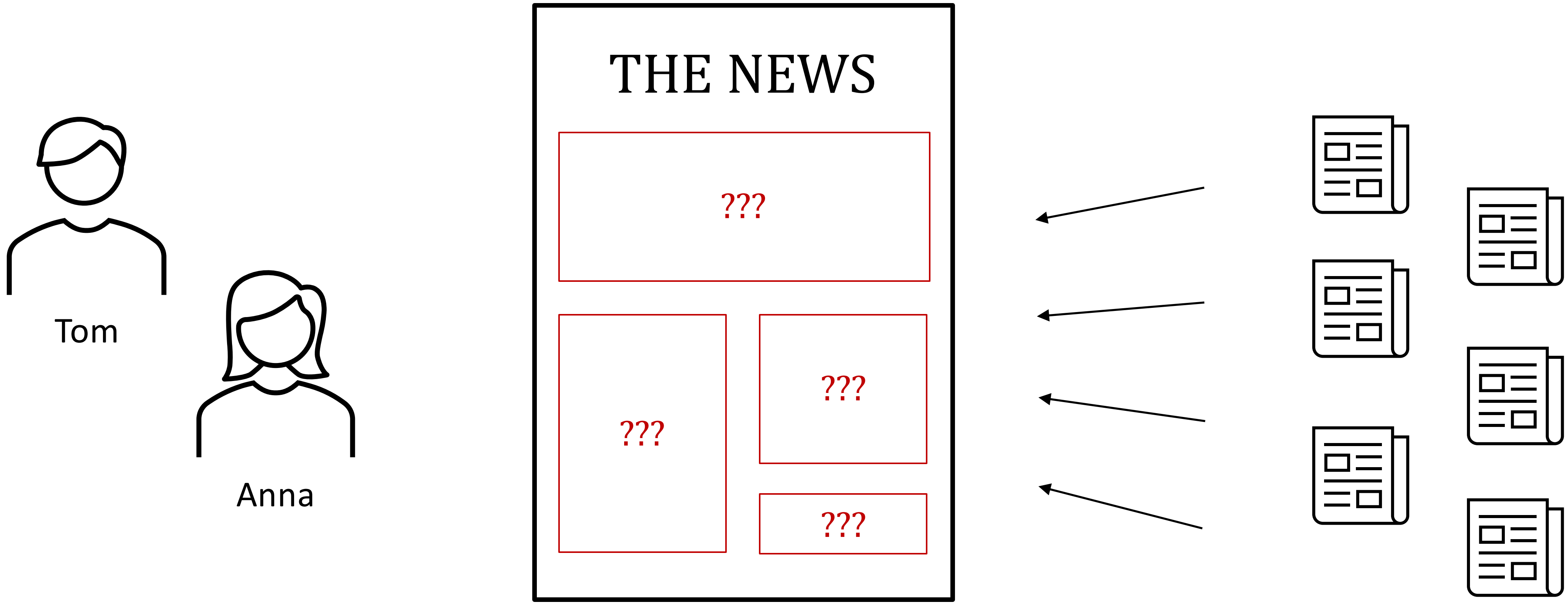
Vowpal Wabbit

A year in review and looking ahead in an LLM world

Agenda

- What is Vowpal Wabbit (a quick overview)
- Learned orchestration with Vowpal Wabbit and LLMs
- Embedding Vowpal Wabbit in your applications
- What is new in Vowpal Wabbit

Scenario: Personalized news recommendation



The contextual bandit problem

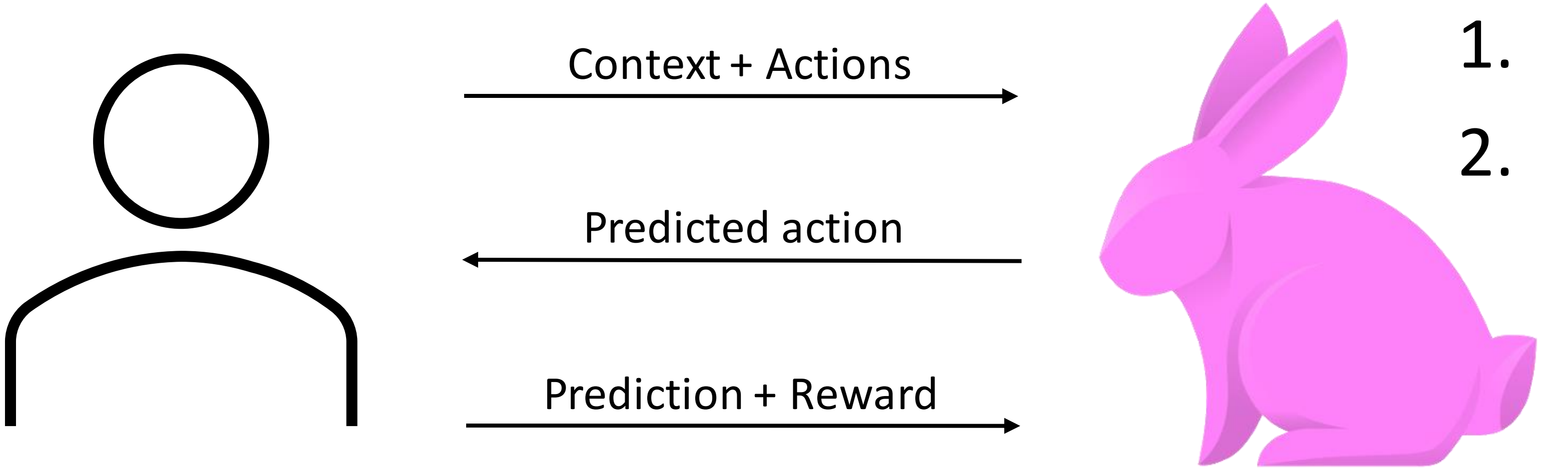
- Receive context and possible actions
  - User, location, time of day, etc.
- Pick an action
  - Recommend an article
- Receive reward for that action
  - User clicks or ignores recommendation
- Repeat

The contextual bandit problem

- Receive context and possible actions
  - User, location, time of day, etc.
  - **Context provides all the info you have**
- Pick an action
  - Recommend an article
  - **Action is independent of past and future contexts**
- Receive reward for that action
  - User clicks or ignores recommendation
  - **No information gained regarding actions not taken**
- Repeat
  - **Exploration vs exploitation tradeoff**

VOWPAL WABBIT

Vowpal Wabbit workflow



1. Predict
2. Learn

How to use Vowpal Wabbit

vw.exe

C++

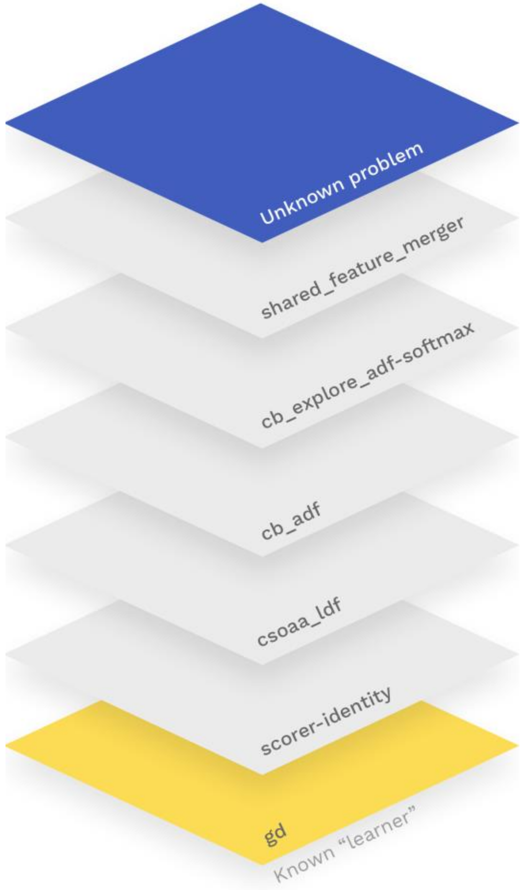
Python
(new and improved!)

Java

C#

Javascript
(WebAssembly)

The Vowpal Wabbit reduction stack



- VW is a sequence of learners
- Examples enter at the top of the stack
- At each layer, a VW learner reduces it to a simpler problem
- Bottom learner implements well-known algorithms like SGD
- Prediction moves up the stack

What reductions should I use?

- `--cb_explore_adf`
  - Contextual bandits + exploration + action dependent features
  - Recommend a news article
- `--ccb_explore_adf`
  - Conditional contextual bandits
  - Recommend top 10 news articles
- `--slates`
  - Like CCB, except with disjoint action sets
  - Recommend one article for each topic
- `--cats`
  - Continuous action, as opposed to a discrete choice
  - Set the temperature on a smart thermostat

Vowpal Wabbit example format

```
shared |User user=Tom time_of_day=morning
|Action article=politics length:100
|Action article=sports length:500
|Action article=finance length:200
```

```
shared |User user=Anna time_of_day=evening
|Action article=politics length:100
|Action article=sports length:500
0:-1:0.5 |Action article=finance length:200
```

Shared features: applies to all possible actions

Action dependent features: unique for each action

Namespaces: defines feature groups for interactions

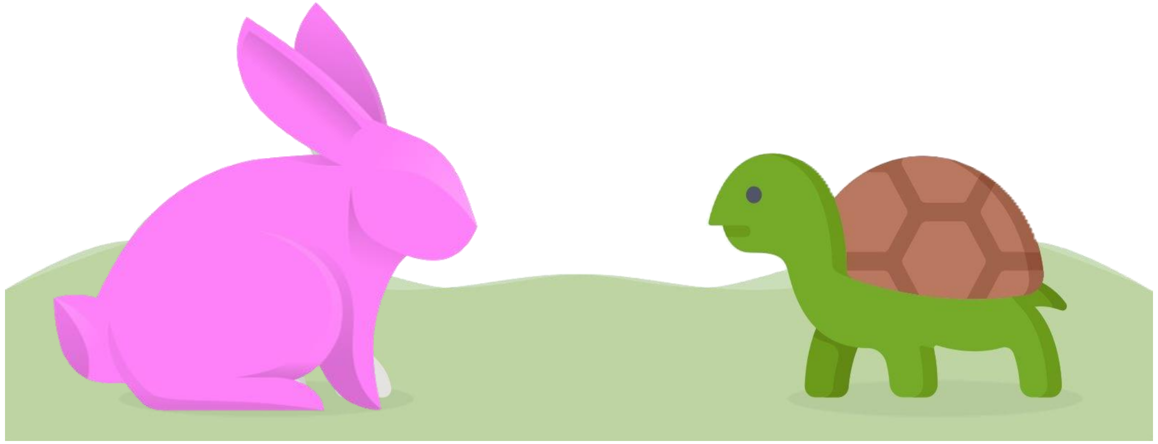
Features: can be strings or numbers

Label: action:cost:probability for the chosen action

Live demo!

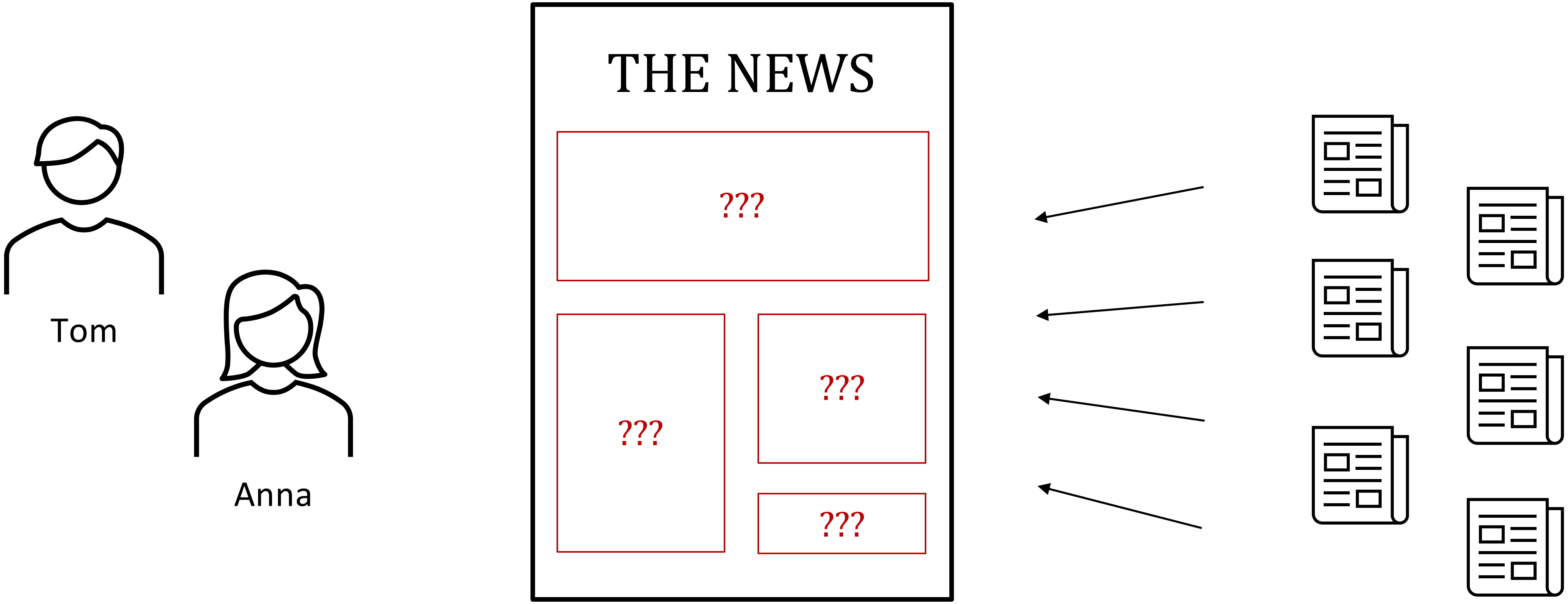
https://github.com/VowpalWabbit/workshop/blob/master/ICML2023/tom_and_anna.ipynb

Vowpal Wabbit ♡ LLMs

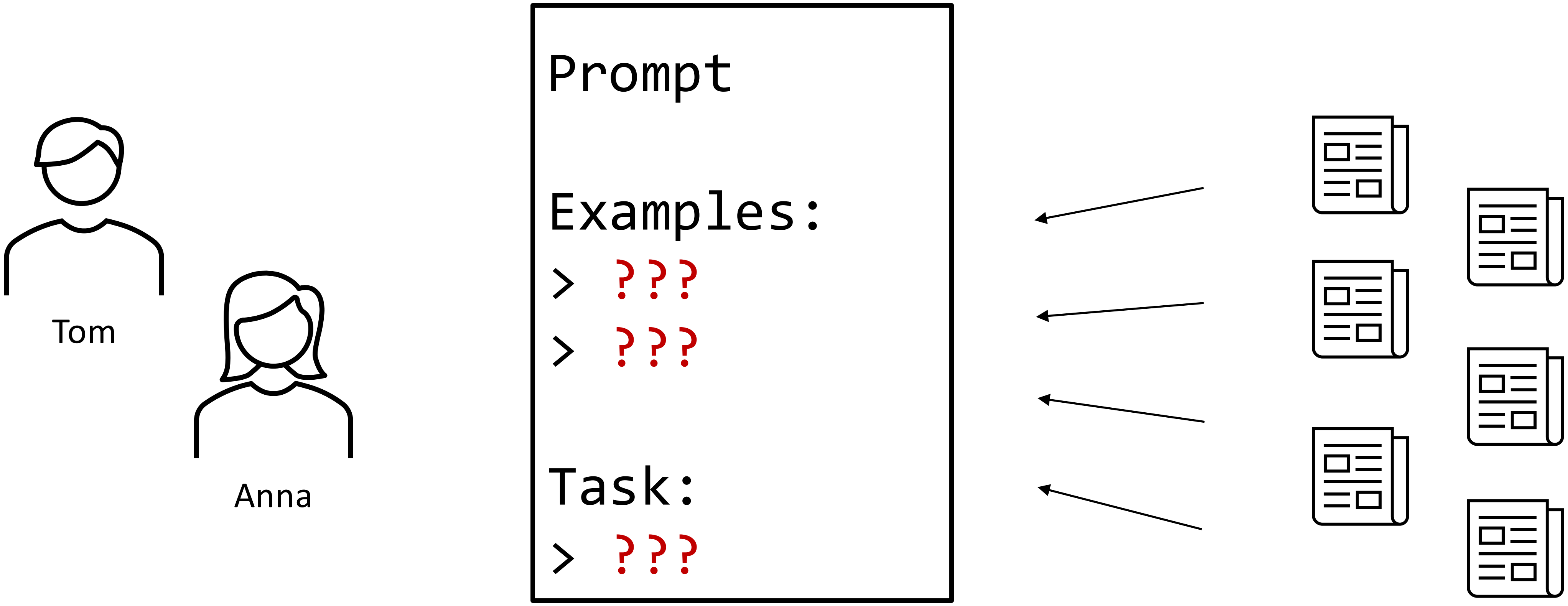


CPU training and inference vs. expensive GPU hardware
Learning from interactions vs. fixed pretrained models
Intelligent decision making vs. prompt-driven completions
Fast vs. slow

Scenario: Personalized news recommendation



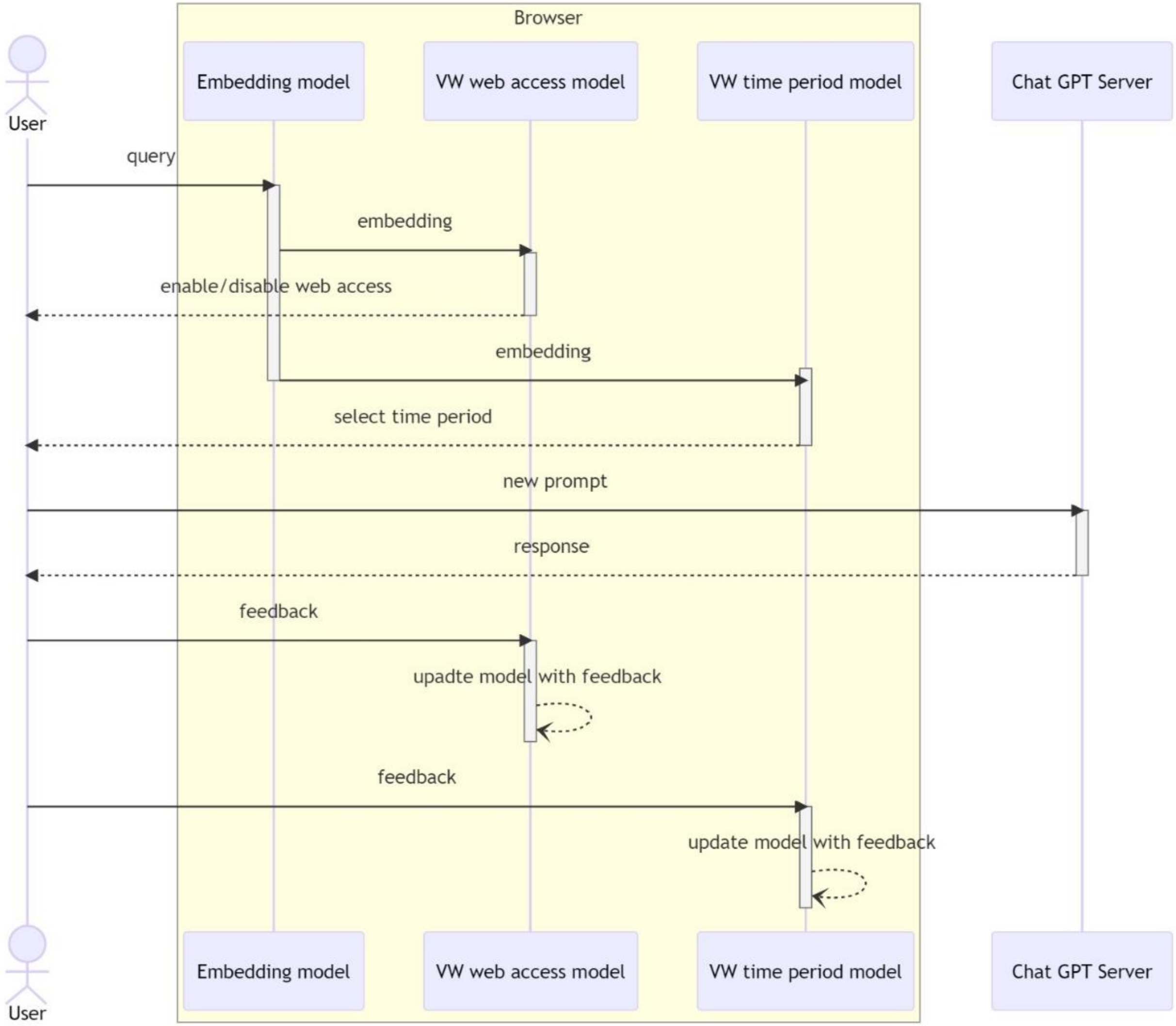
Scenario: Personalized content generation



Learned orchestration with VW and LLMs

Demo

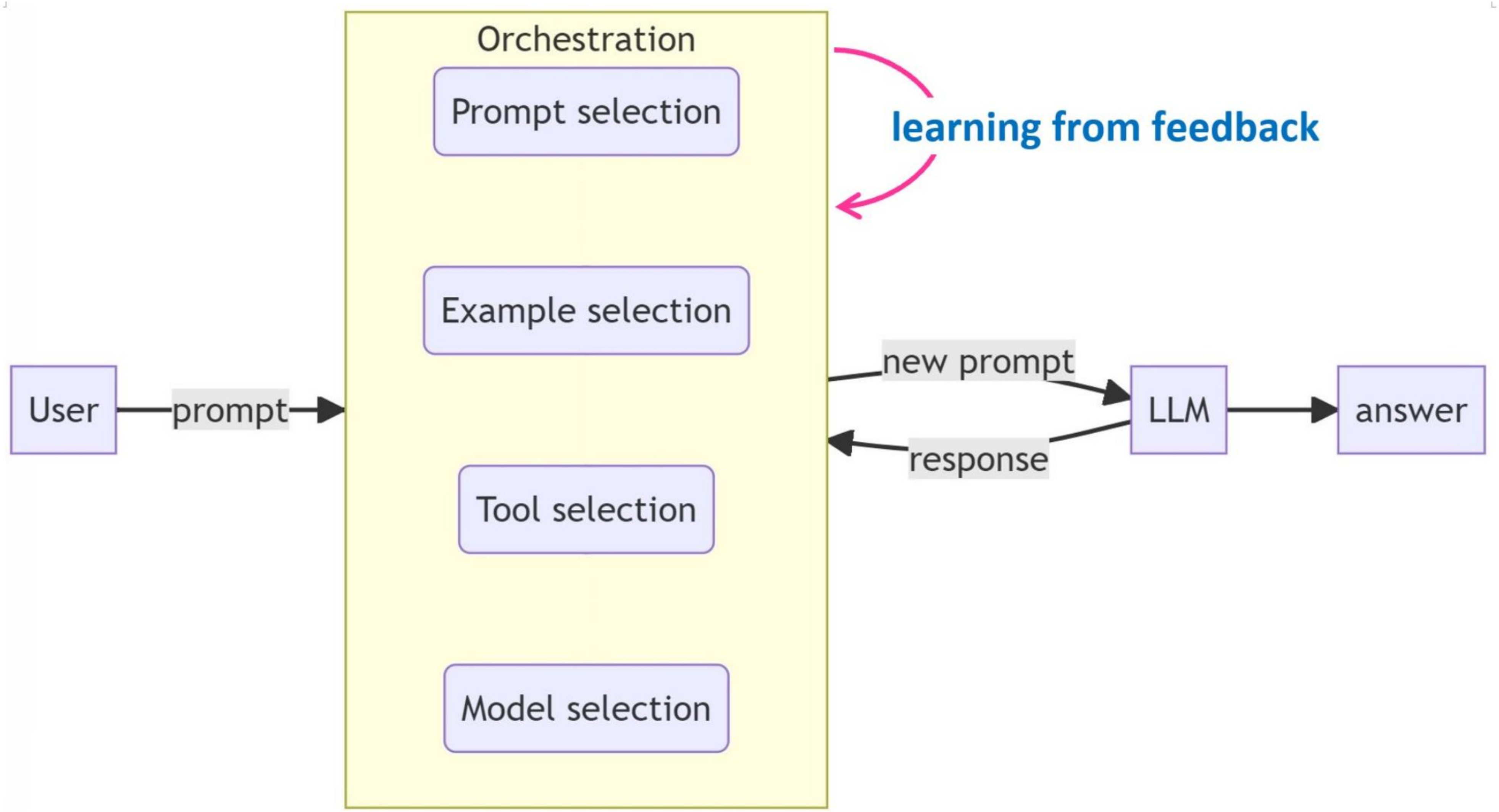
AdaptiveChat



Introducing Learned Orchestration

What Is Learned Orchestration?

What Is Learned Orchestration?



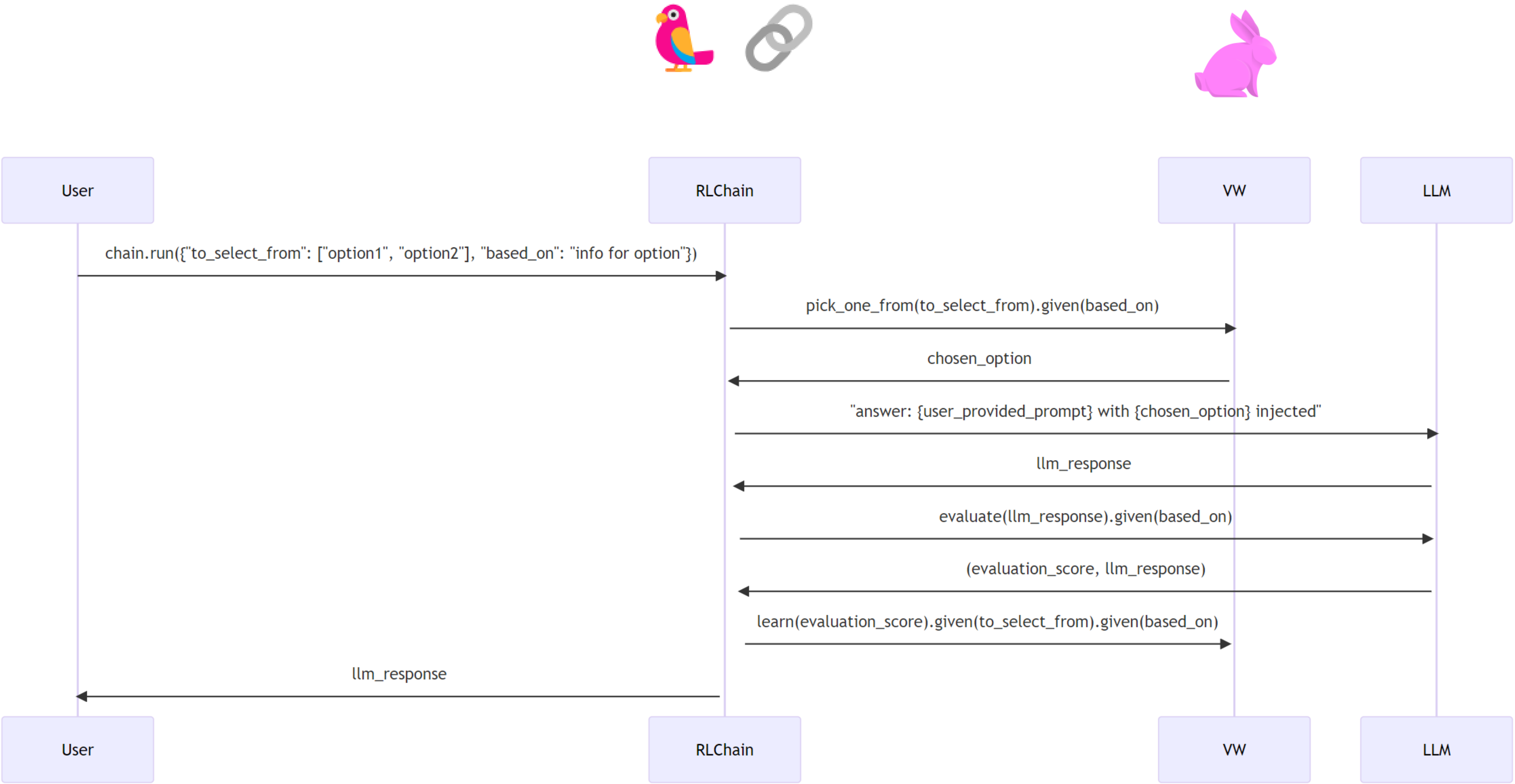
RLChain in langchain

🐇➕🦜🔗

- A langchain chain that has VW embedded into to
- Chain selects from a set of variables to inject into the prompt
- Chain learns to inject better variables into the prompt with time

- Learning happens via reward:
  - auto-reward by having the LLM check VW's output
  - user defined reward function for more specific use cases

https://github.com/VowpalWabbit/rl_chain


User
RLChain
VW
LLM
chain.run({"to_select_from": ["option1", "option2"], "based_on": "info for option"})
pick_one_from(to_select_from).given(based_on)
chosen_option
"answer: {user_provided_prompt} with {chosen_option} injected"
llm_response
evaluate(llm_response).given(based_on)
(evaluation_score, llm_response)
learn(evaluation_score).given(to_select_from).given(based_on)
llm_response
User
RLChain
VW
LLM

Example experiments

- Example selector
  - In context examples clarify the objective
  - Natural Language to SQL
    - Using spider dataset, standard benchmark for query translation
  - Use VW to select examples from ~5.5k available examples
  - Uses Large Action Space algorithm

https://github.com/VowpalWabbit/rl_chain

Example experiments

- Prompt Selector
  - Selecting different commands and placing them in different places in the prompt
  - Use VW to learn what prompt variables are better in which places
  - Slates scenario

https://github.com/VowpalWabbit/rl_chain

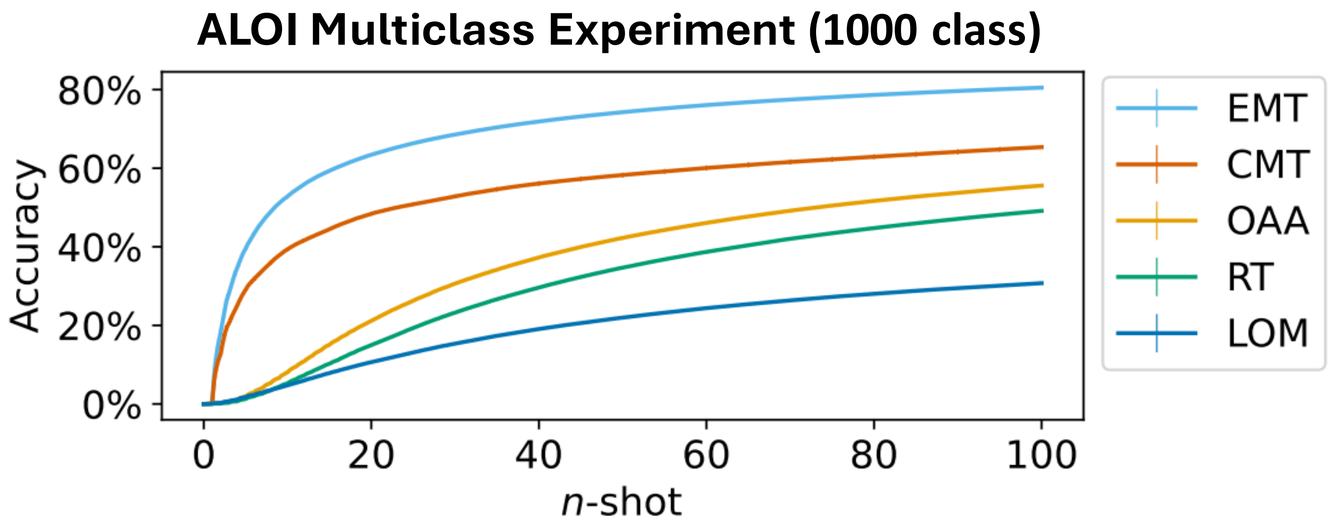
Example experiments

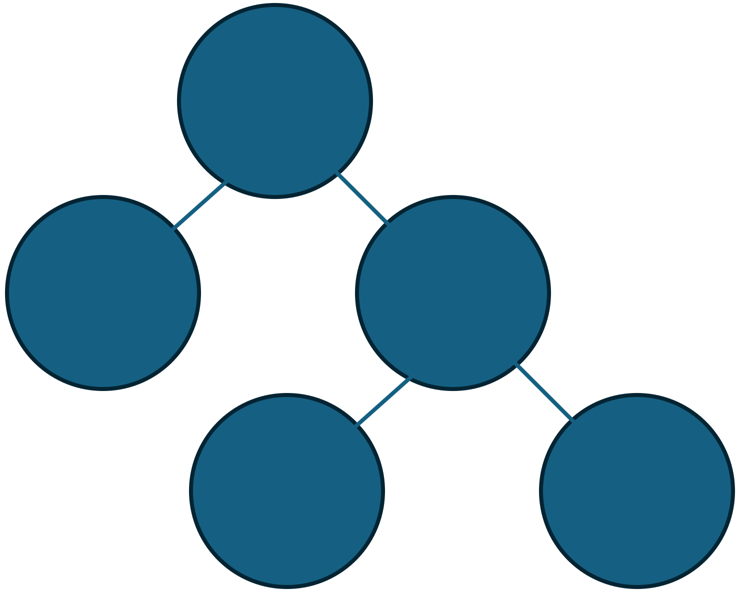
- Model selection, e.g. pick a query to send to GPT 3.5 or GPT 4 or even a smaller model
- VW learns to select the appropriate model and can save resources

https://github.com/VowpalWabbit/rl_chain

Improved Supervised Learning with Episodic Memory

- A new memory-based classifier we call EMT (Eigen Memory Tree)
  - Online insertions
  - Online learning
  - Logarithmic complexity

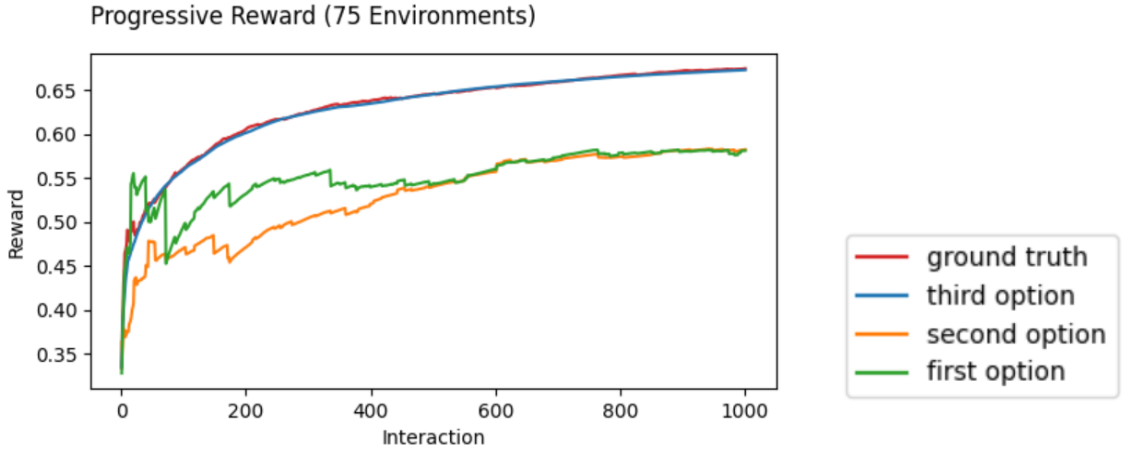


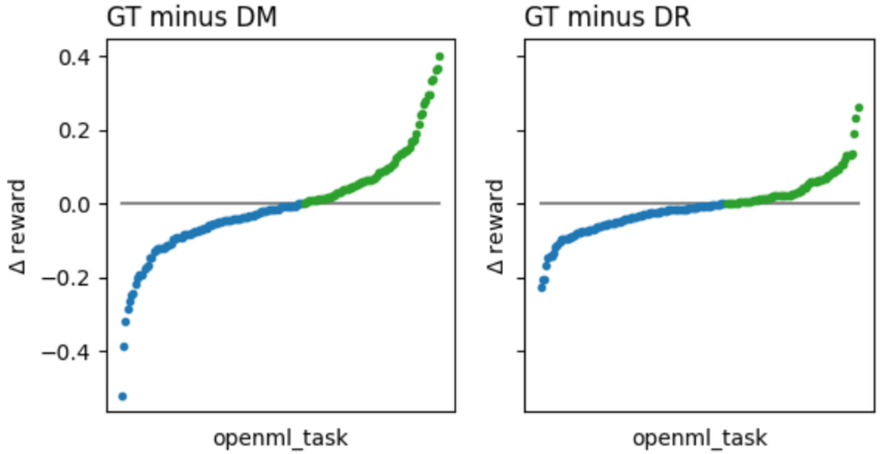


https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Eigen-Memory-Trees-(EMT)

Logged Bandit Experiments With Coba 7.0

- Coba is a Python package created to perform CB experiments
  - In version 7.0 Coba now offers support for logged bandit experiments
    - Easy creation of logged data for experiments
    - Off-Policy learning
    - Off-Policy evaluation (IPS,DR, or DM)
    - Unbiased Exploration Evaluation via Logged Data



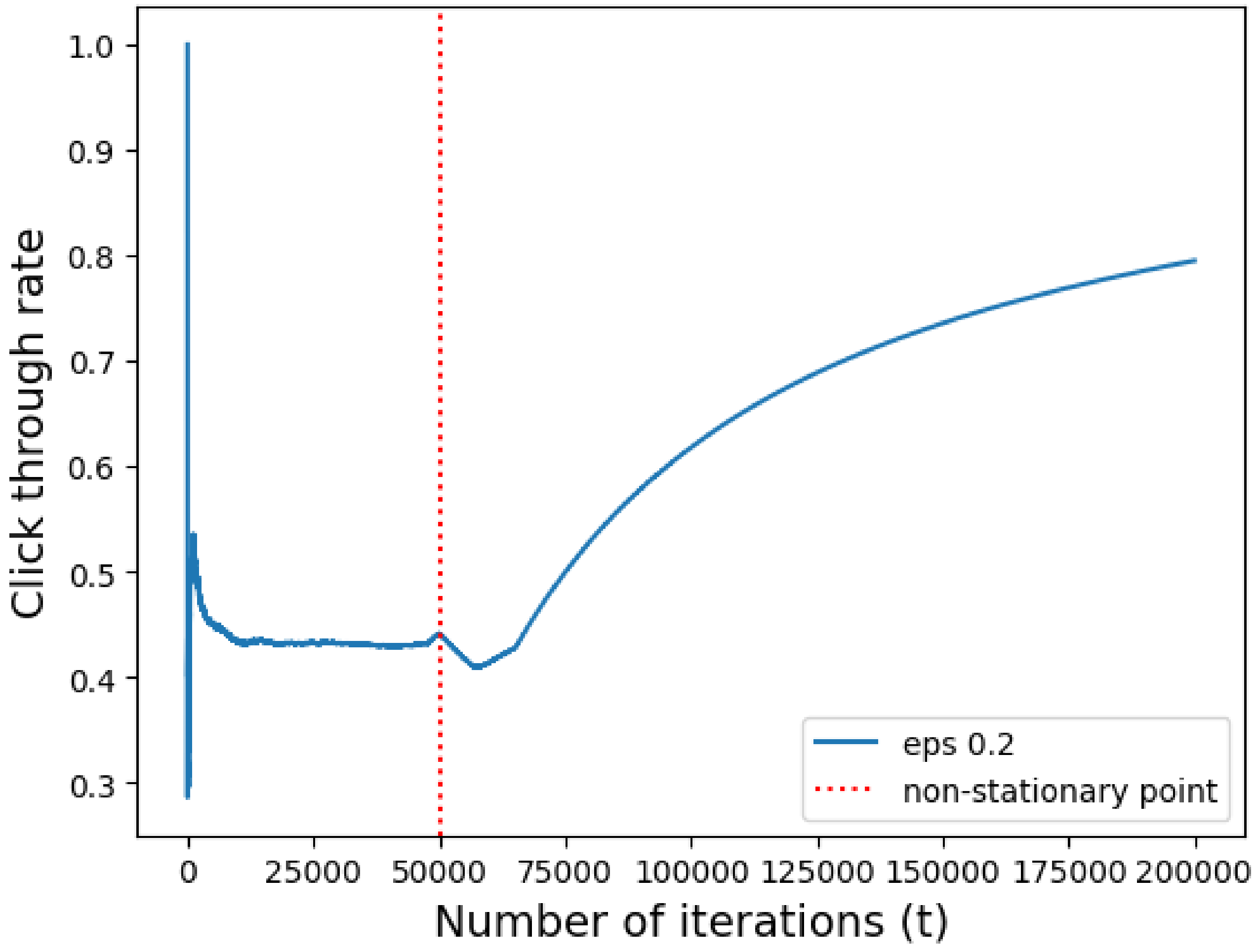


https://github.com/VowpalWabbit/coba/blob/master/examples/notebooks/Logged.ipynb

Non-Stationary Exploration (NSE)

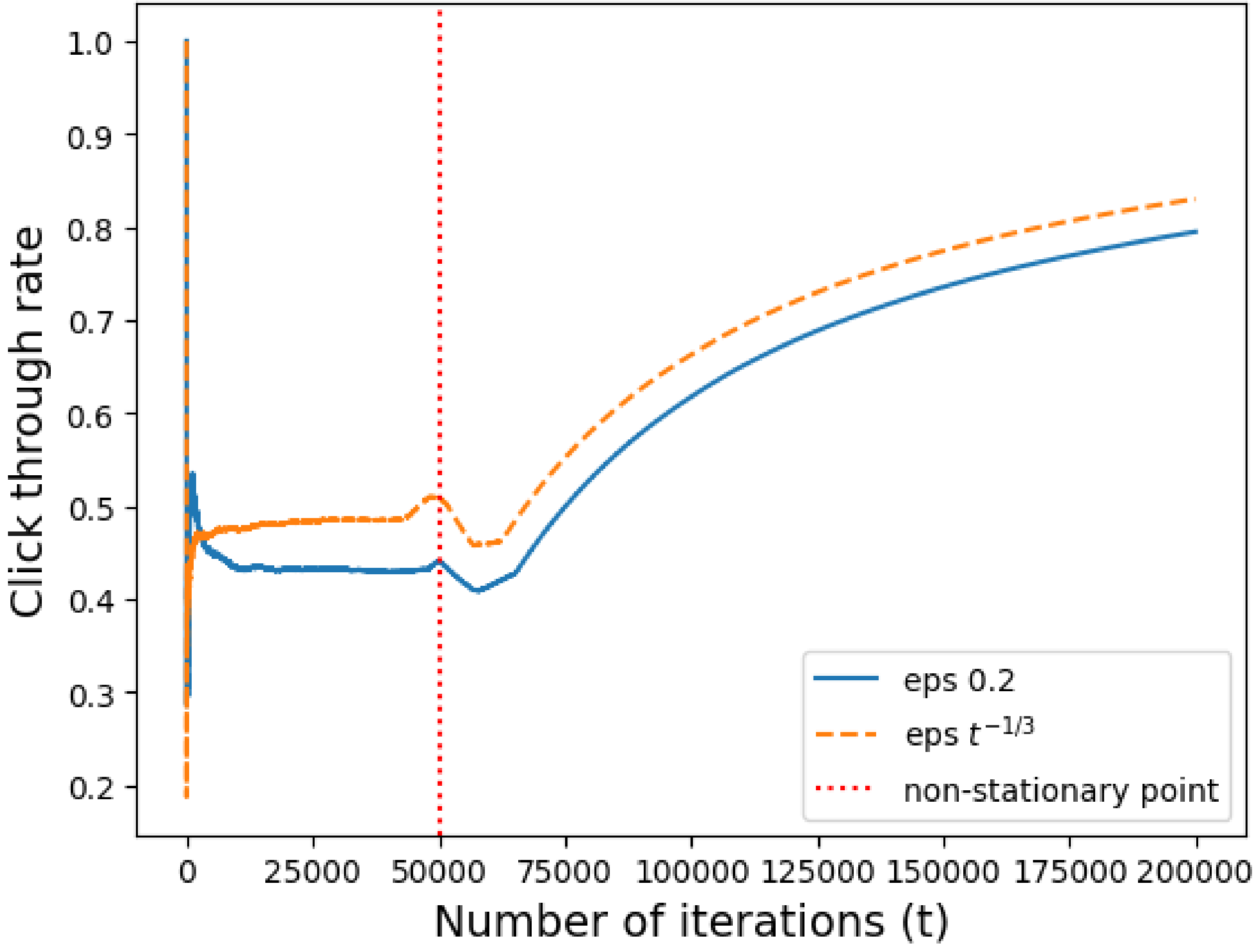
Recap: Tom and Anna

- Epsilon greedy works well
- Can we do better?



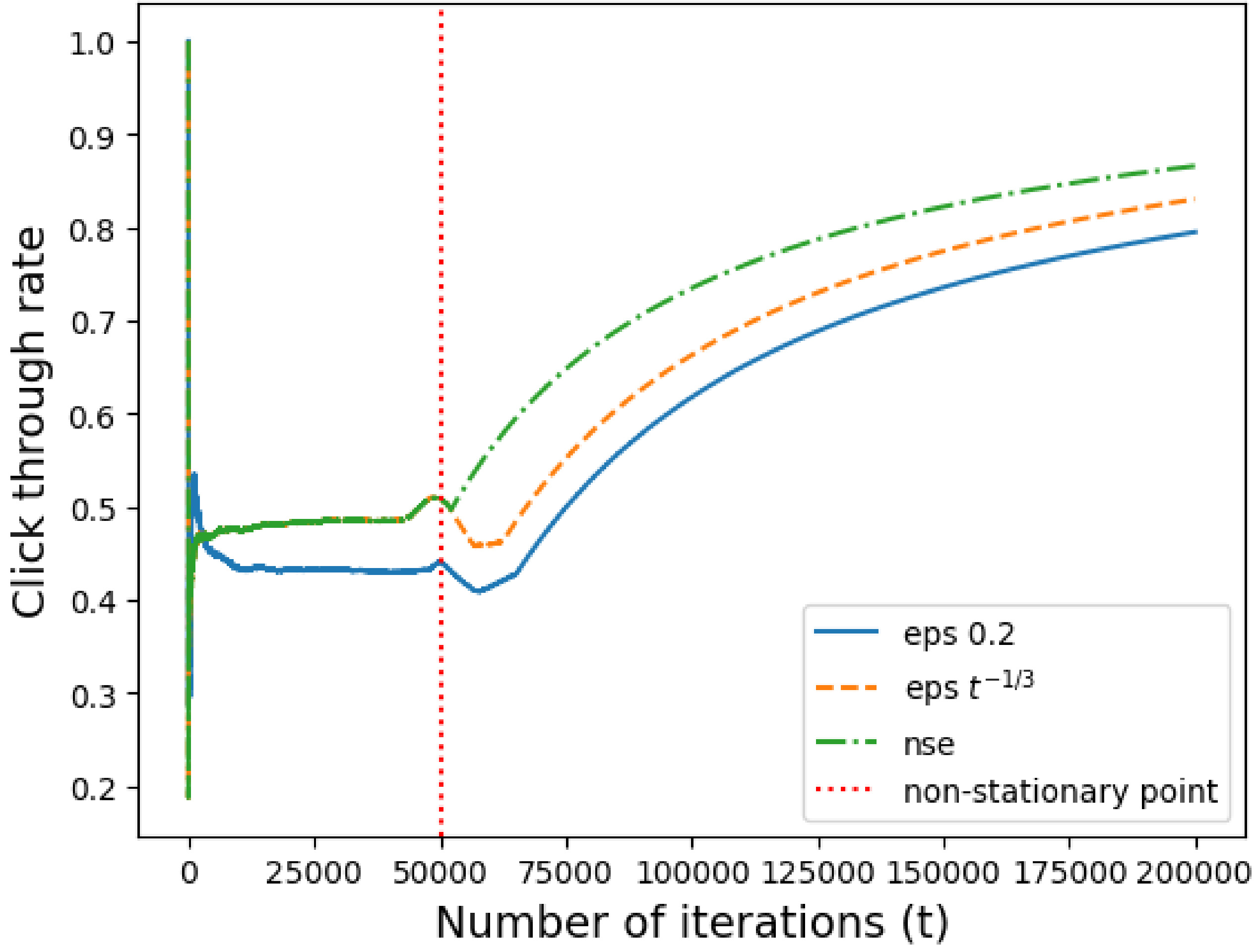
Non-Stationary Exploration

- Decaying exploration rate gets better
- Can we do better?

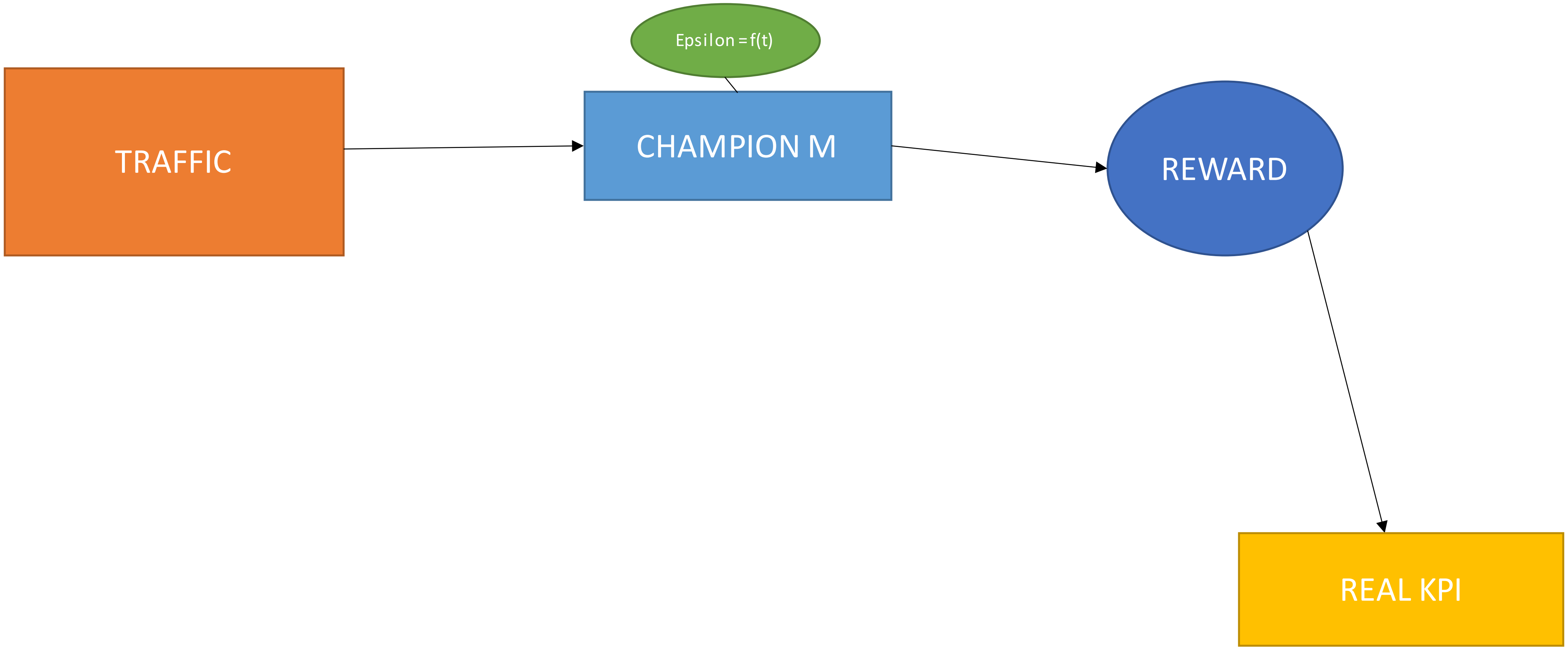


Non-Stationary Exploration

- NSE quickly detects the non-stationary point
- Great! How does NSE work?

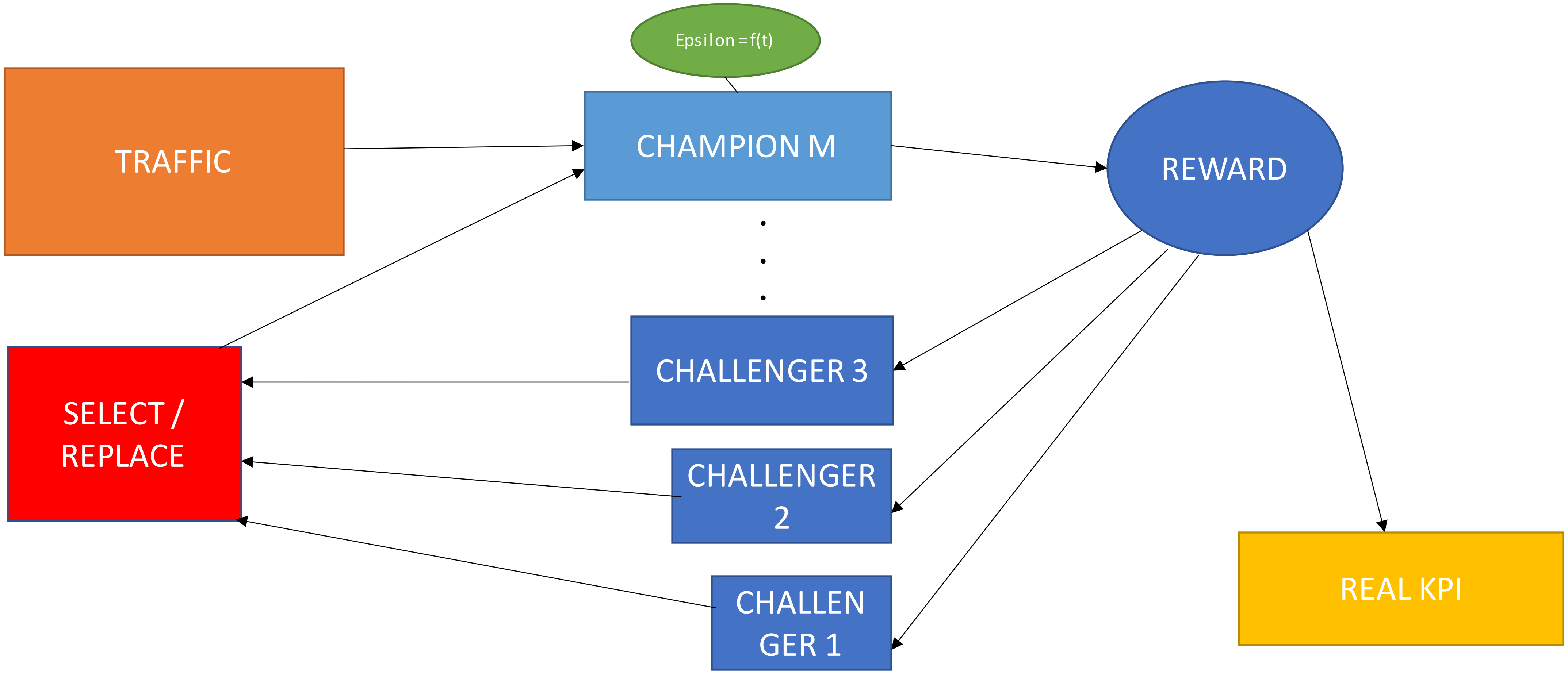


Non-Stationary Exploration



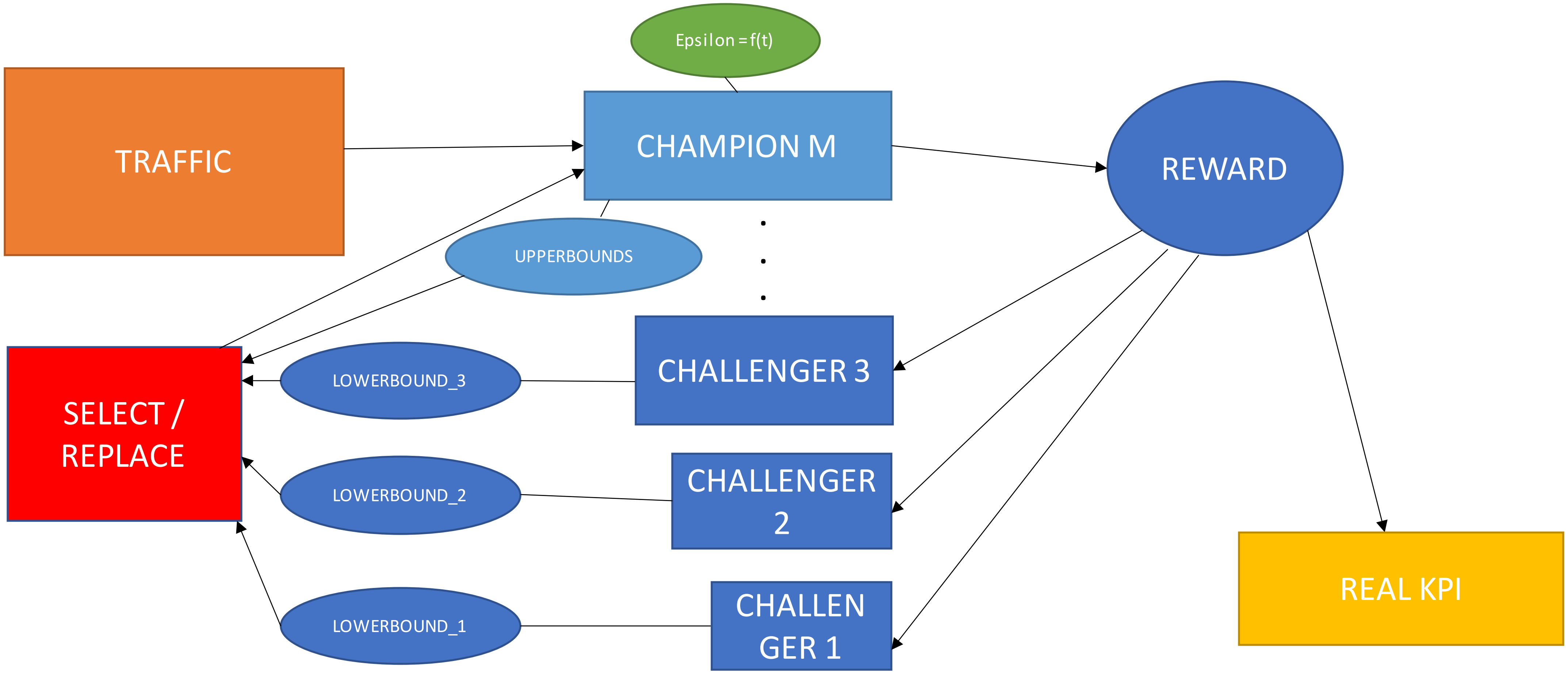
https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Epsilon-Decay-(Nonstationary-Exploration)

Non-Stationary Exploration



https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Epsilon-Decay-(Nonstationary-Exploration)

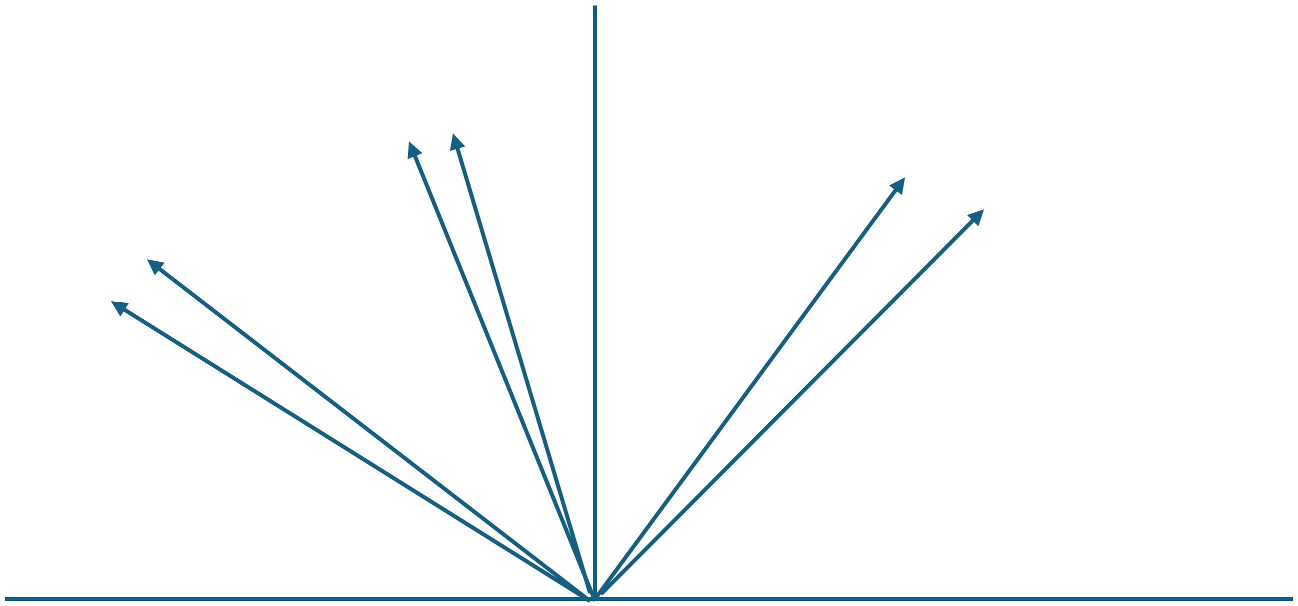
Non-Stationary Exploration



https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Epsilon-Decay-(Nonstationary-Exploration)

Contextual Bandits with Large Action Spaces

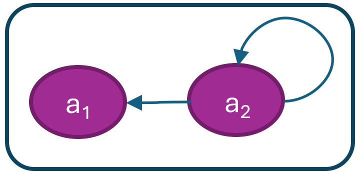
- > 50 actions
- Efficient exploration by filtering out actions that are similar
  - but use linearity of reward function to update all related actions



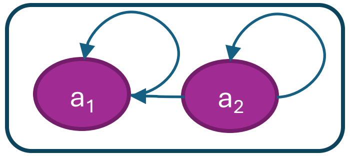
https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Contextual-Bandits-with-Large-Action-Spaces-(LAS)

Contextual Bandits with Graph Feedback

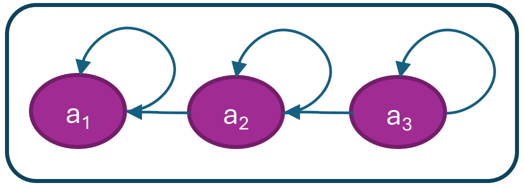
- Prior information about action relationships
- Constraint optimization problem
  - minimize regret while maximizing the information gathering (less exploration waste) by using the graph feedback leads
- Scenarios:
  - (Spam) filtering (aka apple tasting)



  - First-price auction bidding



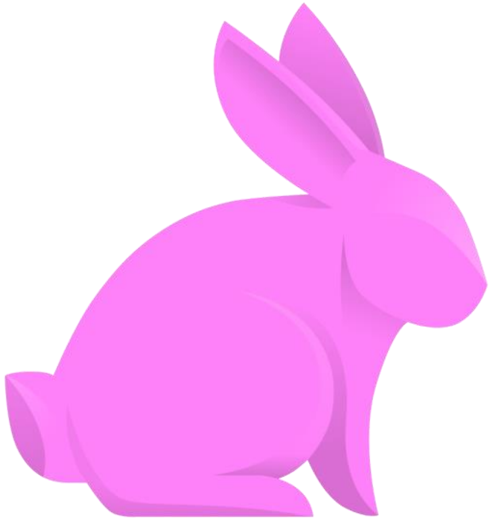
  - Inventory



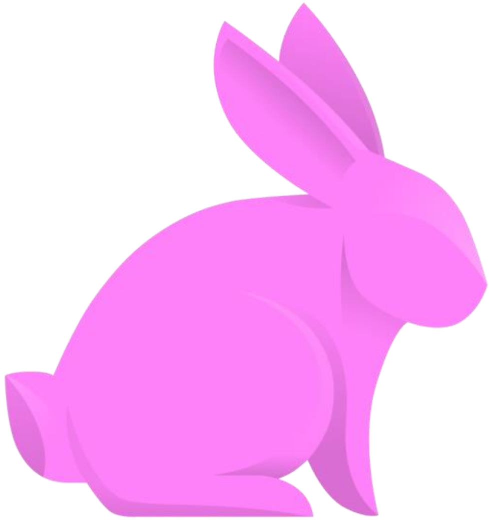
*$a_1 < a_2 < a_3$

https://github.com/VowpalWabbit/vowpal_wabbit/wiki/CB-with-Graph-Feedback

Model Merging

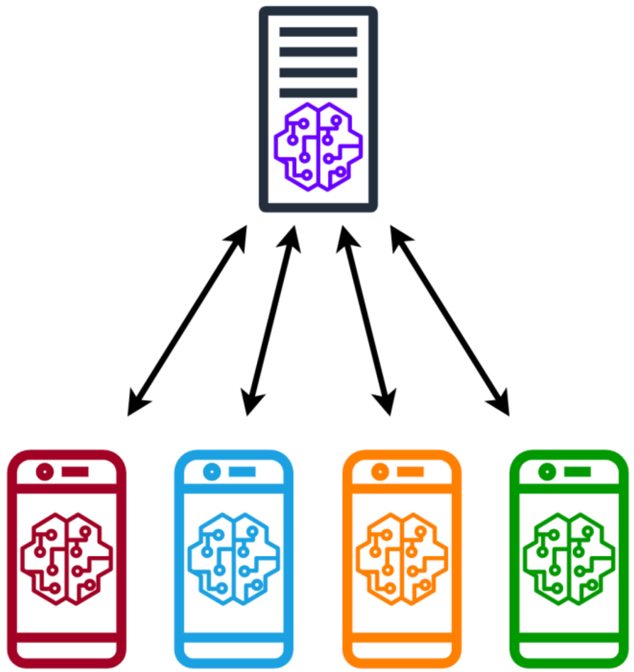


+



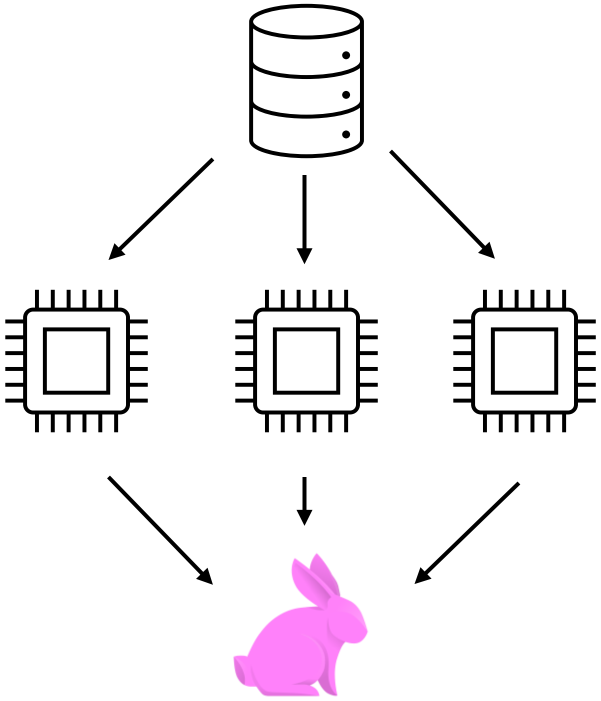
Why model merging?

Federated Learning



Wikipedia: Federated learning

Parallel Training



How model merging works

- Several VW models are trained with different data starting from a shared base model

$$VW_{base} + data_1 \Rightarrow VW_1 \qquad\qquad VW_{base} + data_2 \Rightarrow VW_2$$

- Two models can be "subtracted" to generate a *model delta* object

$$VW_1 - VW_{base} \Rightarrow delta_1 \qquad\qquad VW_2 - VW_{base} \Rightarrow delta_2$$

- Multiple model deltas can be merged together into a single delta

$$merge(delta_1\ ,\ delta_2) \Rightarrow delta_{avg}$$

- A base model is finally updated by "adding" a merged delta

$$VW_{base} + delta_{avg} \Rightarrow VW_{new}$$

Requirements

- Models share a common base
  - Or approximately a common base
  - Or are trained from scratch
- Model is loaded using the `--preserve_performance_counters` flag
- All learners used in the model's reduction stack support merging

How to merge models

- CLI

```
vw-merge --output out.vw --base base.vw model1.vw model2.vw
```

- C++ API

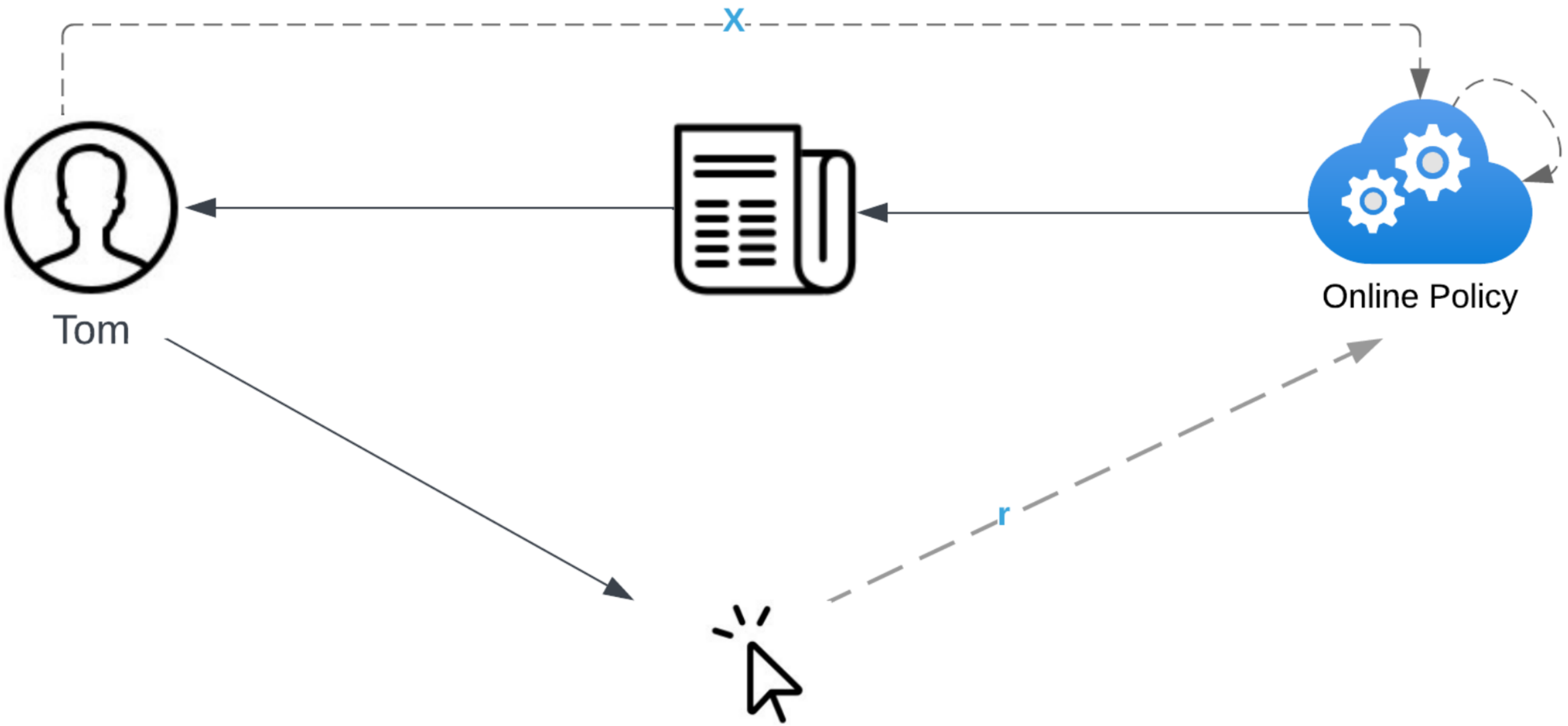
```
std::unique_ptr<VW::workspace> vw_base, vw1, vw2, vw_merged;
std::vector<const VW::workspace*> models_to_merge {vw1.get(), vw2.get()};
// this will internally subtract vw_base, merge deltas, and add back vw_base
vw_merged = VW::merge_models(vw_base.get(), models_to_merge);
```

- Python bindings

```
from vowpal_wabbit_next import calculate_delta, apply_delta, merge_deltas
delta_1 = calculate_delta(vw_base, vw_1)
delta_2 = calculate_delta(vw_base, vw_2)
delta_merged = merge_delta([delta_1, delta_2])
vw_new = apply_delta(vw_base, delta_merged)
```

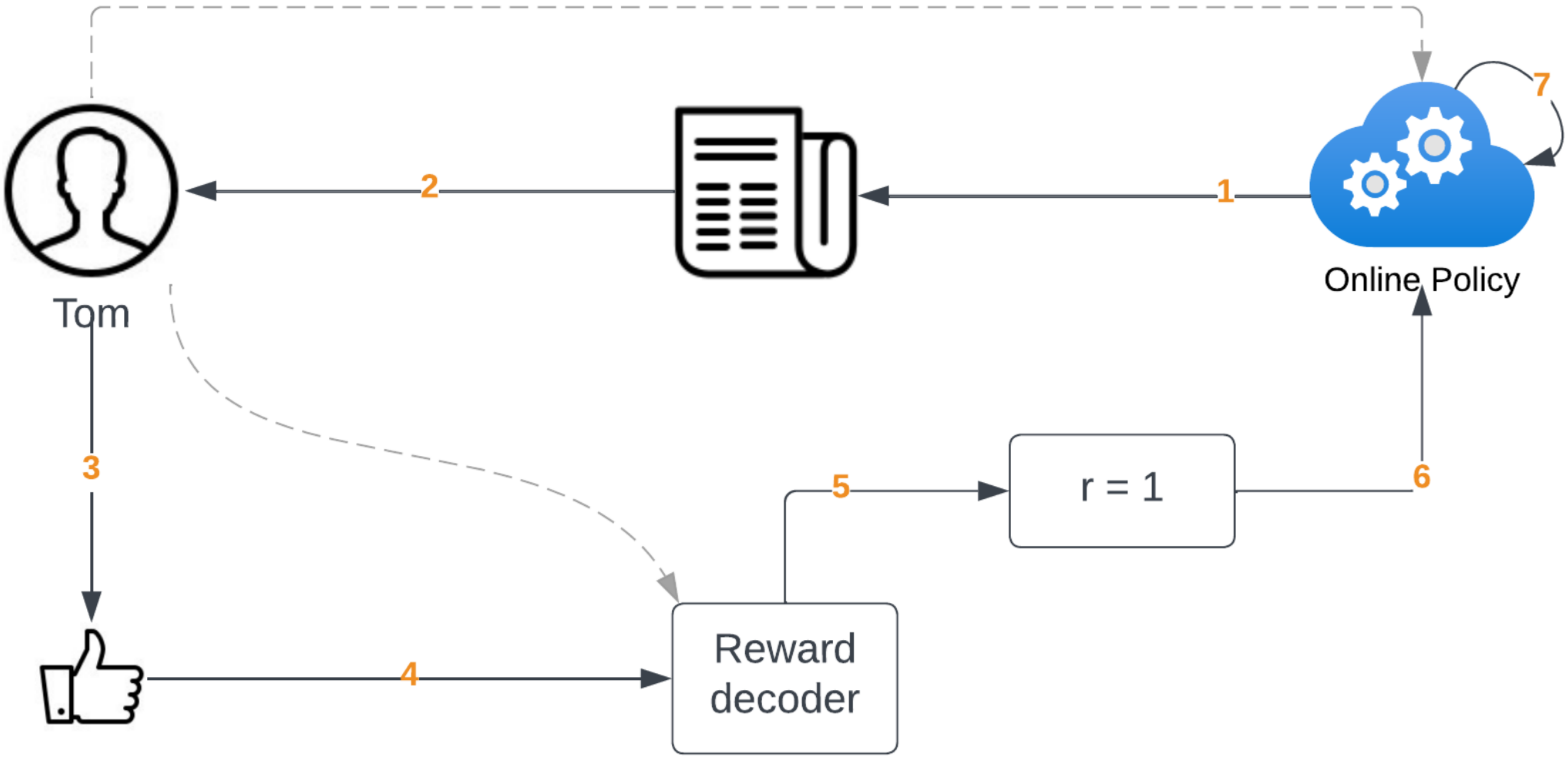
Interaction Grounded Learning

Motivation

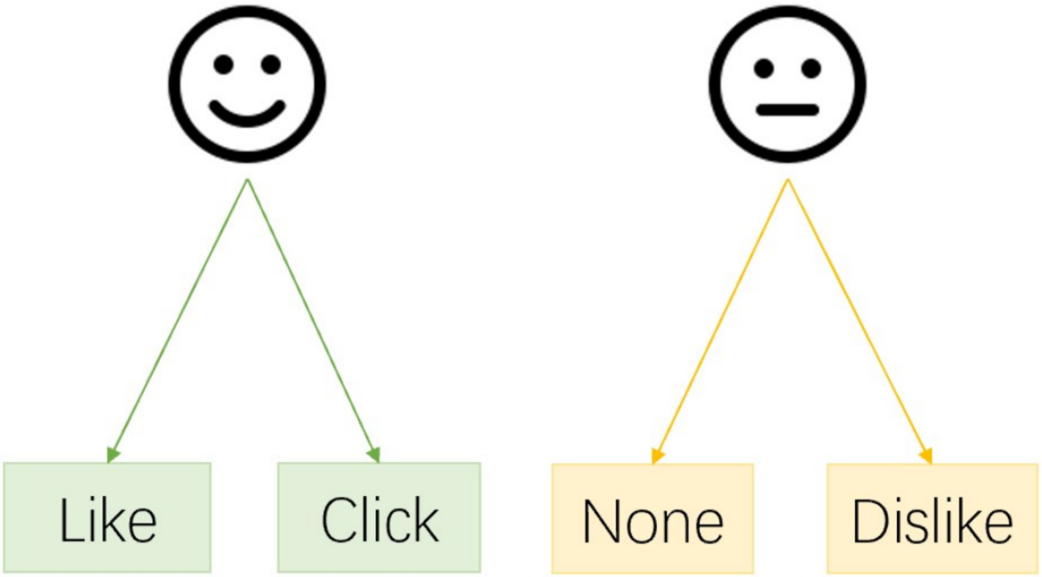


Can We Discover Reward?

IGL

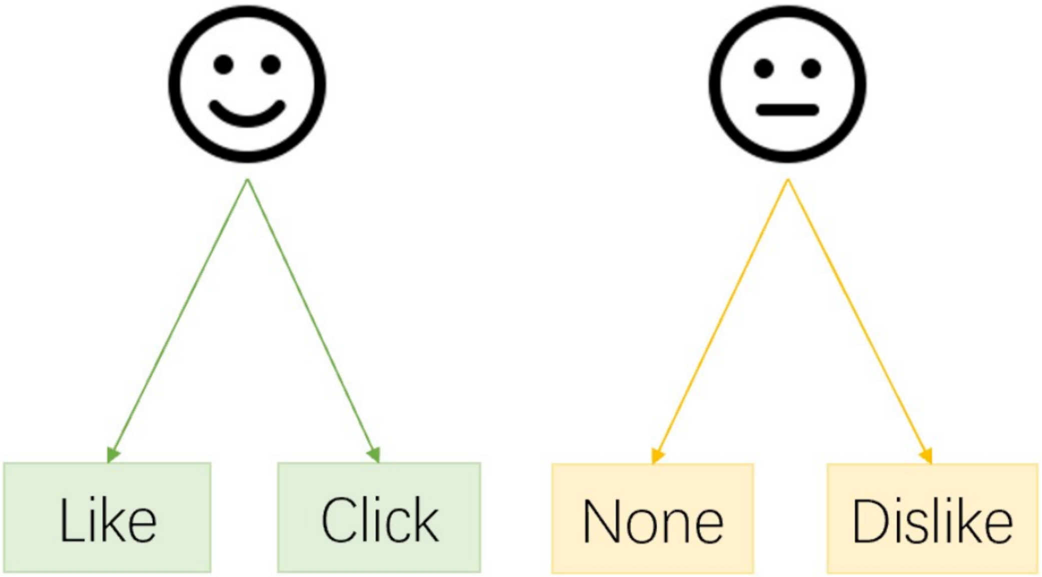


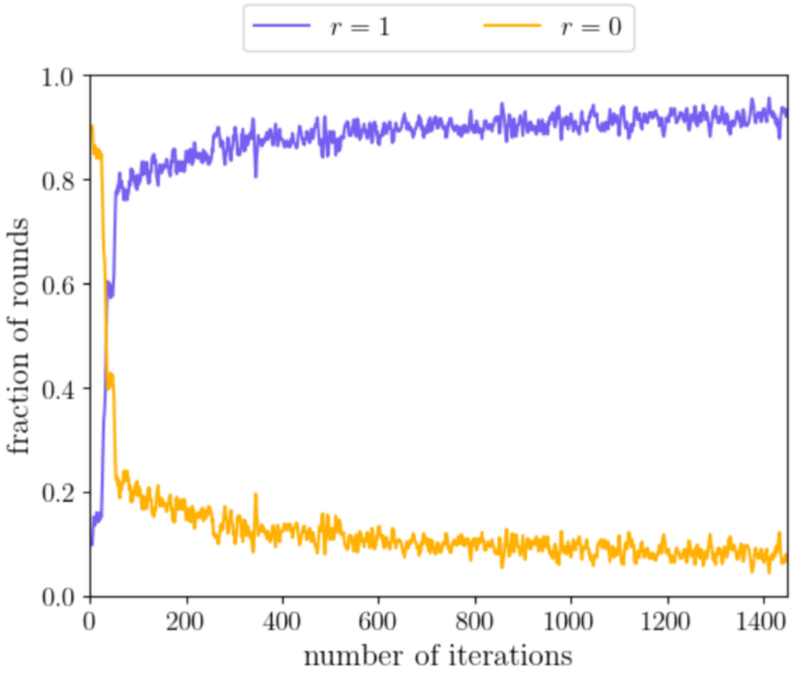
Latent Reward Structure



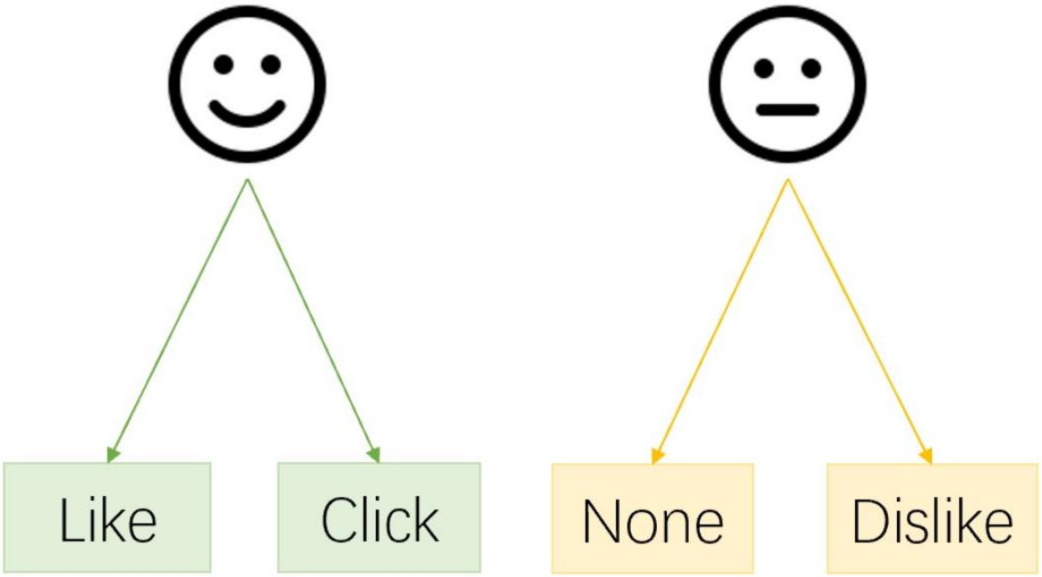
anytime the posterior probability of an action is predicted to be more than twice the prior probability, we deduce r != 0.

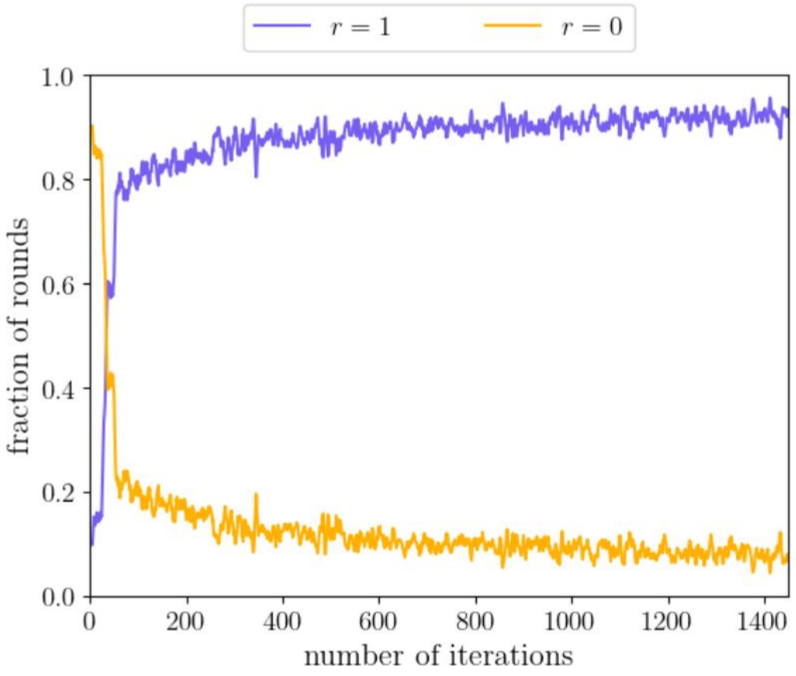
Latent Reward Structure

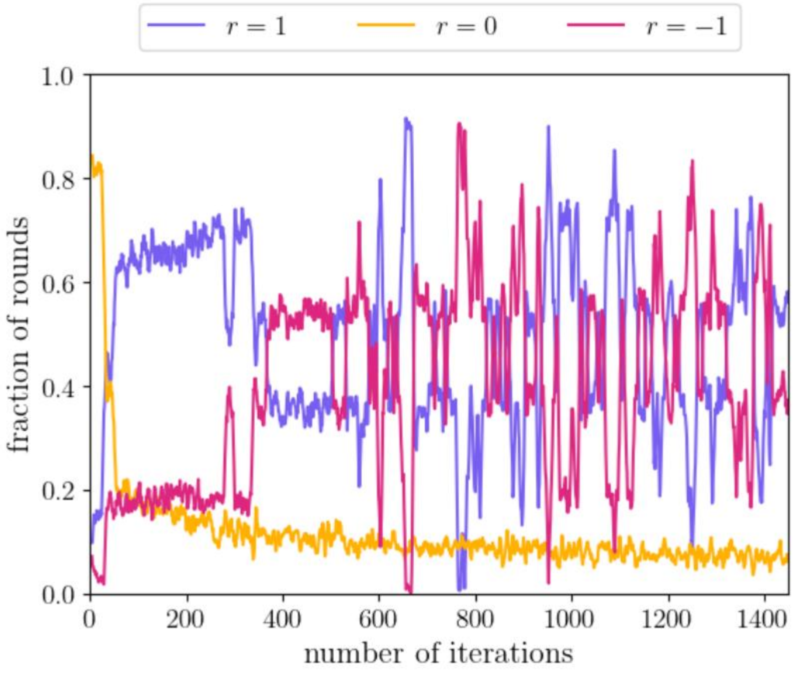




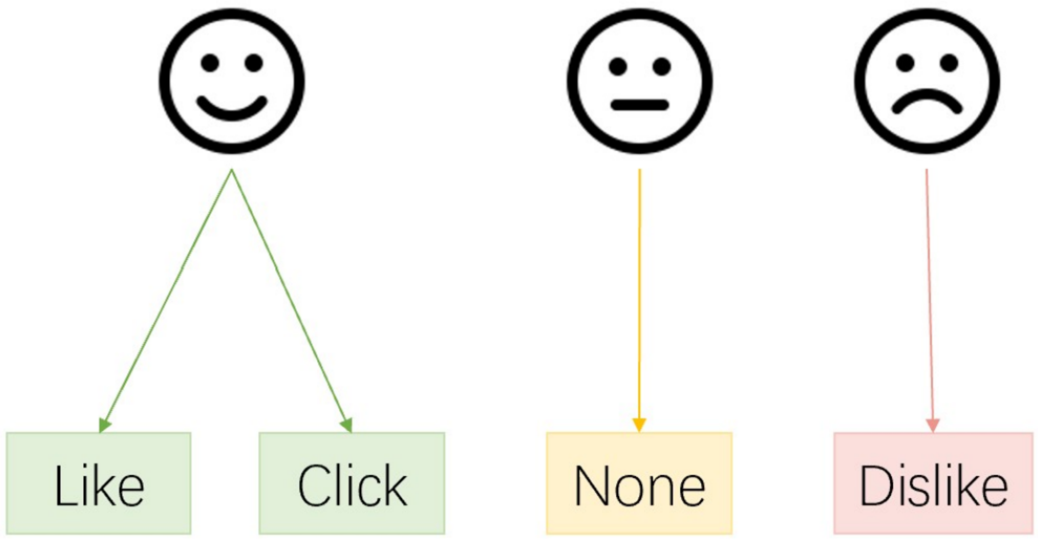
Latent Reward Structure

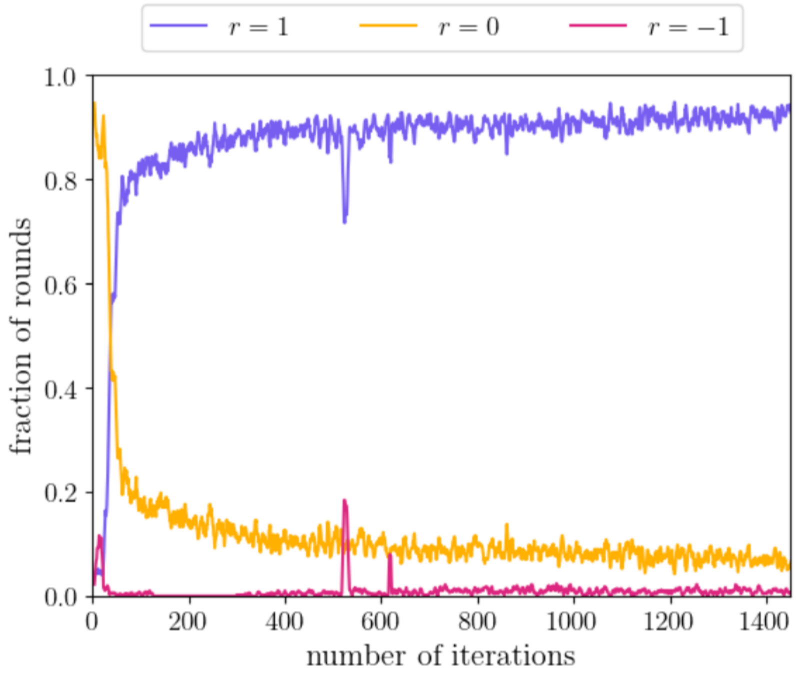






Latent Reward Structure





use a negative oracle to detect "definitely negative" events, then use extreme event detection to identify r = 1

How To Use It With VW

- `--cb_explore_adf --experimental_igl` activates the algorithm
- sending feedback instead of reward in json format

  ```
  e.g.
  {
    "v": "dislike",
    "_definitely_bad": true
  }
  ```

- optional: label the definitely negative feedback

Learn More

- ICML workshop:
  - Interactive Learning with Implicit Human Feedback, Sat 29 Jul, 9 a.m. HST, Meeting Room 315
- Paper:
  - https://arxiv.org/abs/2211.15823
- Wiki:
  - https://github.com/VowpalWabbit/vowpal_wabbit/wiki/Interaction-Grounded-Learning

Q & A