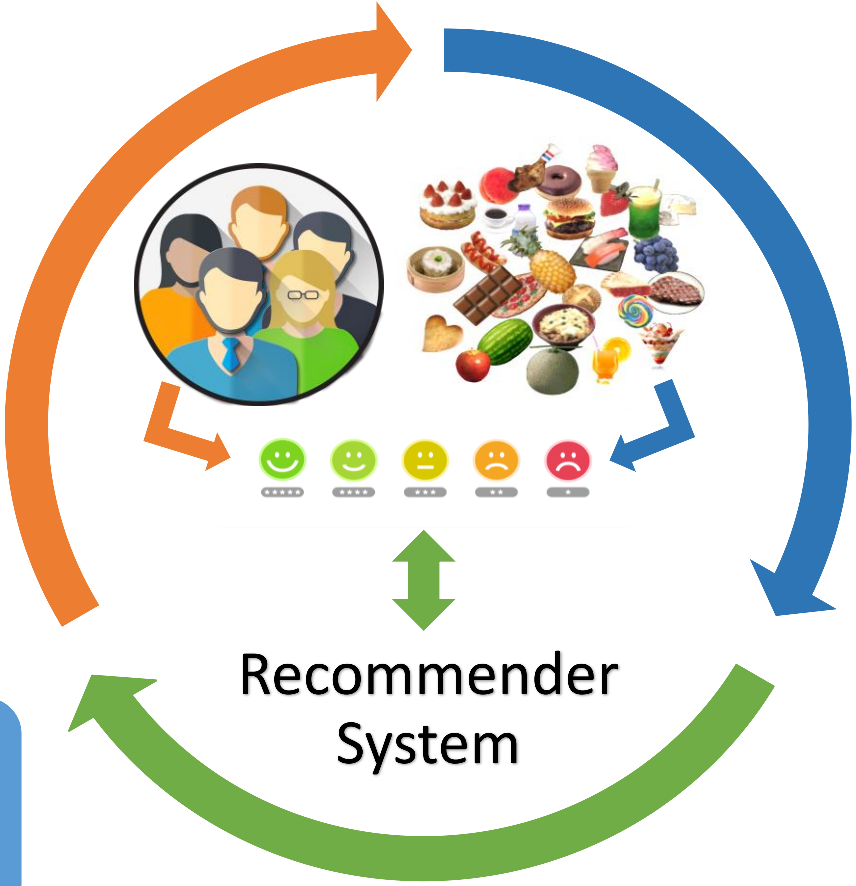
Case Study of Reinforcement Learning on Yelp’s restaurants recommendations

Sriharsha Jana
MS in Machine Learning and AI

Recommender Systems

- Recommender Systems are very essential in the current world, where we have very huge information overload
- Users , Items and Ratings are building blocks.
- Implicit and explicit feedback mechanisms
- Contextual information about items, users and side car information



Yelp, Zomato, Swiggy have restaurant Data

What user Browsed, Rated, clicked, Social Network

Recommender Systems

Top Picks to user

You Might like this

Try this new taste

Literature Review – Recommender Systems

Types Of Recommender Systems

Collaborative Filtering

Model Based

- Clustering
- Association (Matrix Factorization)
- Bayesian approach
- Neural Networks

Memory Based

- User-User
- Item-Item

Content Based

Similarity

- Unsupervised learning
- Supervised Models

Hybrid Techniques

Restaurant Recommendation – Until now…….

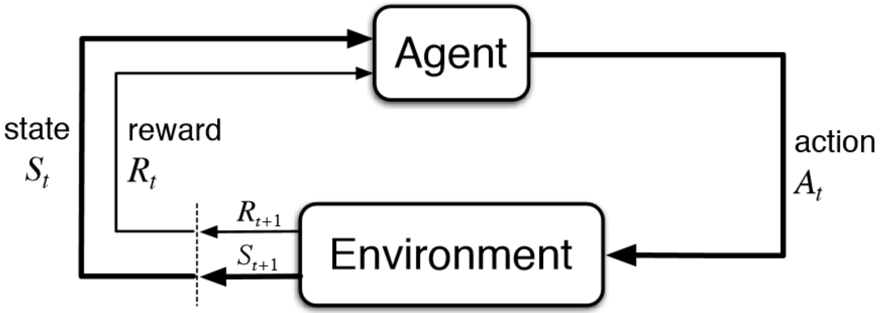
Most commonly used techniques used for Restaurant recommendations are Content based filtering, collaborative filtering and hybrid recommender techniques such as knowledge based and demographic properties.

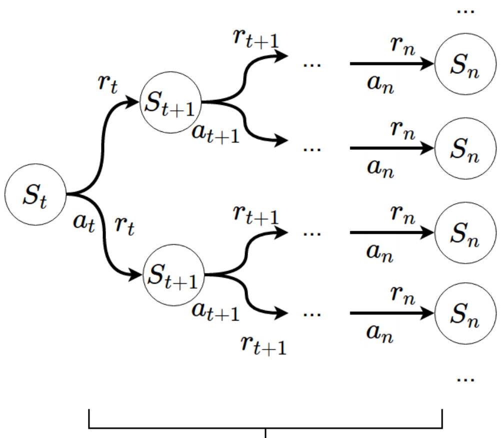
The application of techniques were focused either increase the rating prediction accuracy of CBF and CF techniques and some case studies focused on personalized recommendations using contextual features with hybrid techniques.

Data Sources-Recommendation

- Restaurant Data
- User Data
- Image, Social Networks
- Reviews Data

Literature Review – Reinfocement Learning





$$E_\pi\left[\sum_{i=t}^{n}\gamma^{i-t}r_i\right]$$

- Reinforcement Learning is used in cases where there is explicit reward in the learning process.
- Given a initial state, Agent will try to take an action for which it will receive a Reward.
- In addition Agent will receive transitioned state, from initial state.
- An episode is round about across agent and environment until end state is reached. Can be stochastic or deterministic
- The episodic task is formulated as Markov Decision Process(MDP)

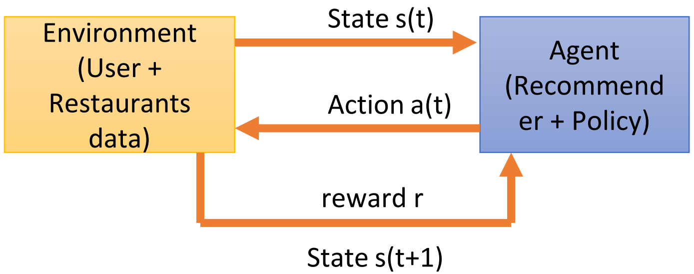
Literature Review – Supervised vs Reinforcement Learning

Supervised Learning	Reinforcement Learning
in **supervised learning**, you deal with **objects or datasets**. There is **no interaction** with the environment and given a dataset, you are required to predict the target	In **reinforcement learning**, you deal with the **processes** where the **agent actively interacts** with the environment
supervised learning is <u>**passive learning**</u>, where the agent learns only by extracting features from a given dataset	RL is an <u>**active learning**</u>, where the agent learns only by interacting
In supervised learning, there is a **teacher (ground-truth)** which tells you whether the result for a given observation is correct or not	The environment acts only as a **critic**, where it tells you how good or bad the action is by giving rewards.
Minimizes the loss function with the ground truth value. In case of **regression mean square error** or in case of **classification sigmoid function**.	Objective function is to **maximize the total rewards** by interacting with environment. Users **exploration and exploitation** techniques.

Problem Statement

1. Current Recommender Systems use Supervised Techniques. These are static in nature. – examples: MF, SVM, DNN.
   - Data Sparsity → Not many users provide reviews, but few users provide lot of reviews
   - Capture Only Current Rewards → Missed to capture the user dynamic change in tastes and preferences
   - Offline/Online learning missing → Missing capturing explicit Feedback
2. Explicit Modelling Required for Ranking recommendations → Explicit training of IR techniques with/without CBF
3. Use & Restaurant contextual information, Type of food, history of ratings → Large number of dimension for features
4. Formulating Recommendation as MDP → Successfully applied in Music, Movie, New/Articles domain.

Aims & Objectives



User Environment

- Use of collaborative Filtering techniques for user reward simulation
- Historic ratings of user over time period as base truth for this simulation

Learning Policy

- The aim is to model the dynamically changing user preferences through RL Agent
- Evaluate performance of RL agent using MAP and NDCG

Large Action space

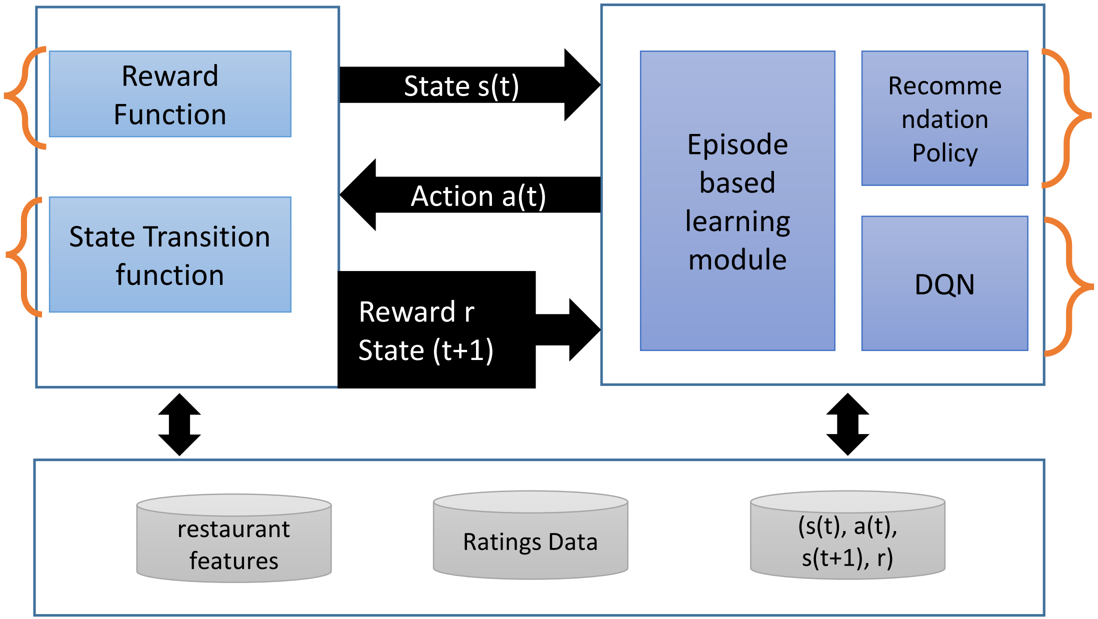
- Where action space is large like recommender systems, exploration becomes time consuming
- Use Content based Filtering Techniques to handle the large action space.

Proposed Methodology - RL System Design

The formal setting of an episode in reinforcement learning will be a tuple of (S(t), A, S(t+1), R) , where each variable in the tuple is explained below.

r = $f(a(t))$

s(t+1) := s(t)



$a(t) = f(s(t))$

$Q(s,a) = f(s(t), a(t))$

(Nomenclature in appendix)

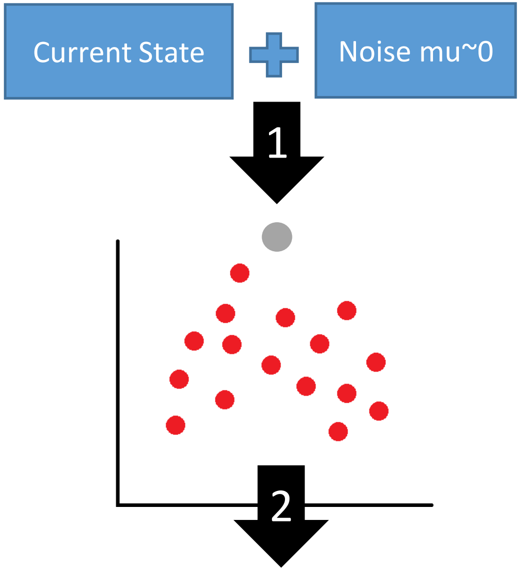
Proposed Methodology – User Environment Simulation

- The user environment considers the user historic browsed/rated restaurants.
- Based on collaborative filtering techniques, users with similar interests will provide similar ratings to the restaurants.
- Discrete rewards with cosine similarity between state and action

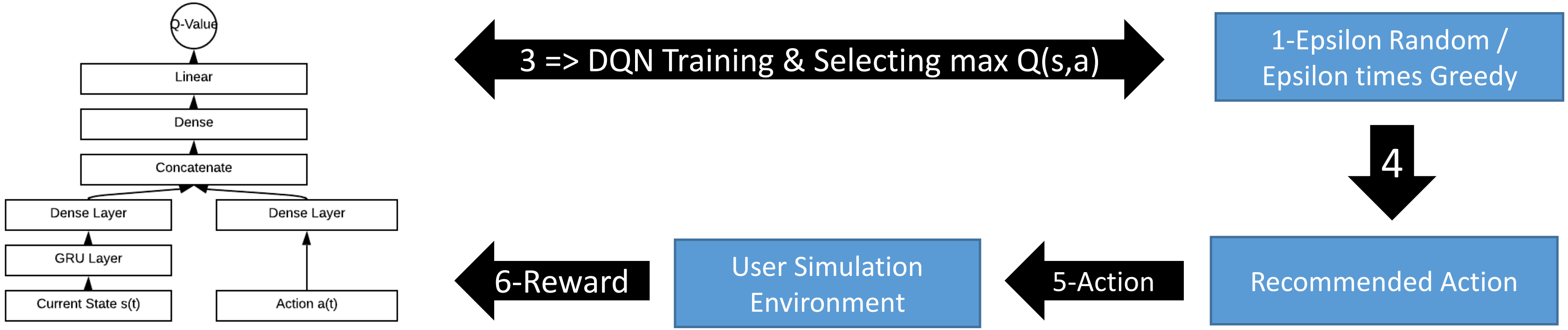
$$Cosine(p_t, m_i) = \alpha \frac{s_t s_i^\top}{\|s_t\|\|s_i\|} + (1-\alpha)\frac{a_t a_i^\top}{\|a_t\|\|a_i\|},$$

Serial Number	Historic <State, Action>	Rating	<Current State, Recom Action>	Cosine Sim
1	<(x1,x2,x3), x4>	5	<(x1,x2,x3), x5>	.9
2	<(x4,x5,x6), x6>	4	<(x1,x2,x3), x5>	.7
3	<(x7,x8,x9), x10>	2	<(x1,x2,x3), x5>	.3

Proposed Methodology – NN-Epsilon Greedy



1. Since there will be very large number of items, Calculating the state-action value for each item is over head for the deep network architecture.
2. Use action and current state as input to the network, to learn the state-action value. Which will be used for learn a policy
3. Generate a query vector from current state and adding some mean 0 noise. Query K nearest neighbours based on the latter.
4. Perform Epsilon-Greedy policy to choose an Action as recommendation. Soft update DQN or Deep-SARSA Policy learning performed.



Evaluation Metrics

MAP – Mean Average Precision

- Average precision at K is defined as the ration of relevant items by total items recommended.
- In RL setting, each episode will be of length N steps. For each episode the recommendations provided is considered for Average precision.
- Summation over multiple users will give Mean average precision

$$\text{MAP@N} = \frac{1}{|U|}\sum_{u=1}^{|}U|(\text{AP@N})_u = \frac{1}{|U|}\sum_{u=1}^{|}U|\frac{1}{m}\sum_{k=1}^{N} P_u(k)\cdot rel_u(k).$$

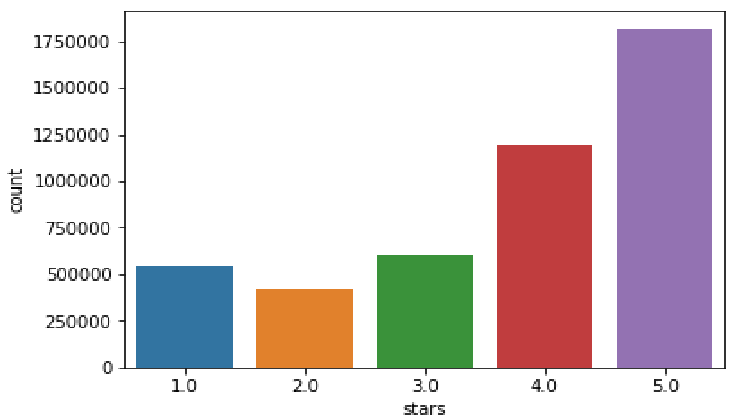
nDCG – Normalized Discounted Cumulative Gain

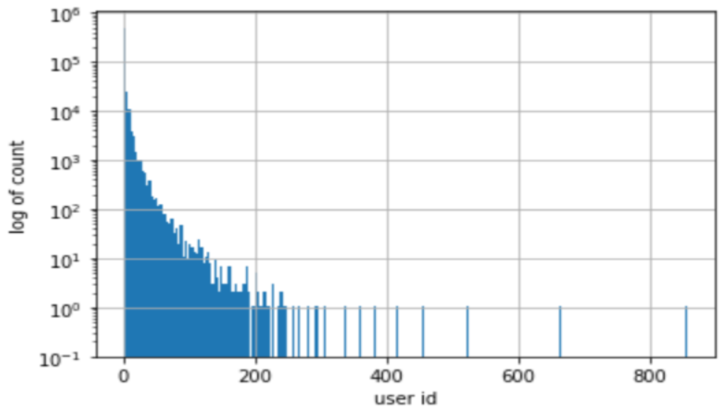
- Uses graded relevance as a measure of usefulness, or gain, from examining a recommendation
- Gain is accumulated starting at the top of the ranking and may be reduced, or discounted, at lower ranks

$$DCG = \sum_{pos=1}^{n} \frac{relevance_{pos}}{ln(pos+1)} \qquad NDCG_{pos} = \frac{DCG_{pos}}{iDCG}$$

Yelp’s Business data – EDA

- The Yelp data set consists of business, reviews, user and tips data
- The total data set contains around 4.5 million reviews provided by 1.1 million users on 74000 businesses
- The total dataset size is around 4GB.
- The ratings are real valued positive integers with are ordinal in nature and follow 1 to 5, 1 being very bad and 5 being good. User rating follow long tail distribution.
- Select only restaurant data using category keywords like food, restaurant. Filtered data consists of 19560 restaurants across 7 US states.
- Filtered reviews data on upper and lowed bounds on reviews counts is 372K.

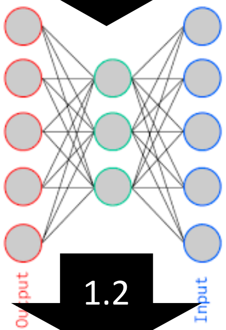




Data Pre-Process Flow

Type	Contextual attributes	Derived Features
Restaurant	Latitude, longitude, Alcohol, Noise Level, Restaurants Attire, takeout, ambience, open hours, word based categories	~1034

1.1



1.2

Restaurant Embedding's

User ID	Business ID	Stars	Date Time

2.1

K-Window Function grouped by Users

2.2

Each Session - first 80% train and last 20% test

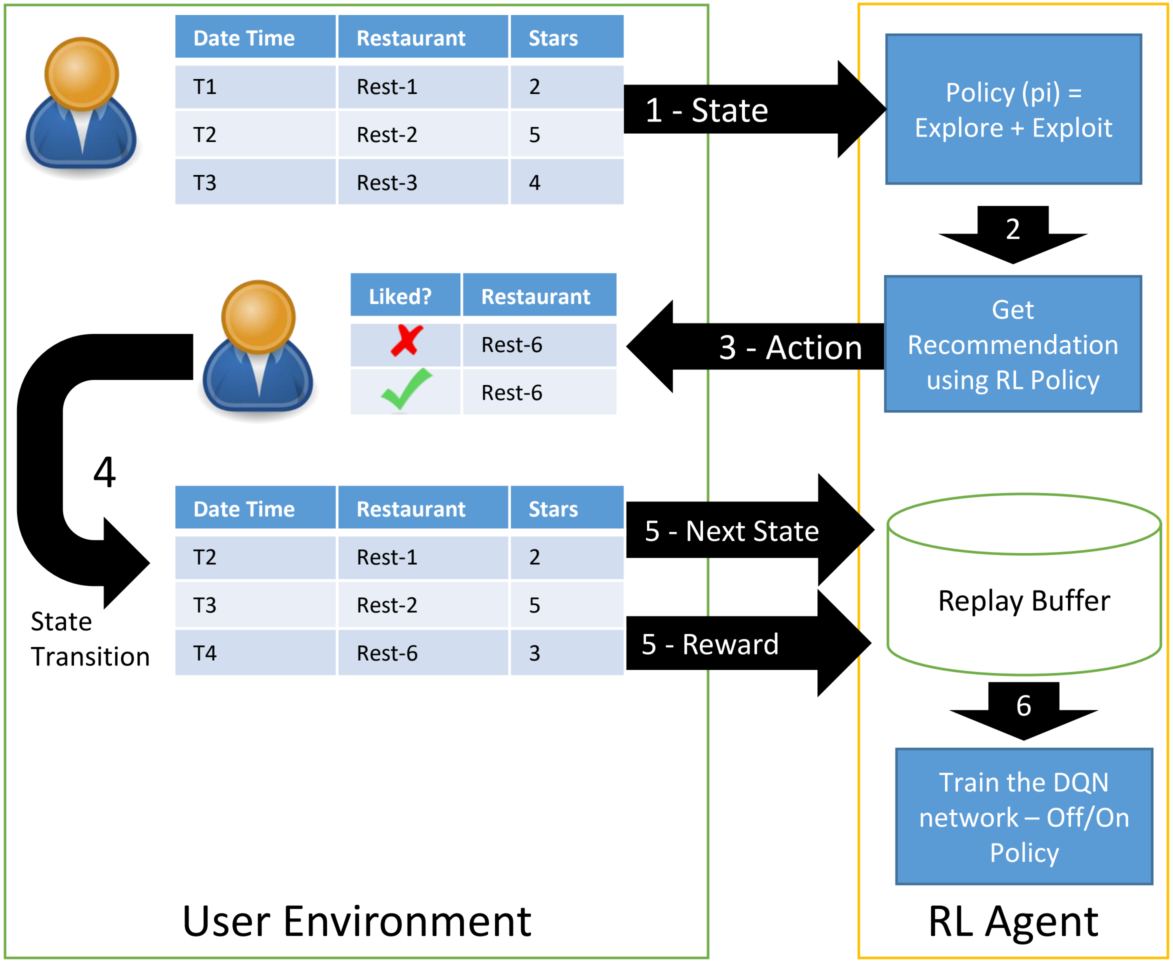
2.3

Train Set

Test Set

RL Framework

RL Framework Training

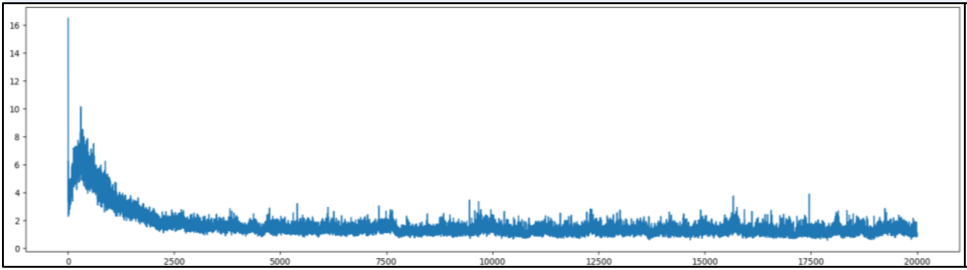


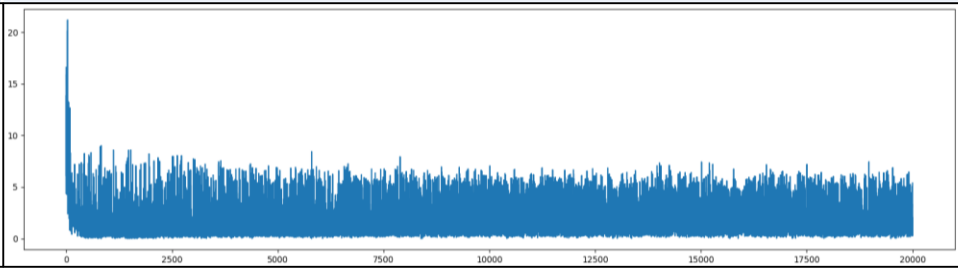

```
function QueryNearestNeighbourActions(CurrentState)
    NNModel ←Load the Nearest Neighbour Model
    QueryActions ←Initialize empty list
    for State ∈ CurrentState do
        Noise ← GuassianNoise(μ = 0, σ = θ)
        ModifiedState ← Noise + CurrentState
        KActions ← NNModel(ModifiedState)
        KActions append to QueryActions
    return QueryActions

function GetRecommendation(CurrentState, I, DQN)
    QValueList ←Initialize empty list
    QueryAction ← QueryNearestNeighbourActions(CurrentState)
    for Action ∈ QueryActions do
        QValue ←DQN.Predict(CurrentState , QueryActions)
        QValue append to QValueList
    return Action corresponding to argmax(QValueList)

Initialize the K for nearest neighbour Query
Initialize Parameters tracking for Rewards, loss and q-value
Create User Environment Simulator with α parameter
Create DQN with learning rate β
Initialize buffer memory of D length
for epochs = 1, M do
    Reset action space of restaurants to I s(t) Initial state from user history
    for t = 1, T do
        Select an action a(t) from I using GetRecommendation(s(t))
        Execute action a(t) and observe reward r and next state s(t+1)
        Append s(t), a(t), r, s(t+1) to memory D
        s(t) = s(t+1)
        if policy = on-policy then
            get a(t+1) using GetRecommendation(s(t + 1))
            get Q(s(t+1), a(t+1)) from DQN
            Q(s(t), a(t)) := r + discount-factor * Q(s(t+1), a(t+1))
            Update DQN parameters using Q(s(t), a(t))
    if policy = off-policy then
        sample mini-batch M from D
        for m = 1, M do
            get a(m+1) using GetRecommendation(s(m + 1))
            get Q(s(m+1), a(m+1)) from DQN
            Q(s(m), a(m)) := r + γ * Q(s(m+1), a(m+1))
            Update DQN parameters using Q(s(m), a(m))
```

Results Discussion – RL Convergence

Q-Network Loss Function – network loss vs episodes

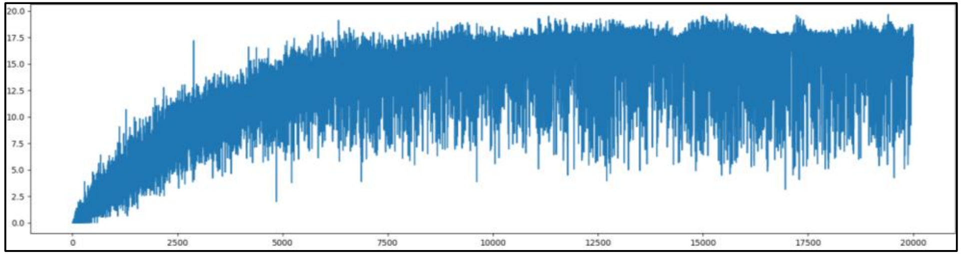
DQN	Deep SARSA
Shows convergence of the loss value of the DQN, due to the soft update DQN methodology. Good generalization observed	Overfitting of the loss value observed and hence no convergence observed in DQN used.

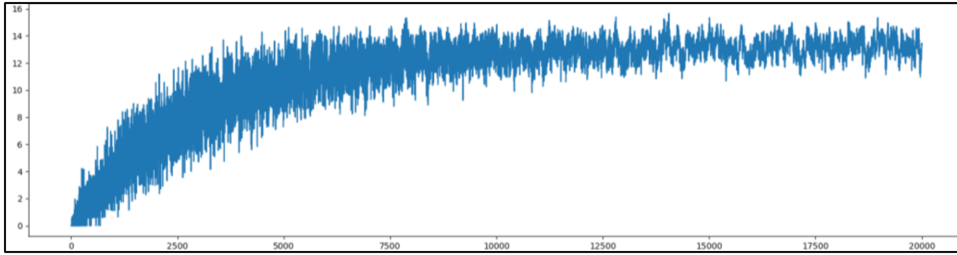




Q-Value convergence – Q-value vs episodes

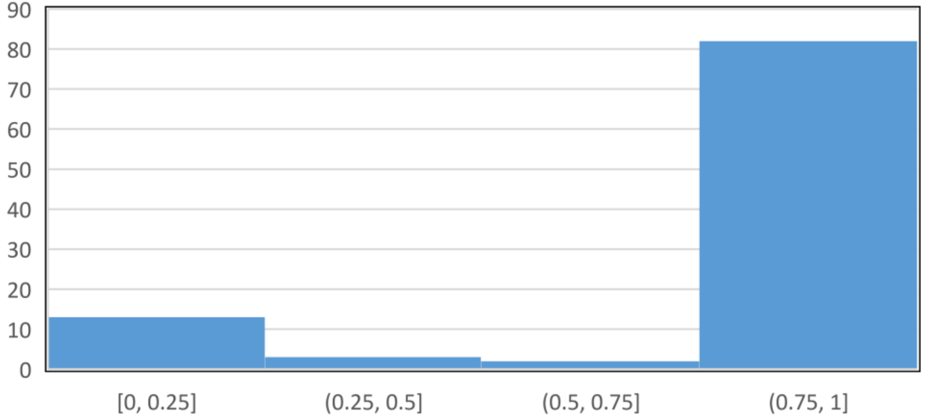
DQN	Deep SARSA
The Q-value is having constant upper bound value. Since there is exploration, the q value variance is observed	The graph shows the convergence of the q value and fluctuations are observed due to overfitting.



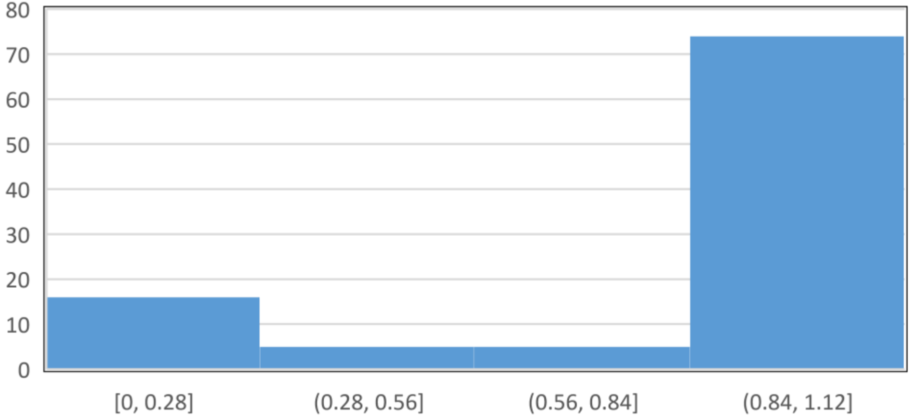


Results Discussion – Accuracy/Performance

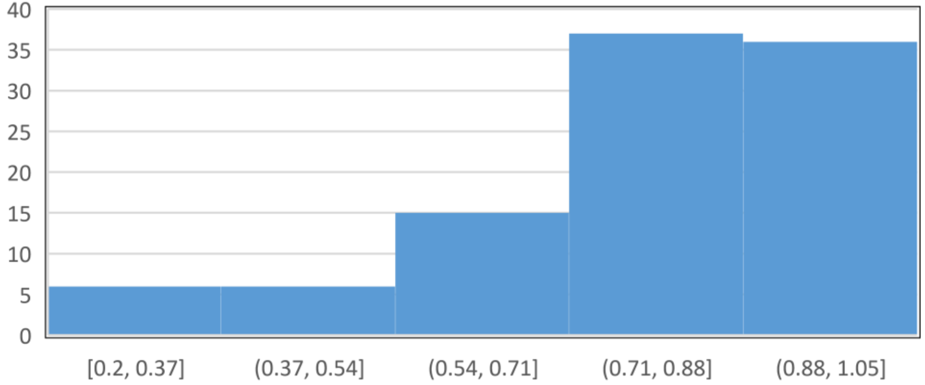
MAP – Test - DQN



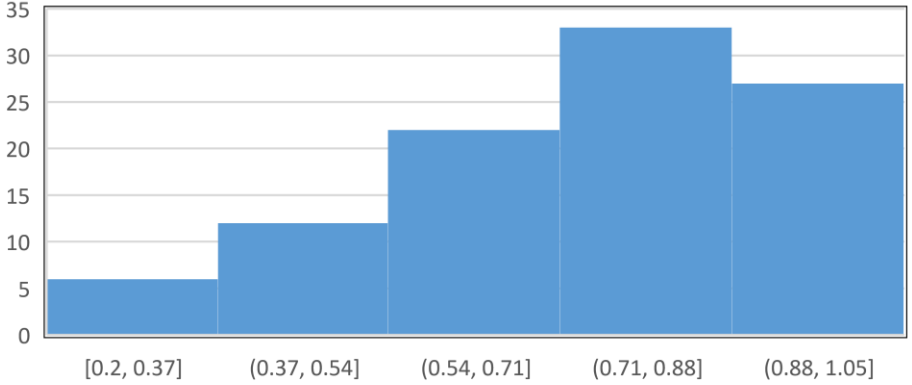
MAP - Test - DeepSARSA



nDCG - Test - DQN



nDCG - Test - DeepSARSA



Future Scope & Conclusion

Conclusion

- Application of RL techniques for recommendation in e-commerce , new articles can be extended for Restaurant recommendations as well. This is a knowledge addition for Yelp's Restaurant data
- Effective recommendation policy by using epsilon greedy strategy and applying Nearest Neighbours for handling large action space.
- Using Nearest Neighbour causes deterministic policy, applied zero mean noise with controlled variance to current state for optimal recommendation strategy.
- Successfully applied Deep-SARSA and Deep Q-learning technique of soft update DQN for optimal recommendation policy.

Future Scope

- Yelp's data set has review text data that effectively when incorporated as sentiment analysis into user-environment simulation for rating a restaurant
- The learning policies rely upon Deep Q learning and Temporal Difference techniques in RL. Try out Policy Gradient algorithm in RL is to optimize the recommendations
- For user environment is simulated using cosine similarity based on user historic state action ratings. This can be converted into a neural network with state action as input and rating as output.

Thank you

Appendix

Reinfocement Learning Nomenclature

RL Component	Nomenclature	Recommender System Settings	Example/function
State Space	$s(t)$	K-windowed previous state of the user at time t, where $x(i)$ $\{x(1), x(2)......x(n)\}$. $X(n)$ being feature vector of restaurant n.	x1 x2 x3 x4 x5
Action	$a(t)$	Single action from the RL Agent based on state space $s(t)$	x6
Reward	r	Based on user history, reward R set of real value	$\{1,2,3,4,5\}$
Transition State	S'	If user likes the action then transition to $s(t+1)$, else be at $s(t)$	x2 x3 x4 x5 x6
Discount Factor	γ	Discount factor for the future rewards	
Policy	$\pi(a\|s)$	Probability to take action a given state s.	
State value	$v_\pi(s)$	represents the value of the state s.	$\sum_a \pi(a\|s) q_\pi(s,a)$
Action Value	$q_\pi(s,a)$	represents the value of performing a particular action a while in state s	$\sum_{s'} \sum_r p(s',r\|s,a)(r+\gamma v_\pi(s'))$

Model Free Q-Learning

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma * \text{argmax} Q(s',a) - Q(s,a))]$$

References

1. Beel, J., Gipp, B., Langer, S. et al. Int J Digit Libr (2016) 17: 305. Research-paper recommender systems: a literature survey 10.1007/s00799-015-0156-0
2. Zhang, S.Y., Yao, L., Sun, A., & Tay, Y. (2017). Deep Learning Based Recommender System: A Survey and New Perspectives. ArXiv, abs/1707.07435.
3. Kaluza, C.D. (2016). Recommender System for Yelp Dataset CS 6220 Data Mining Northeastern University.
4. Lei, Y., & Li, W. (2019). When Collaborative Filtering Meets Reinforcement Learning. ArXiv, abs/1902.00715.
5. Zheng, Guanjie & Zhang, Fuzheng & Zheng, Zihan & Xiang, Yang & Yuan, Nicholas & Xie, Xing & Li, Zhenhui. (2018). DRN: A Deep Reinforcement Learning Framework for News Recommendation. 167-176. 10.1145/3178876.3185994.
6. M. Fu, H. Qu, Z. Yi, L. Lu and Y. Liu, "A Novel Deep Learning-Based Collaborative Filtering Model for Recommendation System," in IEEE Transactions on Cybernetics, vol. 49, no. 3, pp. 1084-1096, March 2019.doi: 10.1109/TCYB.2018.2795041
7. Zhao, X., Zhang, L., Ding, Z., Yin, D., Zhao, Y., & Tang, J. (2017). Deep Reinforcement Learning for List-wise Recommendations. ArXiv, abs/1801.00209.
8. Choi, S., Ha, H., Hwang, U., Kim, C., Ha, J., & Yoon, S. (2018). Reinforcement Learning based Recommender System using Biclustering Technique. ArXiv, abs/1801.05532.
9. Silveira, T., Zhang, M., Lin, X. et al. Int. J. Mach. Learn. & Cyber. (2019) 10: 813. How good your recommender system is? A survey on evaluations in recommendation 10.1007/s13042-017-0762-9
10. Chen, Hung-Hsuan & Chung, Chu-An & Huang, Hsin-Chien & Tsui, Wen. (2017). Common Pitfalls in Training and Evaluating Recommender Systems. ACM SIGKDD Explorations Newsletter. 19. 37-45. 10.1145/3137597.3137601.
11. Seyednezhad, S.M., Cozart, K.N., Bowllan, J.A., & Smith, A.O. (2018). A Review on Recommendation Systems: Context-aware to Social-based. *ArXiv, abs/1811.11866*.

Additional References

1. Charu C Aggarwal, (2016). Recommender Systems the Textbook, Springer ISBN 978-3-319-29657-9
2. Sergey Levine, Deep Reinforcement Learning (2017), http://rail.eecs.berkeley.edu/deeprlcourse-fa17/index.html
3. Xing Xie, Jianxun Lian, Zheng Liu, Xiting Wang, Fangzhao Wu, Hongwei Wang, and Zhongxia Chen (November 2 2018), https://www.microsoft.com/en-us/research/lab/microsoft-research-asia/articles/personalized-recommendation-systems/
4. The Lazy Programmer, Recommender Systems in python, (all videos) https://www.udemy.com/recommender-systems/
5. Minmin Chen, ACM Channel, Reinforcement Learning for Recommender Systems: A Case Study on Youtube, (March 28 2019) https://www.youtube.com/watch?v=HEqQ2_1XRTs