ODSC APAC
VIRTUAL CONFERENCE

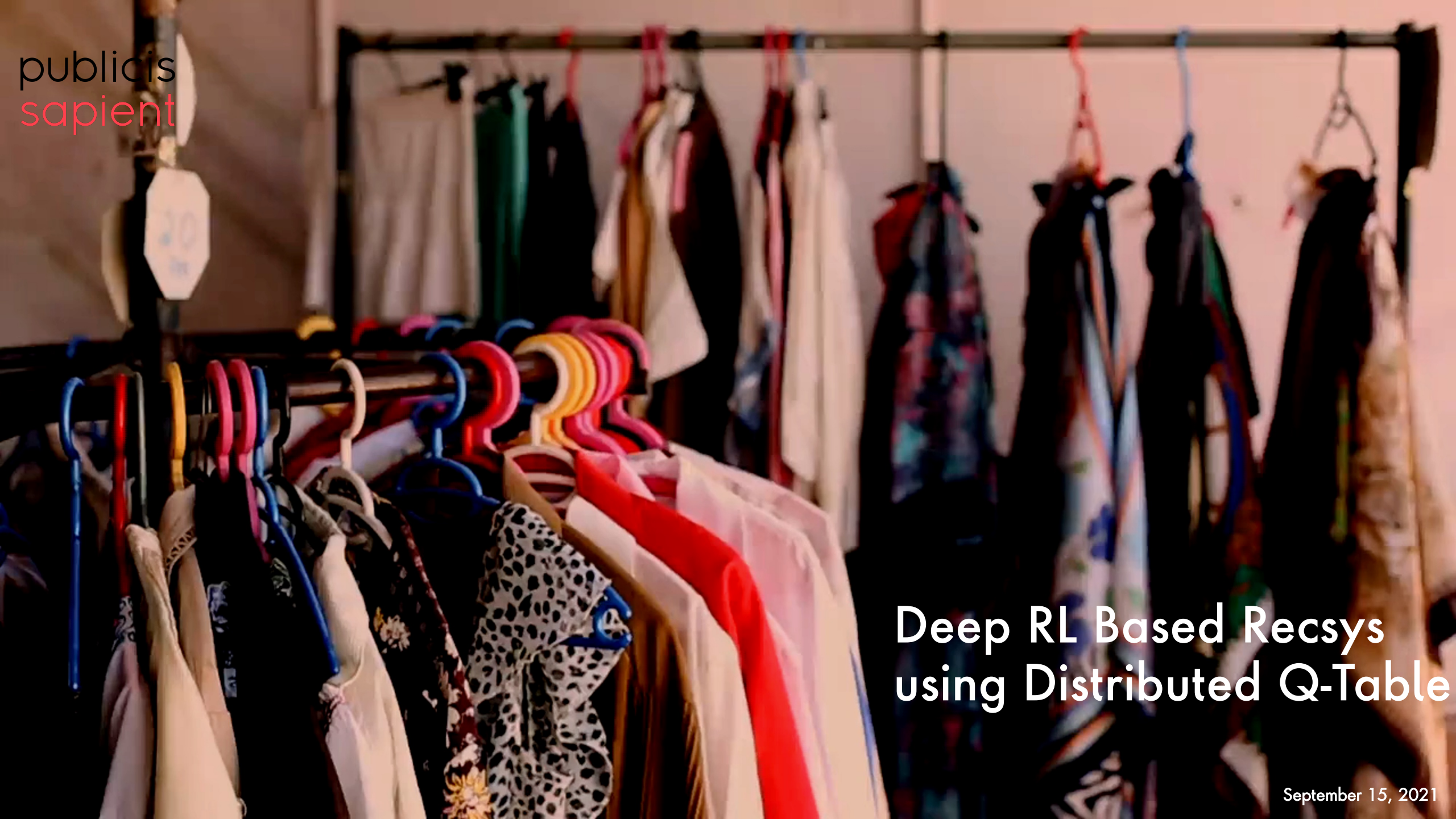
September 15th–16th

# Deep Reinforcement Learning Based RecSys Using Distributed Q-table

**TUTORIAL**

- DEEP LEARNING

**Ravi Ranjan**
Senior Data Scientist
**Publicis Sapient**



publicis
sapient

# Deep RL Based Recsys using Distributed Q-Table

September 15, 2021

# About the Speakers

Ravi Ranjan is working as Lead Data Scientist at Publicis Sapient (India) with expertise in budling scalable ML solutions. He has done proficiency course in Reinforcement Learning from IISc, Bangalore. He is a certified Google Cloud Architect. He is contributor and member at Kubeflow.

publicis
sapient

# Session Logistics

1. Access to the session environment using the following link. [https://bit.ly/join-the-ODSC-session]
2. Presentation and research paper will be available at link. [https://bit.ly/ODSC-conference-India-2021]
3. Connecting to the speaker [Please send introductory note in LinkedIn invite]

    https://bit.ly/ravi-ranjan-03
4. Don't forget to tweet and share the session with #ODSCAPAC

publicis
sapient

# Learning Outcome

1. Gain an understanding of Recommendation Systems.
2. How to transform the concepts and build Recommendation Systems?
3. Gain a detailed understanding of deep reinforcement learning-based recommendation system.
4. How to recommend using the deep RL based model with distributed Q-table?
5. Reference architecture of recommendation engines.

# Session Agenda

1. Introduction: Recommendation Systems
2. Classical approaches for building a recommendation system.
3. Limitations of classical approaches
4. Introduction: Reinforcement learning
5. Deep reinforcement learning-based algorithm for building a recommendation system.
6. Training methodology and result discussion.
7. Question-Answers

Section 1

# Recommendation Systems

# Recommendation System is everywhere!

## Amazon

Product, Video and music recommendations

## Netflix

Video recommendation and artwork personalisation

## Facebook

Content, ad, account and entity recommendations

## Spotify

Music and podcast recommendations

## Youtube

Video content and ad recommendations

## Myntra

Product and personalized size recommendations

## Coursera

Courses and learning path recommendations

## LinkedIn

Connections, job and content recommendations

# What is Recommendation System?

"Serve the relevant items to users in an automated fashion to optimize short- and long-term business objectives."

## RELEVANT (WHAT)

1. Novelty
2. Serendipity
3. Diversity

## AUTOMATED (HOW)

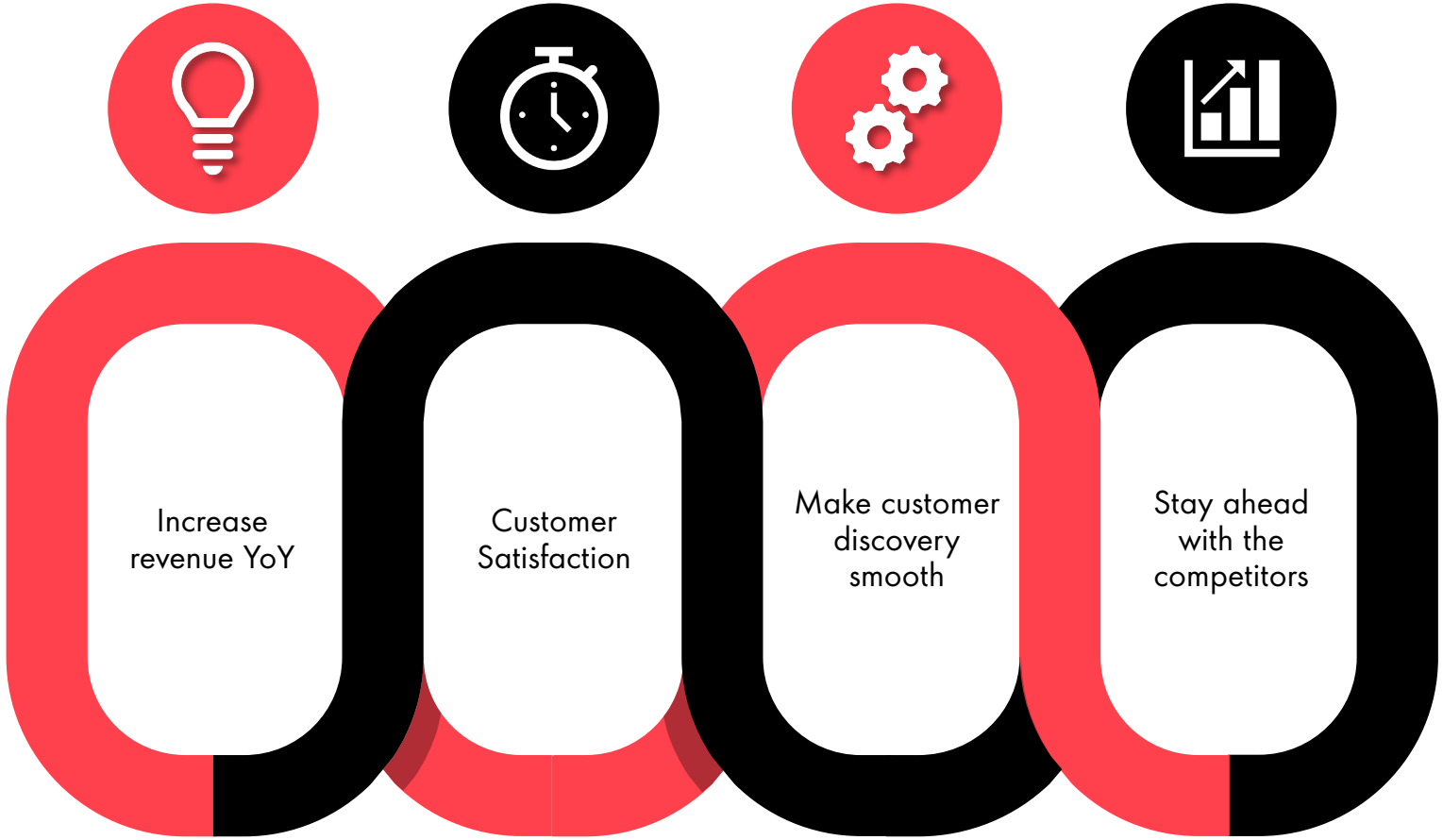
1. No manual intervention
2. Scale up

## BUSINESS OBJECTIVES ( WHY )

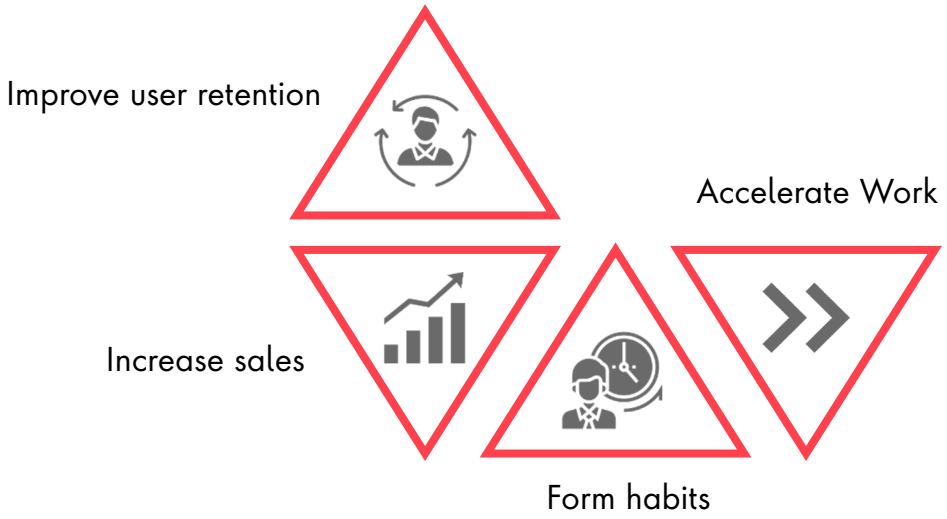
1. Short Term Business Objectives
    a. High clicks
    b. Revenue
    c. Positive explicit ratings

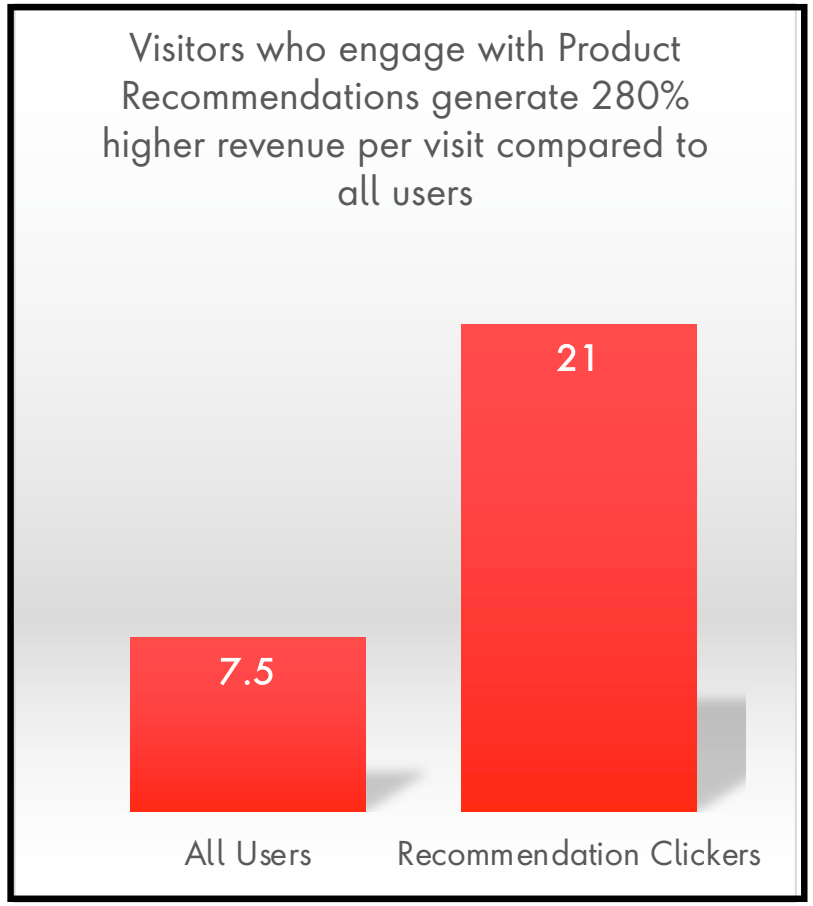
2. Long Term Business Objectives
    a. Increased engagement
    b. Increase in social action
    c. Increase in Subscriptions

# Why Recommendation Systems important in 2021?



# Facts On Companies Using Recommendation Systems




## Enterprises using Recommendation Systems

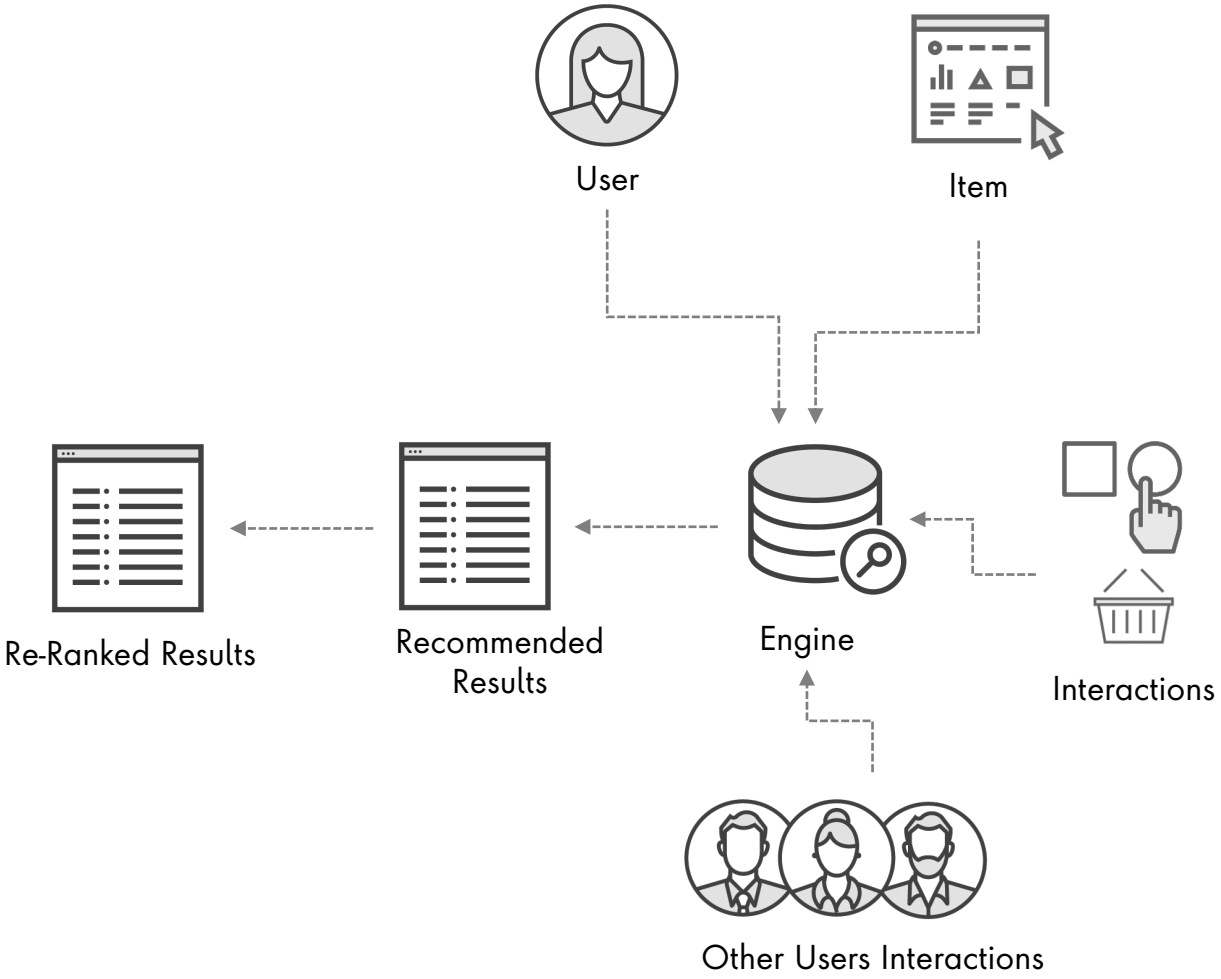
75% of what consumers watch on Netflix comes from recommendation system

Amazon credits recommender systems with 35% of its revenue

Best Buy reported a 23% increase, thanks in part to their recommender system

# Problem Space : Recommendation

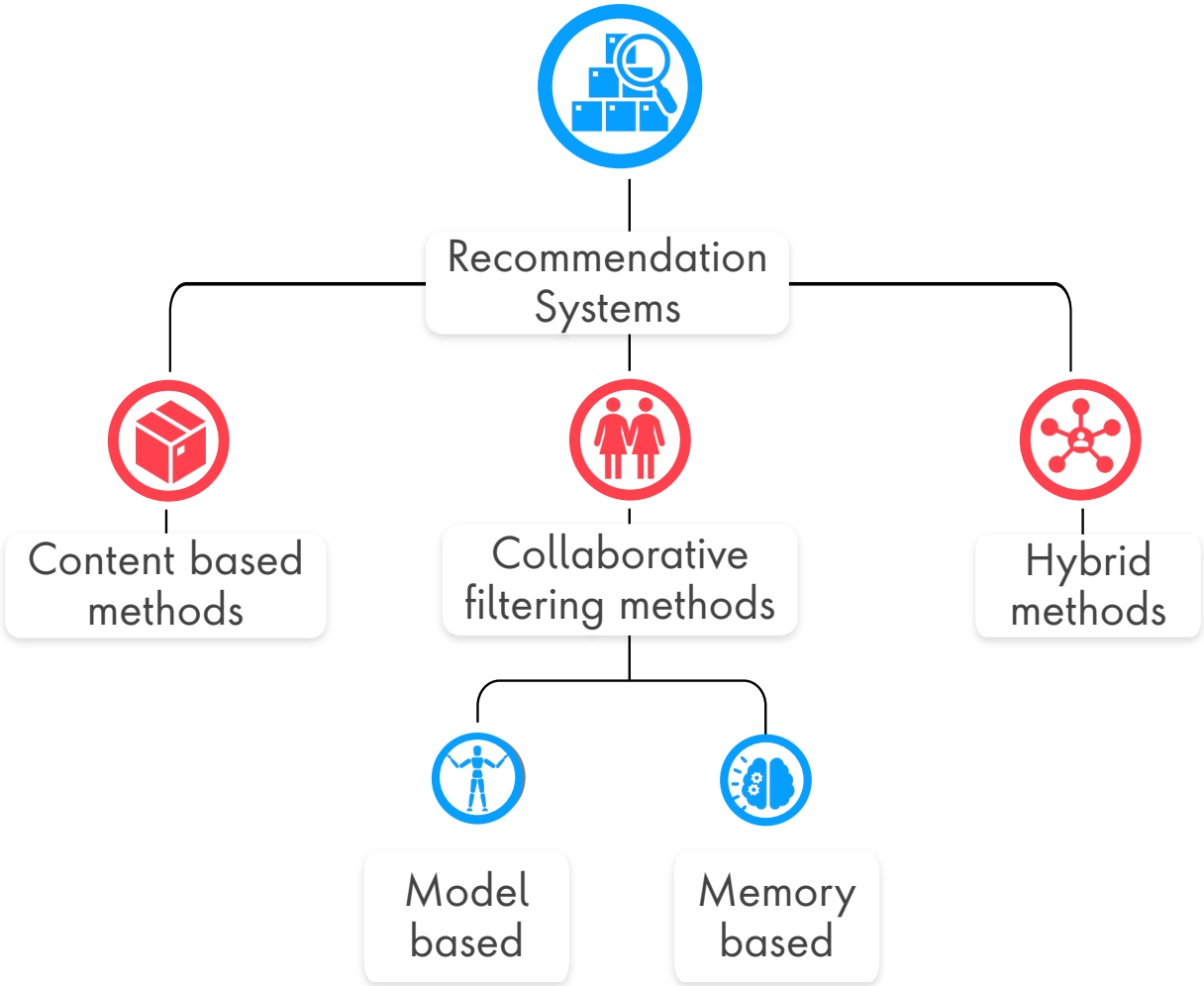
## Recommendation Engines



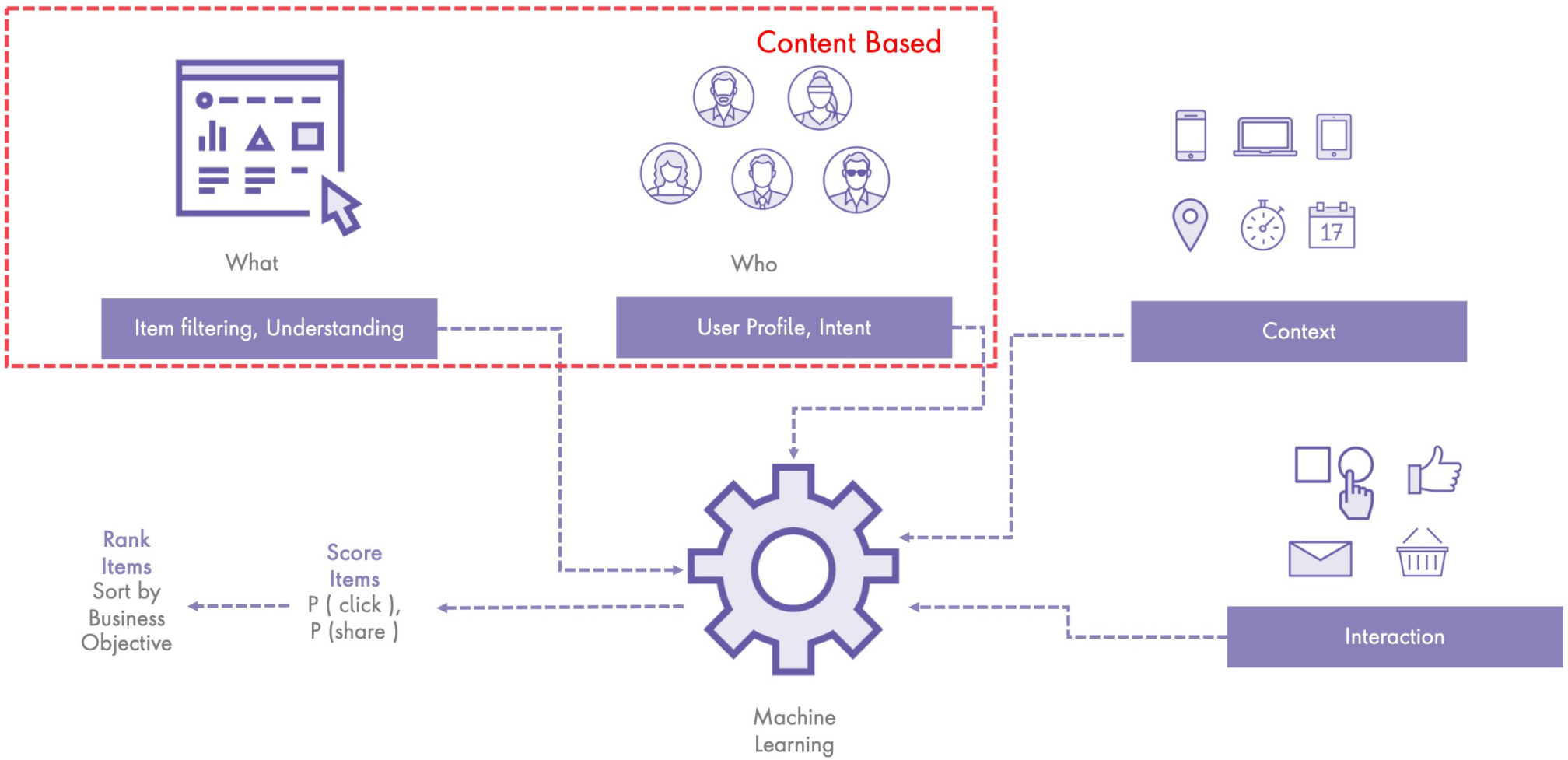

## Challenges

- How to represent users and items?
- How to use dynamic user behaviors?
- How to use implicit ( view, share )or explicit (rating or review) feedback ?

# Existing Research & Development



# Internals


Content Based
What
Who
Item filtering, Understanding
User Profile, Intent
Context
Interaction
Rank
Items
Sort by
Business
Objective
Score
Items
P ( click ),
P (share )
Machine
Learning

# Content Based Recommendation

## Pros

No need of other users' data

Easy to understand reason behind recommendation

Capable of recommending new and unknown items

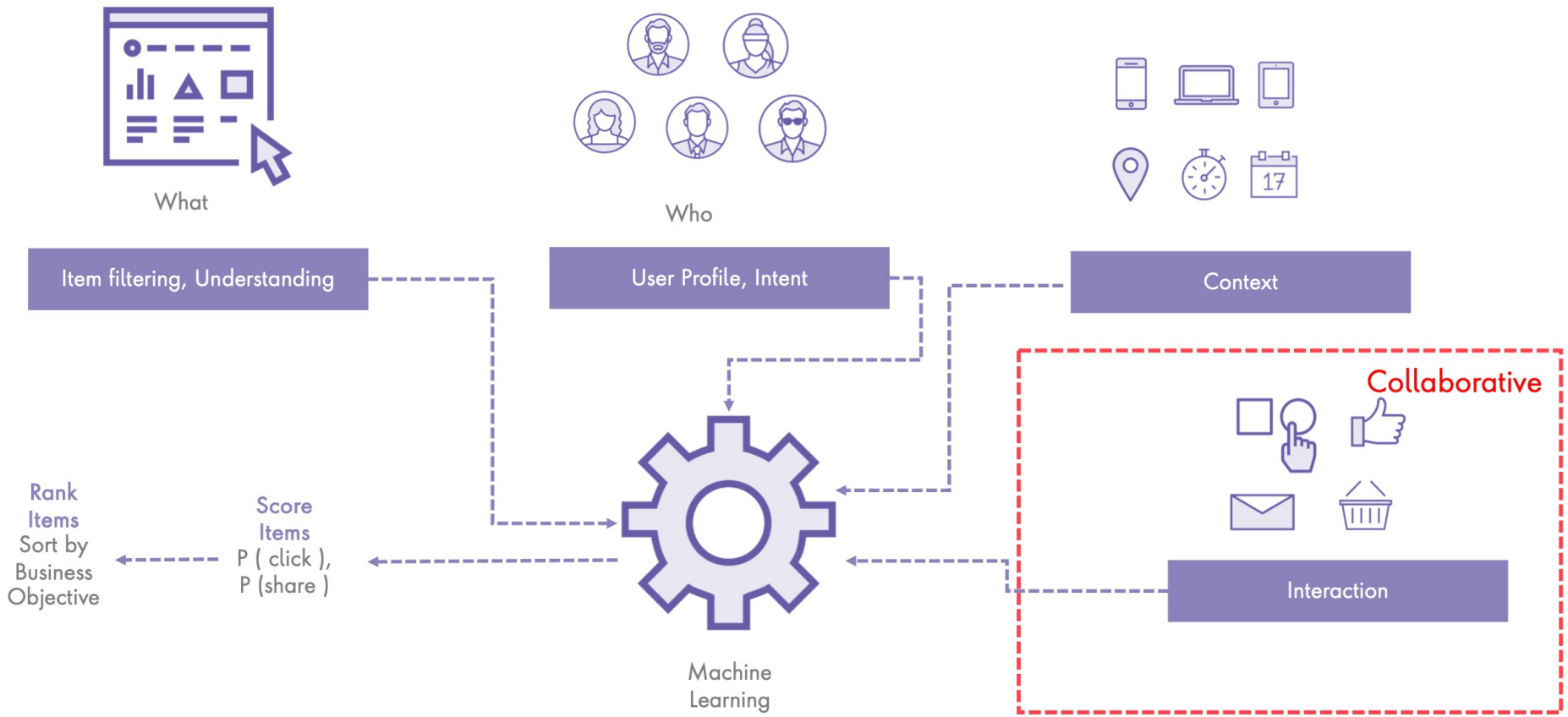
## Cons

Can only be effective in limited circumstances

No suitable suggestions if content doesn’t have enough information

Depend entirely on previous selected items and therefore cannot make predictions about future interests of users

# Internals


What
Who
17
Item filtering, Understanding
User Profile, Intent
Context
Collaborative
Rank
Items
Sort by
Business
Objective
Score
Items
P ( click ),
P (share )
Interaction
Machine
Learning

# Collaborative Filtering

## Pros

Content information not required either of users or items

Personalized recommendations using another user's experience

No domain experience required

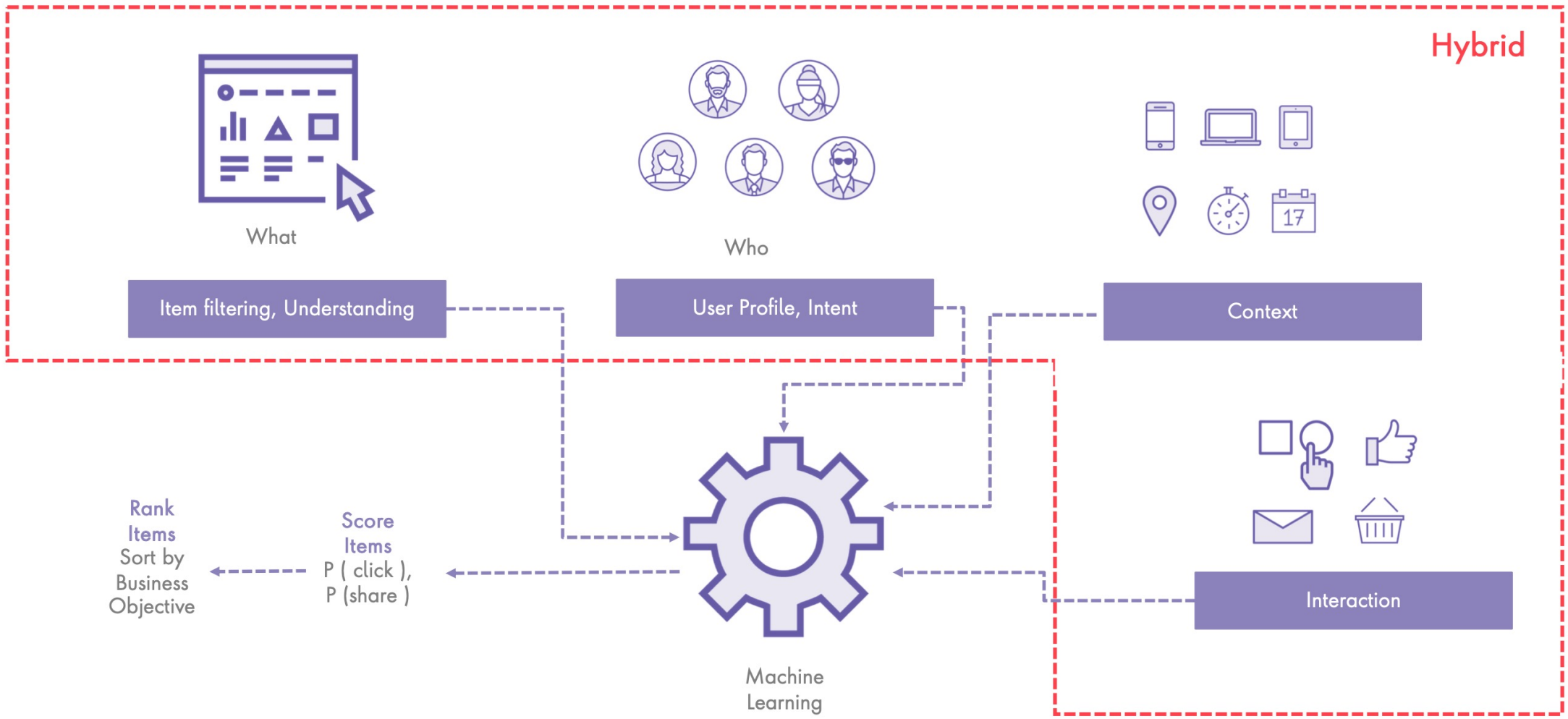
## Cons

Cannot produce recommendations if there is no interaction data available
( Cold Start Problem )

Often demonstrate poor accuracy when there is little data about users' ratings
(Sparsity)

Popular items get more feedback
( Popularity bias )

# Internals


Hybrid
What
Who
17
Item filtering, Understanding
User Profile, Intent
Context
Rank
Items
Sort by
Business
Objective
Score
Items
P ( click ),
P (share )
Machine
Learning
Interaction

# Hybrid Recommendation Engine

## Pros

Solve the issue of Cold Start by leverage both content and collaboration

Use of Implicit feedback reduces the sparsity issues to a large extent

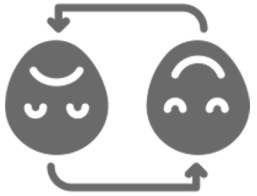
Can include higher order feature interactions as well

## Cons

Difficult to implement

# Limitations of the Classical Approaches

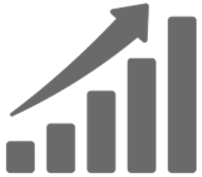
Changing Data



Dynamic User Behavior

Static Recommendations

Grey Sheep



Scalability

Cold Start

# Why need Reinforcement Learning to develop Recommendation Systems?

Works on Changing Data

Tackles Changing User Taste

Tackles Static Recommendations

Grey Sheep

Utilizes User Feedback to Give Recommendations

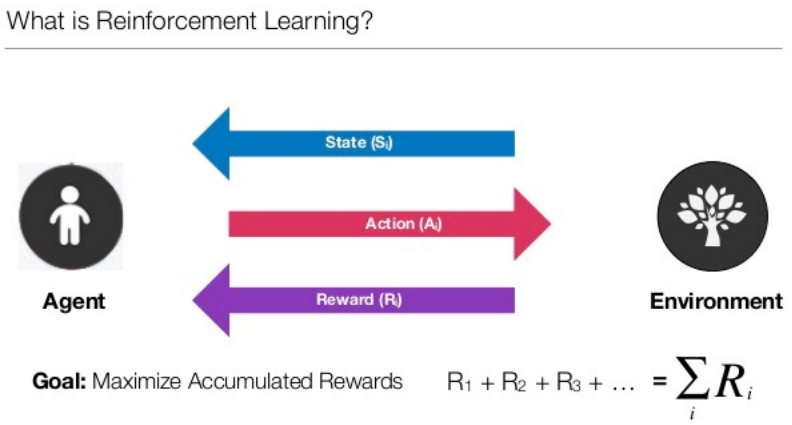
User Retention

Section 2

# Reinforcement Learning

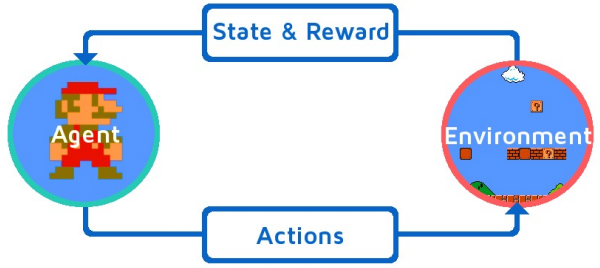
# What is Reinforcement Learning?

Reinforcement learning (RL) is the new approach to teach machines to interact with the environment and receive rewards for performing the right actions until they successfully meet their goal. The attractiveness of reinforcement learning is that it teaches systems to focus on the long-term reward.



# Reinforcement Learning – An elaborative way

- Let's suppose that the reinforcement learning agent is learning to play Mario as a example. The reinforcement learning process can be modelled as an iterative loop that works as below:




- The RL Agent receives state $S^0$ from the environment i.e. Mario
- Based on that state $S^0$, the RL agent takes an action $A^0$, say—the RL agent moves right. Initially, this is random.
- Now, the environment is in a new state $S^1$ (new frame from Mario or the game engine)
- Environment gives some reward $R^1$ to the RL agent. It probably gives a +1 because the agent is not dead yet.

- This RL loop continues until the agent is dead or it reaches the destination, and it continuously outputs a sequence of state, action and reward.

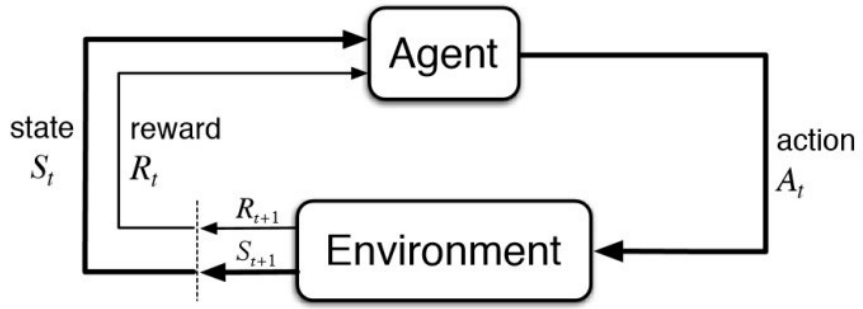
- The basic aim of our RL agent is to **maximize the reward**.

# Reinforcement Learning – Building blocks

- A goal based learning, based on interaction with environment.

- Some key terms that describes the elements of a Reinforcement Learning problem are:
  1. **Environment**: Physical world in which the agent operates
  2. **State**: A state is a concrete and immediate situation in which the agent finds itself. In recommendation system, state is the previous history of the user
  3. **Action**: In recommendation system, action is the items we are recommending to a user to maximise the rewards
  4. **Reward/Penalty**: The feedback which the agent gets after taking an action in a certain state
  5. **Policy**: The policy is the strategy that the agent employs to determine the next action based on the current state. It maps states to actions, the actions that promise the highest reward.
  6. **Agent**: Agent is the recommendation system itself

- In order to build an optimal policy, the agent faces the dilemma of exploring new states while maximizing its reward at the same time. This is called Exploration vs Exploitation trade-off.



# Types of Reinforcement Learning

Model Based

1. Learn the model of the world, then plan using the model
2. Update the model often
3. Re-plan often
4. Example: AlphaZero

Value based

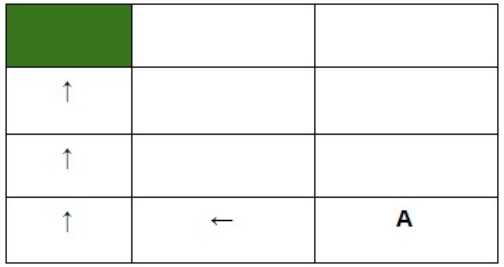
1. Learn the state and state-action value
2. Act by choosing the best action in the state
3. Exploration is the necessary add-on
4. Example: Q-Learning

Policy based

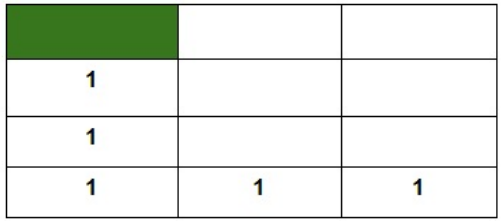
1. Learn the stochastic policy function that maps state to action
2. Act by sampling the policy
3. Exploration is baked in
4. Example: Actor-Critic

# Reinforcement Learning Example

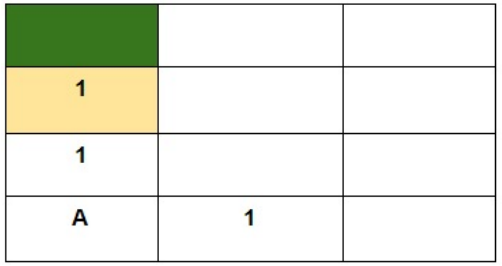
- Suppose a robot needs to go to the block, marked in green from its current position (A) using the specified




- How can the robot do this programmatically? One idea would be to introduce footprint which the robot would be able to follow like below direction.




- The robot now sees footprints in two different directions. It is, therefore, unable to decide which way to go in order to get to the destination (green room). It happens primarily because the robot does not have a way to remember the directions to proceed. So, our job now is to enable the robot with a memory.



# Bellman Equation

$$V(s) = \max_a \left( R(s,a) + \gamma V\left(s'\right) \right)$$

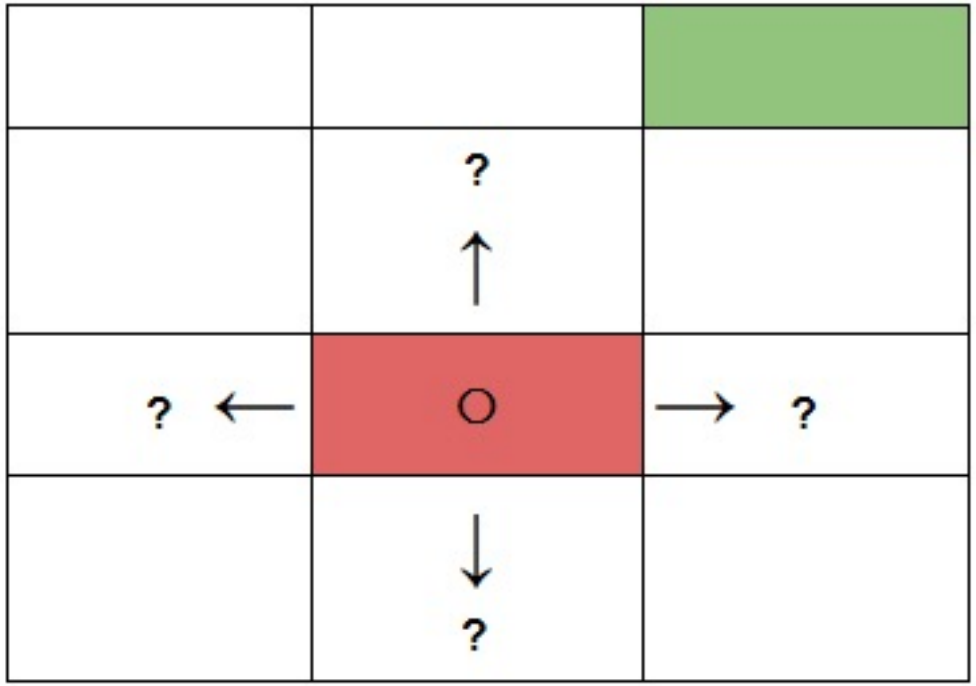
where,

- s = a particular state
- a = action
- s′ = state to which the robot goes from s
- $\gamma$ = discount factor
- R(s, a) = a reward function which takes a state s and action a and outputs a reward value
- V(s) = value of being in a particular state (the footprint)

| | | |
|---|---|---|
| | 1 | 0.9 |
| 1 | 0.9 | 0.81 |
| 0.9 | 0.81 | 0.729 |
| 0.81 | 0.729 | 0.66 |

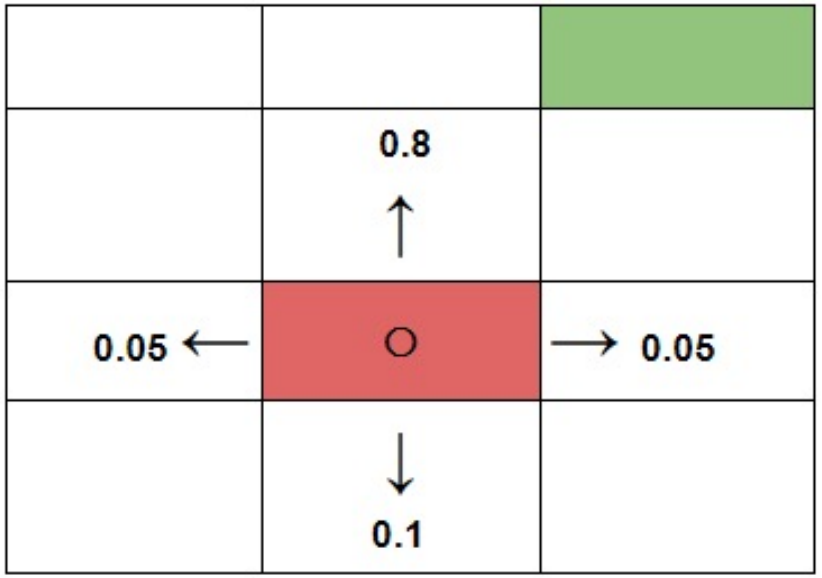
# Modeling stochasticity: Markov Decision Processes

- The robot is currently in the red block, and it needs to go to the green block.
- The robot has a slight chance of dysfunction and might take the left or right or bottom turn instead of taking the upper turn in order to get to the green block from where it is now (red block)
- This is a situation where the decision making regarding which turn is to be taken is partly random and partly under the control of the robot. Partly random because we are not sure when exactly the robot might dysfunction and partly under the control of the robot because it is still deciding of taking a turn on its own and with the help of the program embedded into it.



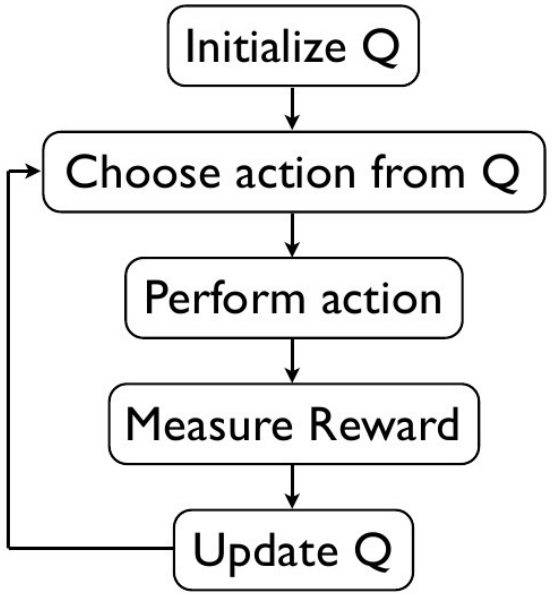
# Solution: Markov Decision Processes

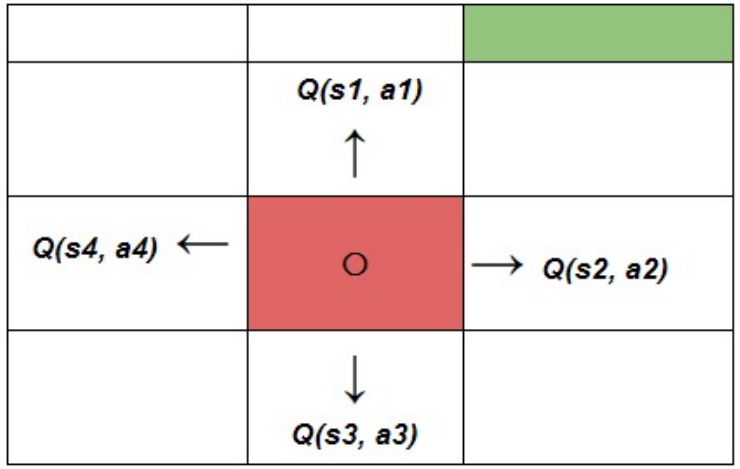
$$V(s) = \max_a \left( R(s,a) + \gamma \sum s' P\left(s,a,s'\right) V\left(s'\right) \right)$$



# Q- Learning

Q learning is a value-based off-policy temporal difference(TD) reinforcement learning. Off-policy means an agent follows a behavior policy for choosing the action to reach the next state s_t+1 from state s_t.




$$Q(s,a) = R(s,a) + \gamma \sum_{s'} \left( P\left(s,a,s'\right) \max_{a'} Q\left(s',a'\right) \right)$$

$$Q^{new}(s_t,a_t) \leftarrow \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left( \underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1},a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t,a_t)}_{\text{old value}} \right)}^{\text{TD error}}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{new value (temporal difference target)}}$$

# Dataset Description

- Movie Lens dataset
- Top 627 movies were picked n terms of maximum frequency of the ratings provided to the movie.
- The movies belonged to multiple genre but for ease of understanding, we have considered each movie has only one genre.
- Coming to formulation on our dataset, if we keep the previous three movies seen by the users as a state then there will be total of 627P3 states which is equal to 245,313,750 and 627 actions. Here we have considered the same three movies having different arrangements as different state. Basically, we are interested in knowing what are the previous movies a user has seen to recommend him/her next movie. So, here order of the movies doesn't matter. So instead of permutations, we dealt with combinations which significantly reduced the number of states to 40,885,625. But still it was a huge number and maintaining such a large Q matrix was way too difficult and taking lots of memories.

# How to build Recommendation Systems using Reinforcement Learning?

## State Space

Movies - 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

States -
1,2,3
1,2,4
.
.
2,3,4
.
.
3,4,5
.
.
.
.
.
8,9,10

Combinations of Movies

States represent the previous 3 movies that are viewed by the user.

State Space = NC3

Now, we can have a user that can enter in any state corresponding to the combinations that we have created.

## Action Space

Movies - 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10

Each movie corresponds to an individual action.
Action Space = Number of movies to Recommend

# How do we use Reinforcement Learning to make Recommendations?

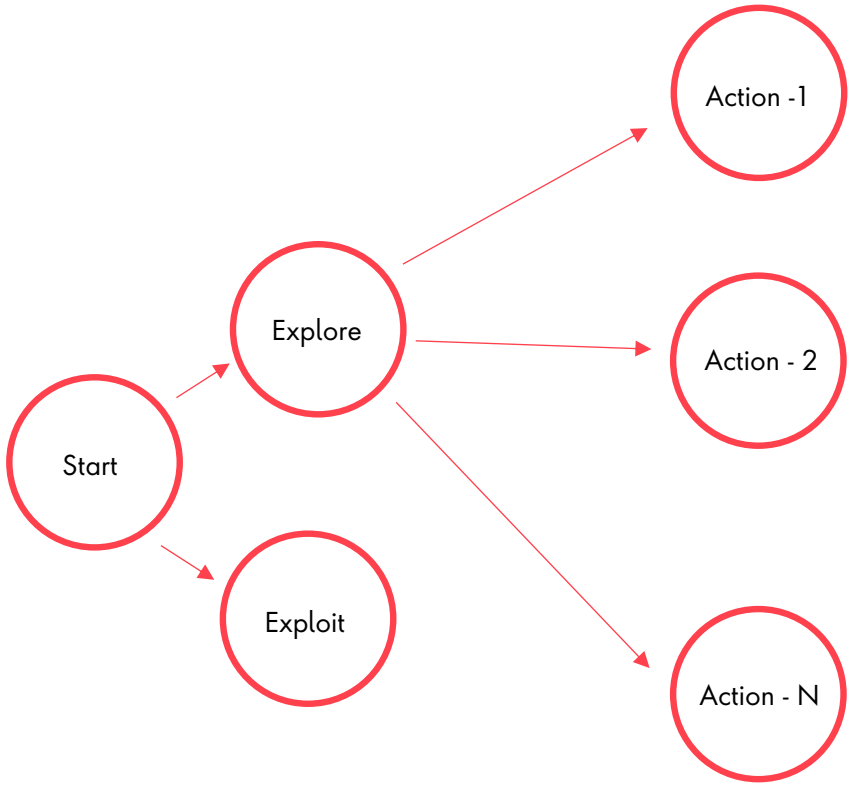
## Q-Learning Algorithm

### Exploration

If the actions taken have Q-Values less than the Epsilon Value, take a random action as recommendation to the user

### Exploitation

If the actions taken have Q-Values more than the Epsilon Value, return the action with the highest Q-Value as recommendation to the user




$$A = \begin{cases} \text{random}(a), & \text{if } Q(s,a) < \varepsilon \\ \max_a Q(s,a), & \text{if } Q(s,a) > \varepsilon \end{cases} \quad \text{where } 0 \leq \varepsilon \leq 1$$

# How to Deep Reinforcement Learning solves the problem?

Actions

| States | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1,2,3 | | | | | | | | | | |
| 1,2,4 | | | | | | | | | | |
| ..... | | | | | | | | | | |
| 3,4,5 | | | | | | | | | | |
| ..... | | | | | | | | | | |
| 7,9,10 | | | | | | | | | | |
| 8,9,10 | | | | | | | | | | |

Q -Table

Policy – Mapping of States and Actions

When we have a large number of Items to Recommend, formulation of a Q-Table is computationally expensive.

Solution – Multiple Q-Tables

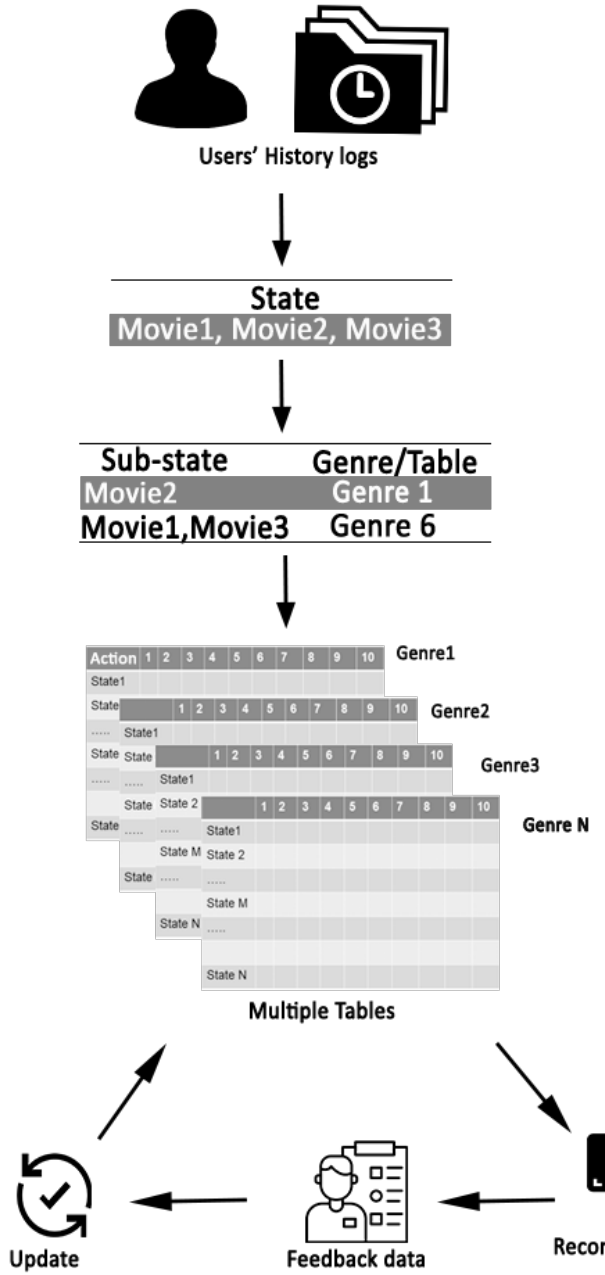
# Multiple Tables Q-Learning

To deal with very large number of  states and action, we decided to make multiple table corresponding to each genre type.

Every Genre gets its own Q-Table.

When a user with State ‘S’ enters, we find the genre of the movies he has watched and sends him to the respective table accordingly.

Recommendations for the user are made from all genres he has watched in decreasing order of Score of Recommendation ( Q-Value)

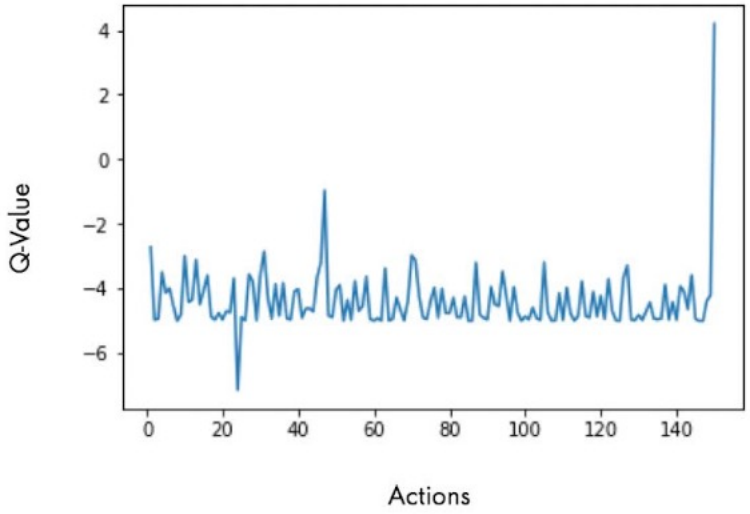


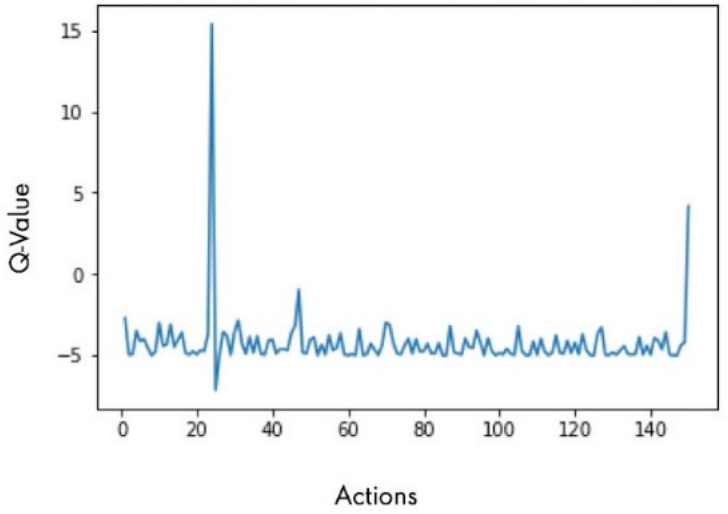
# Scenario – States and Sub-states

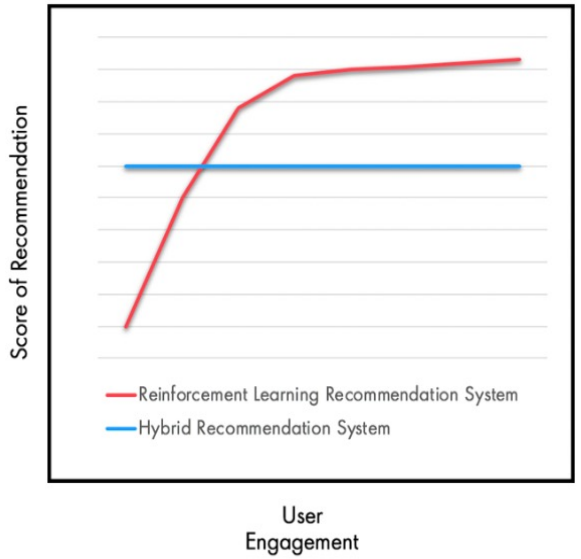
| ***State***- Movie1, Movie2, Movie3 |
|---|

| ***Sub-state*** possibilities | Table |
|---|---|
| All movies are of different genres<br>*Sub-states-*<br>• Movie1- Genre1<br>• Movie2- Genre3<br>• Movie3- Genre5 | Look for respective table and then the sub-state in nC1 states of that table |
| One is of different genre and other two of same henre<br>*Sub-states-*<br>• Movie1- Genre3<br>• Movie2, Movie3 - Genre5 | Look for respective table and then the sub-state in ${}^{n}C_{1}$ states of that table for sub state containing one movie<br><br>Look for respective table and then the sub-state in ${}^{n}C_{2}$ of that table for sub state containing two movies |
| All three movies are of same genre<br>*Sub-state-*<br>• Movie1, Movie2, Movie3- Genre4 | Form ${}^{3}C_{2}$ sub-states and the look for the sub-states in ${}^{n}C_{2}$ states of that table |

# Training and Evaluation



# Code Snippets discussion

# Question & Answer

publicis
sapient

## References:


1. https://blog.floydhub.com/an-introduction-to-q-learning-reinforcement-learning/

2. https://github.com/raviranjan0309/ODSC-conference-India-2021/blob/main/README.md