

ODSC INDIA
VIRTUAL CONFERENCE

Deep Reinforcement Learning Based RecSys Using Distributed Q Table



**December
8th — 9th**

Ravi Ranjan
Senior Data Scientist
Publicis Sapient

Session Logistics

1. Access to the session environment using the following link. [https://bit.ly/join_the_session]
2. Presentation and research paper will be available at link. [<https://bit.ly/ODSC-conference-India-2020>]
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 #ODSCVirtual

About the Speaker

- Working as Senior Data Scientist for Publicis Sapient
- AI and ML at scale with expertise in building enterprise solutions and ML Engineering.
- Certified Google Cloud Architect
- Kubeflow member and contributor

Learning Outcome

1. Gain an understanding of deep reinforcement learning-based recommendation system.
2. How to train and evaluate the RL model with distributed Q-table?
3. 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

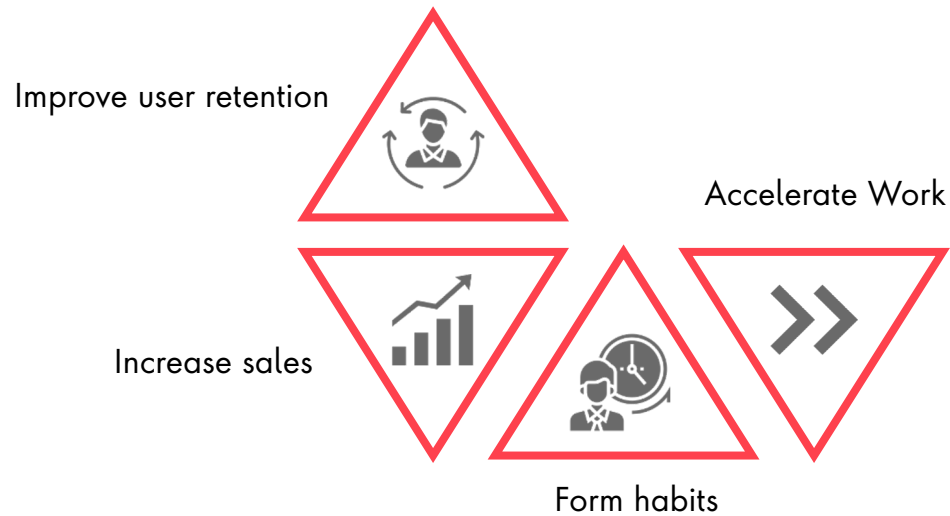
What is Recommendation Systems?

“Serve the **relevant** items to users in an **automated** fashion to optimize **short- and long-term business objectives**”

Usage of Recommendation system:

- Product recommendation in ecommerce website
- Video recommendation in online streaming platform
- Songs recommendation in music listening application
- Personalize user experience on any application

Why companies implement Recommendation Systems?

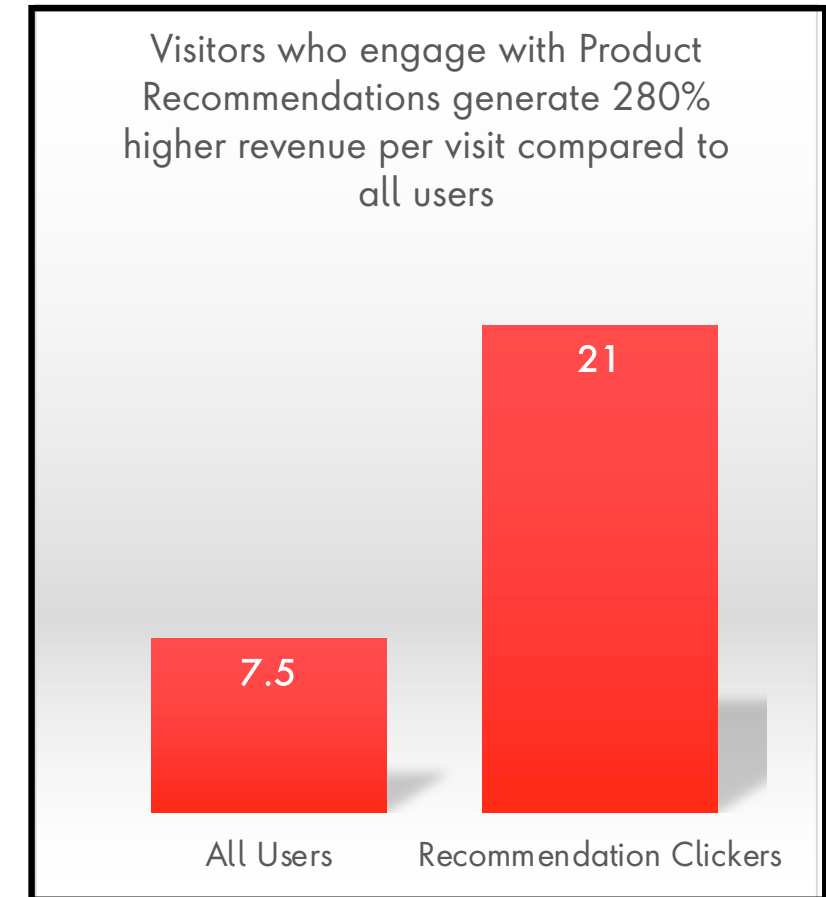


Enterprises using Recommendation Systems

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

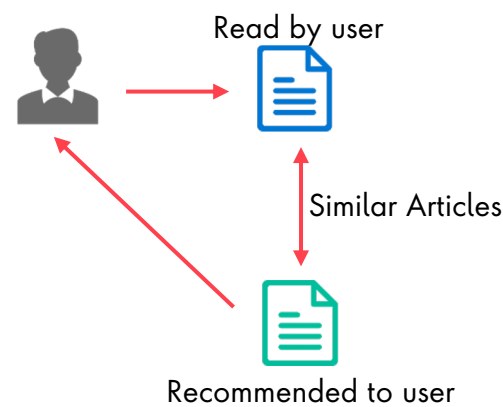
Amazon credits recommender systems with 35% of its revenue

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

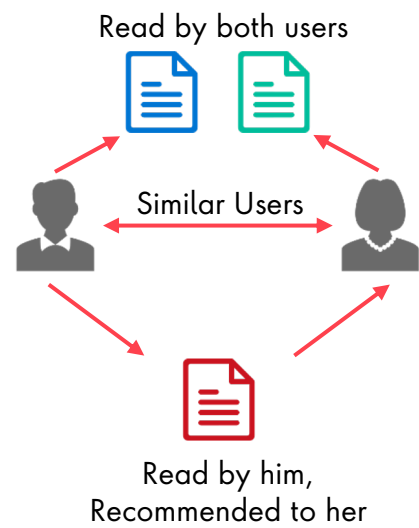


Classical Approaches to build Recommendation System

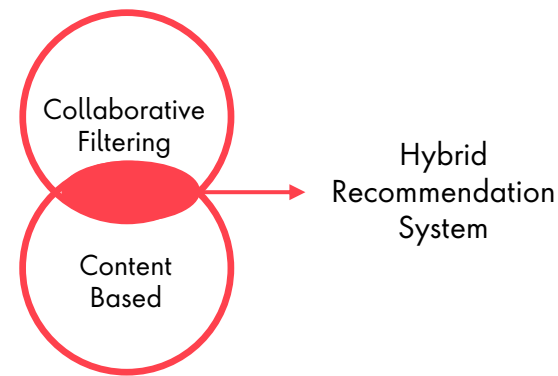
Content Based



Collaborative Filtering



Hybrid Recommendation System



Limitations of the Classical Approaches



Changing Data



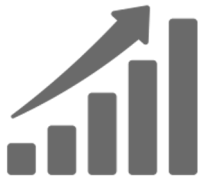
Dynamic User Behavior



Static Recommendations



Grey Sheep



Scalability



Cold Start

Section 2

Reinforcement Learning

Why need Reinforcement Learning to develop Recommendation Systems?



Works on Changing Data



Tackles Changing User Taste



Tackles Static Recommendations



Grey Sheep



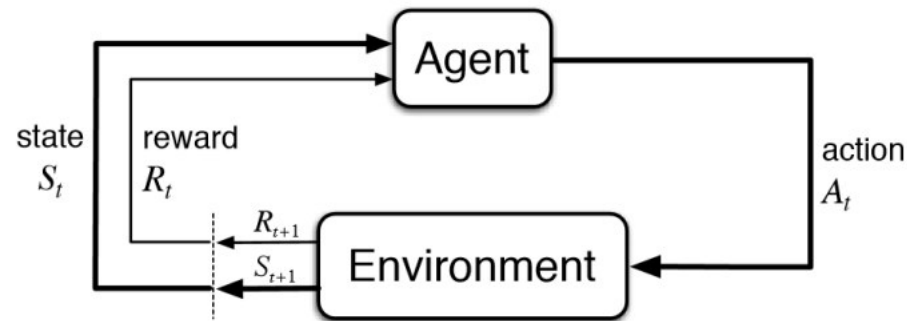
Utilizes User Feedback to Give Recommendations



User Retention

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.



How do we use Reinforcement Learning to make Recommendations?

Q-Learning Algorithm



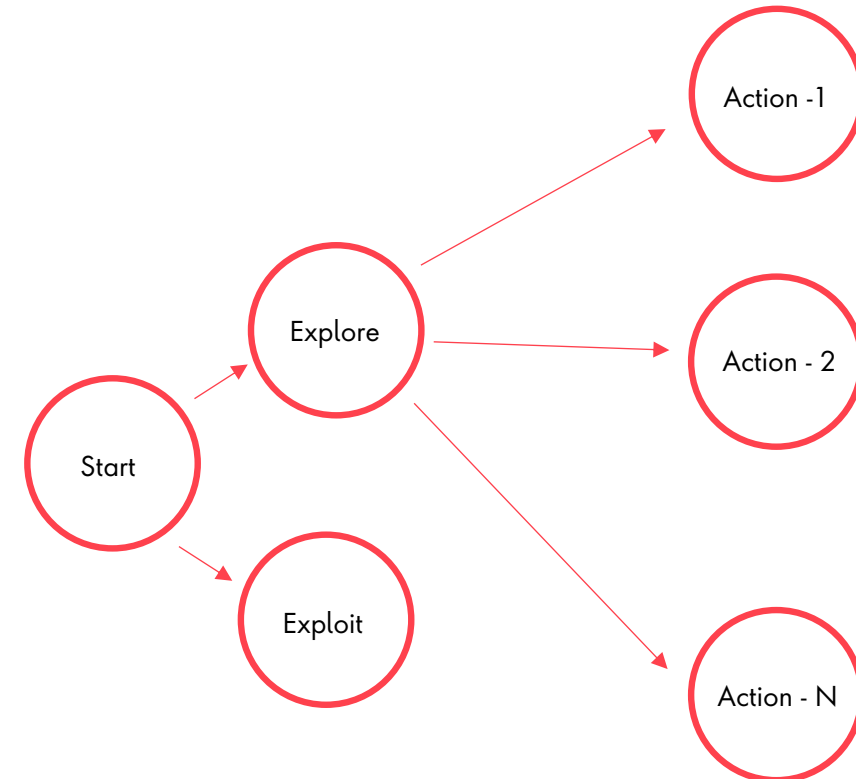
Exploration

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




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 = NC^3

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 to Deep Reinforcement Learning solves the problem?

States

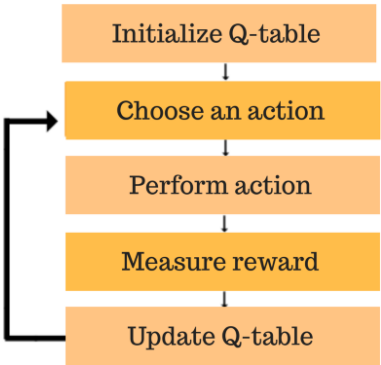
Actions										
	1	2	3	4	5	6	7	8	9	10
1,2,3										
1,2,4										
.....										
3,4,5										
.....										
7,9,10										
8,9,10										

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



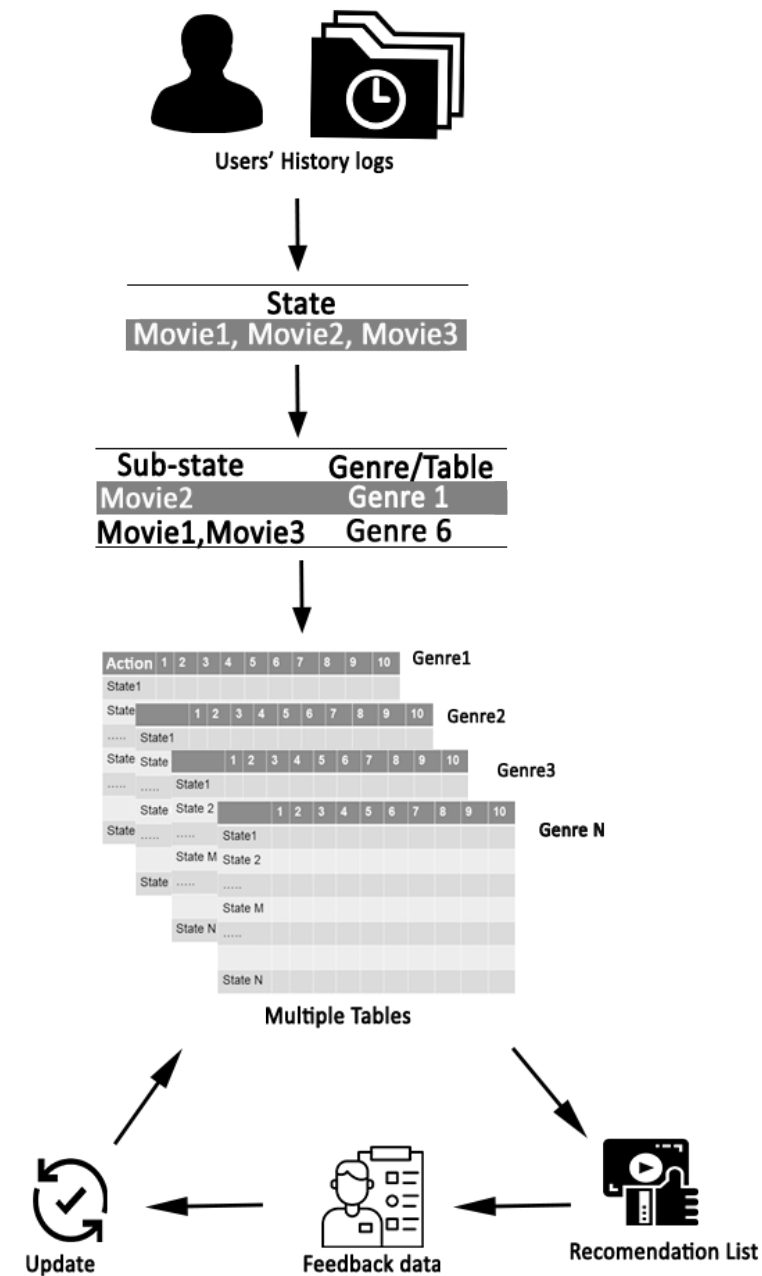
Multiple Tables Q-Learning

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

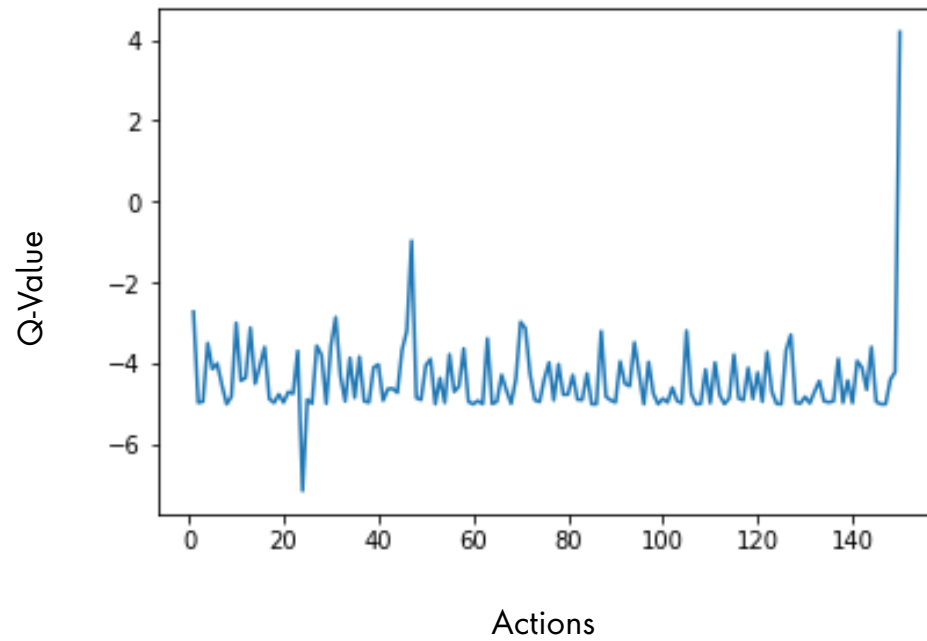
Every Genre gets its own Q-Table.

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.

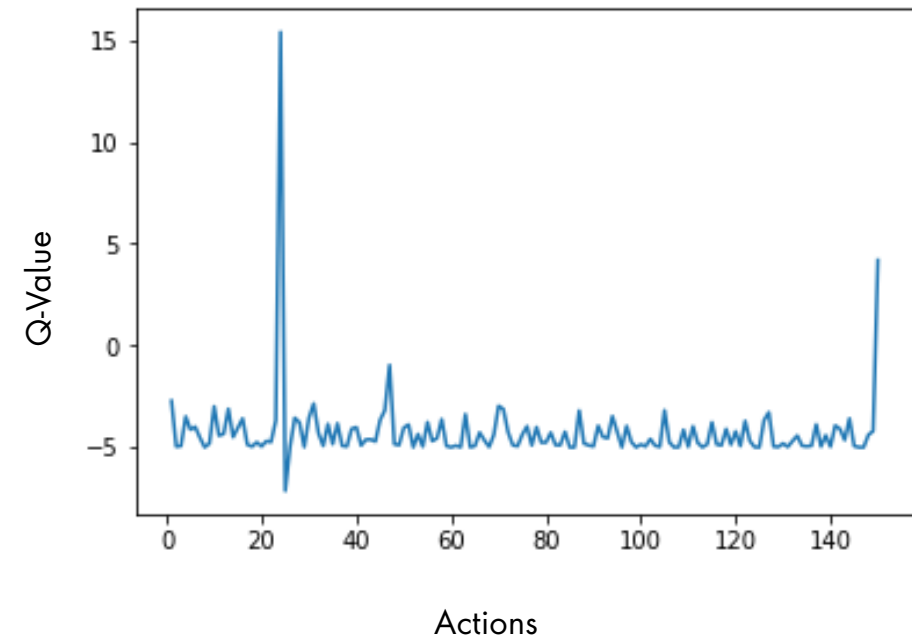
Recommendations for the user are made from all genres he has watched in decreasing order of Score of Recommendation (Q-Value)



Evaluation



Evaluation using traditional approach



Evaluation using distributed Q-learning approach

Question & Answer