_rllib\\_tutorial](https://bit.ly/odsc_rllib_tutorial)**

**Github:**  
**[bit.ly/odsc\\_rllib\\_github](https://bit.ly/odsc_rllib_github)**

---