Recommendation System for E- Commerce

Anirudh Kovuru - 20161189
Rachit Jain - 20161005
Sri Keshav Kothapalli - 20161023

Guide: Prof. Praveen Paruchuri

Problem statement

Recommendation System

A recommender system or a recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications.

Once the user makes a choice, a new list of recommended items is presented. Thus, the recommendation process is a sequential process.

An MDP based recommendation system

Taking this idea one step farther, we suggest that recommending is not simply a sequential prediction problem, but rather, a sequential decision problem.

At each point the Recommender system makes a decision: which recommendation to issue.

This decision should be optimized taking into account the sequential process involved and the optimization criteria suitable for the Recommender system. Thus, we suggest the use of Markov Decision Processes

An MDP based recommendation system

With this view in mind, a more sophisticated approach to recommender systems emerges.

First, one can take into account the utility of a particular recommendation – for example, we might want to recommend a product that has a slightly lower probability of being bought, but generates higher profits.

Second, we might suggest an item whose immediate reward is lower, but leads to more likely or more profitable rewards in the future.

An MDP based recommendation system

These considerations are taken into account automatically by any good or optimal policy generated for an MDP model of the recommendation process.

In particular, an optimal policy will take into account the likelihood of a recommendation to be accepted by the user, the immediate value to the site of such an acceptance, and the long-term implications of this on the user's future choices.

These considerations are taken with the appropriate balance to ensure the generation of the maximal expected reward stream.

Benefits of this model

The benefits of an MDP-based recommender system discussed above are offset by the fact that the model parameters are unknown. Standard reinforcement learning techniques that learn optimal behaviors will not do, they take considerable time to converge and their initial behavior is random. No commercial site will deploy a system with such behavior.

Thus, we must find ways for generating good initial estimates for the MDP parameters. The approach we suggest initializes a predictive model of user behavior using data gathered on the site prior to the implementation of the recommender system. We then use the predictive model to provide initial parameters for the MDP.

Dataset used

1. The dataset we used was collected from Steam, one of the most popular PC gaming hubs with over 6000 games and community of millions of gamers.
2. This dataset is a list of user behaviors, with columns: user-id, game-title, behavior-name, value. The behaviors included are 'purchase' and 'play'.

Research Paper

An MDP-Based Recommender System
Guy Shani, Ronen I. Brafman, David Heckerman

We are following the above research paper where recommendation systems have been modelled as sequential problems.

The sequential nature of the process is taken into account and the use of MDP's is suggested.

The system not only takes into account the likeliness of the product being relevant but also the profitability of suggesting that product.

For instance, consider a site selling electronic appliances faced with the option to suggest a video camera with a success probability of 0.5, or a VCR with a probability of 0.6.

The site may choose the camera, which is less profitable, because the camera has accessories that are likely to be purchased, where as the VCR does not.

Some Background

The First Order Markov Chain (MC)

- The model is described as a first order Markov chain.
- A Markov chain (MC) consists of a set of states, a stochastic transition function denoting the probability distribution over states at sequence point t, given the state at sequence point t, and an initial probability distribution over states.
- **States.** The states in our MC model represent the relevant information that we have about the user. This information corresponds to previous choices made by users in the form of a set of ordered sequences of selection.
- **The Transition Function.** The transition function for our Markov chain describes the probability that a user whose k recent selections were x1, . . . , xk will select the item x' next. This is denoted trMC (<x1, . . . , xk> , <x2, . . . , xk, x'>). Initially, this transition function is unknown to us; and we would like to estimate it based on user data.

Improvements to the model

Skipping

- A form of skipping is described to improve the computation of the transition function. First, each state transition is initialized to the number of observed transitions in the data. A fractional count $1/2^{(j-(i+3))}$ is added to transitions <xi , xi+1, xi+2> to <xi+1, xi+2, xj>, for all i+3 < j ≤ n. This fractional count corresponds to a diminishing probability of skipping a large number of transactions in the sequence. Finally, the counts are normalized to obtain the transition probabilities.

$$tr_{MC}(s, s') = \frac{count(s, s')}{\sum_{s'} count(s, s')}$$

where count(s, s') is the (fractional) count associated with the transition from s to s'.

Improvements to the model

Finite Mixture Modeling

- A third enhancement is the use of finite mixture modeling. The mixture model is motivated by the fact that larger values of k lead to states that are more informative whereas smaller values of k lead to states on which we have more statistics.

- To balance these conflicting properties we mix k models, where the ith model looks at the last i transactions. Thus, for k = 3, we mix three models that predict the next transaction based on the last transaction, the last two transactions, and the last three transactions.

- For simplicity, $\pi 1 = ... = \pi k = 1/k$ was applied. Because the primary model is based on the k last items, the generation of the models for smaller values entail little computational overhead.

Modeling the MDP

- MDP is a four tuple < S, A, R , T > where S is a set of states, R is reward function, t is a state transition function.

- The actions of the MDP correspond to a recommendation of an item.
  - In our project we are assuming there is single recommendation only.
  - So when we recommend a game, let's say x the user can either accept the recommendation and go from state <x1,x2,x3> to <x2,x3,x> assuming we are only storing three most recent transactions in a state or the user can select a non recommended item.
  - The reward of the state <x1,x2,x3> can be the profit generated for the site from the sale of game x3 which is the last item in the transaction sequence.

- There are some assumptions
  - A recommendation increases the probability that a user will buy an game. This probability is proportional to the probability that the user will buy this item in the absence of recommendations
  - The probability that the user will buy an item which is not recommended is lower than the user will buy it in absence of recommendation but it is still proportional to it.

Solving the MDP

- We will use the policy iteration to solve the MDP. As it terminates after a handful of iterations. The computation of optimal policy would be extremely time consuming as each iteration requires the computation of expected rewards for each state.

- In the approximation, they do not compute a policy for a state that was not encountered in their training data because the probability of making a transition into an unencountered state is very low.

- They use the immediate reward of such states as an approximation of their long-term expected value for the purpose of computing the value of a state that appeared in our training data. Once we have a policy, they can use it to generate the recommendation for each state.

Evaluating the Model

- Predictions were evaluated as follows. For every user sequence t1 , t2 , …. , tn in the test set, the following test cases are generated: <t1>, <t1, t2>, <t1, t2, t3>, …., <t1, t2, t3, …., tn>.
- For each case various models are used to determine the probability distribution for ti given ti- k , ti- k +1 , …. , ti-1 and ordered the items by this distribution. Finally, we used the ti actually observed in conjunction with the recommendation list to compute a score for the list.
- **Evaluation metrics:**

❖ Recommendation score.

➢ A recommendation is deemed successful if the observed item ti is among the top m recommended items (m is varied in the experiments). The score RC is the percentage of cases in which the prediction is successful.

❖ Exponential decay score.

➢ This measure of accuracy is based on the position of the observed t i on the recommendation list. it is assumed that a user will see the mth recommendation with probability (m 1). (Note that α is the half-life parameter—the item in the list having probability of 0.5 being seen) $p(m) = 2^{-(m-1)/(\alpha-1)}$. The score is given by

$$100 \cdot \frac{\sum_{c \in C} p(m = pos(t_i|c))}{|C|}$$

where C is the set of all cases, c = ti- k ; ti- k+1 ; :::; ti-1 is a case, and pos(ti | c) is the position of the observed item ti in the list of recommended items for c.

Approach

Collection of data and identifying the states

- The first task was to collect relevant data for our project. We made sure that the data is cleaned, that is it does not have any missing or any arbitrary values. The data was such that it contains enough features required to give suitable predictions for our recommendation system
- The second task was to identify the states of the Markov Chain Model. It corresponds to previous choices made by the users in the form of ordered sequences of selection

Transition function and MDP

- Then we came up with a transition function which describe the probability which the user will select a particular item next.
- Then comes the major task of defining the MDP where we have to assign four tuples < S, A, R, T> where S is a set of states, R is reward function, t is a state transition function. Then comes the task of solving it. For this we will be mostly using Policy iteration for solving the MDP.

Evaluation of the model

- The final task of our project was to test and evaluate our model that we have built.
- We used some evaluation metrics to calculate how accurate our model is like recommendation score and Exponential Decay score.
- Comparison of the results was done using random generator model and then we tried to compare the results obtained by other non sequential algorithms such as regression which will give recommendation of a product.
- We plotted various graphs of the evaluation metrics using different models on the same test dataset. This will show the effectiveness of the MDP model as compared to other models

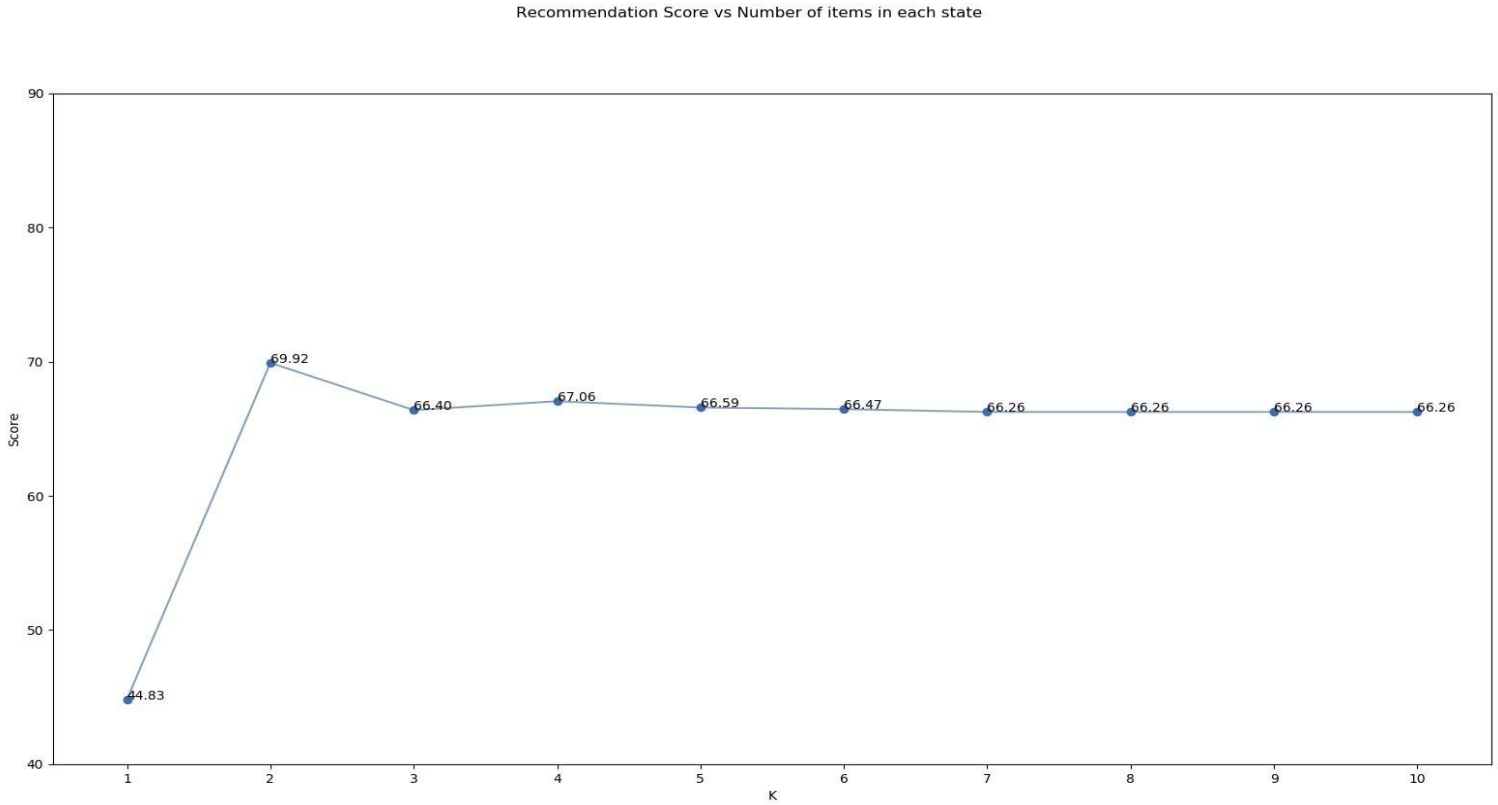
Results

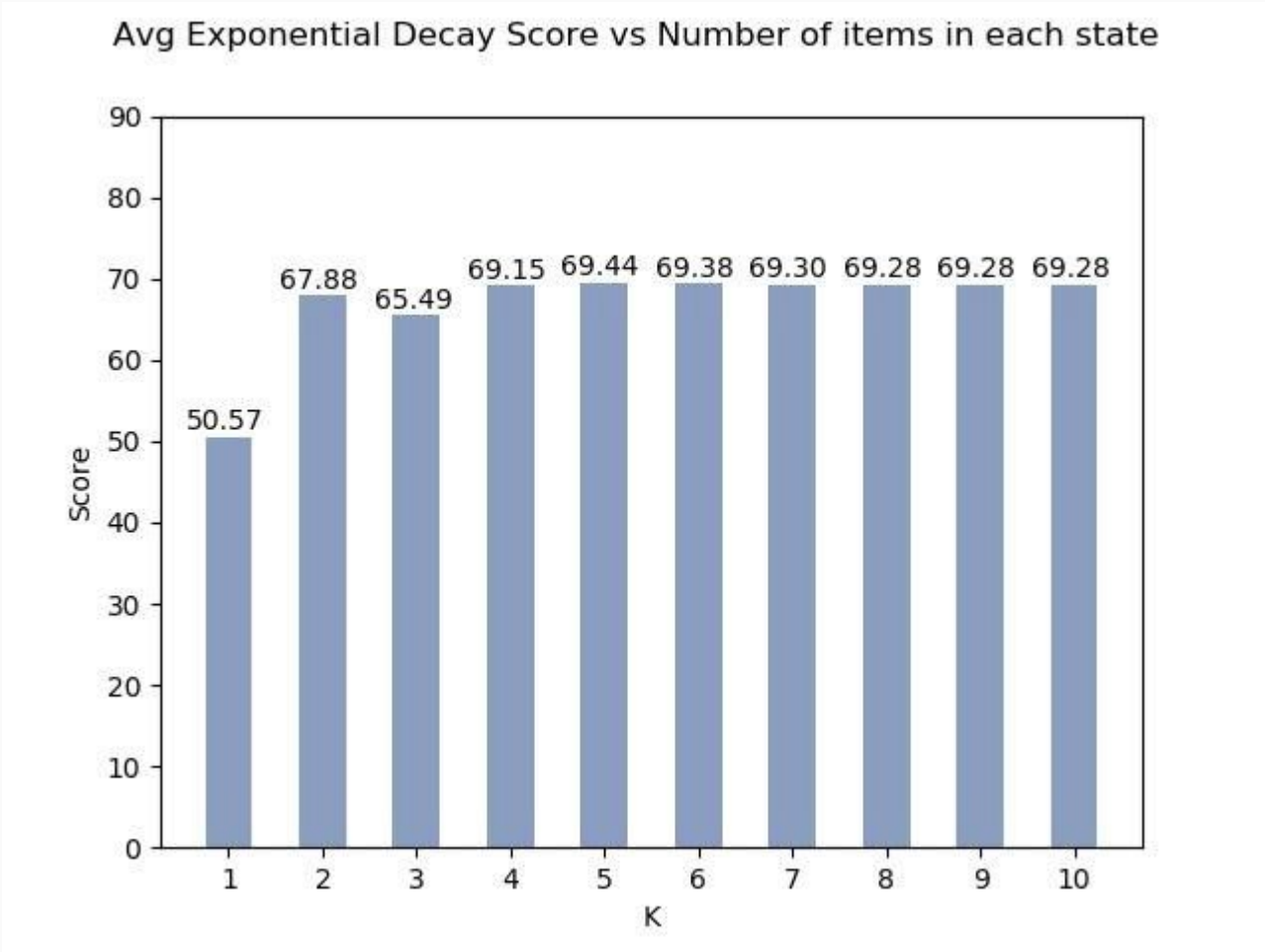
Scores vs the number of items in each state

We have plotted graphs comparing

- The recommendation score vs no. of items in each state.
- Average exponential decay score vs no. items in each state.

We observed that the recommendation score was highest when the no. of items in the state is 2.


Recommendation Score vs Number of items in each state
Score
K
90
80
70
60
50
40
1
2
3
4
5
6
7
8
9
10
44.83
69.92
66.40
67.06
66.59
66.47
66.26
66.26
66.26
66.26

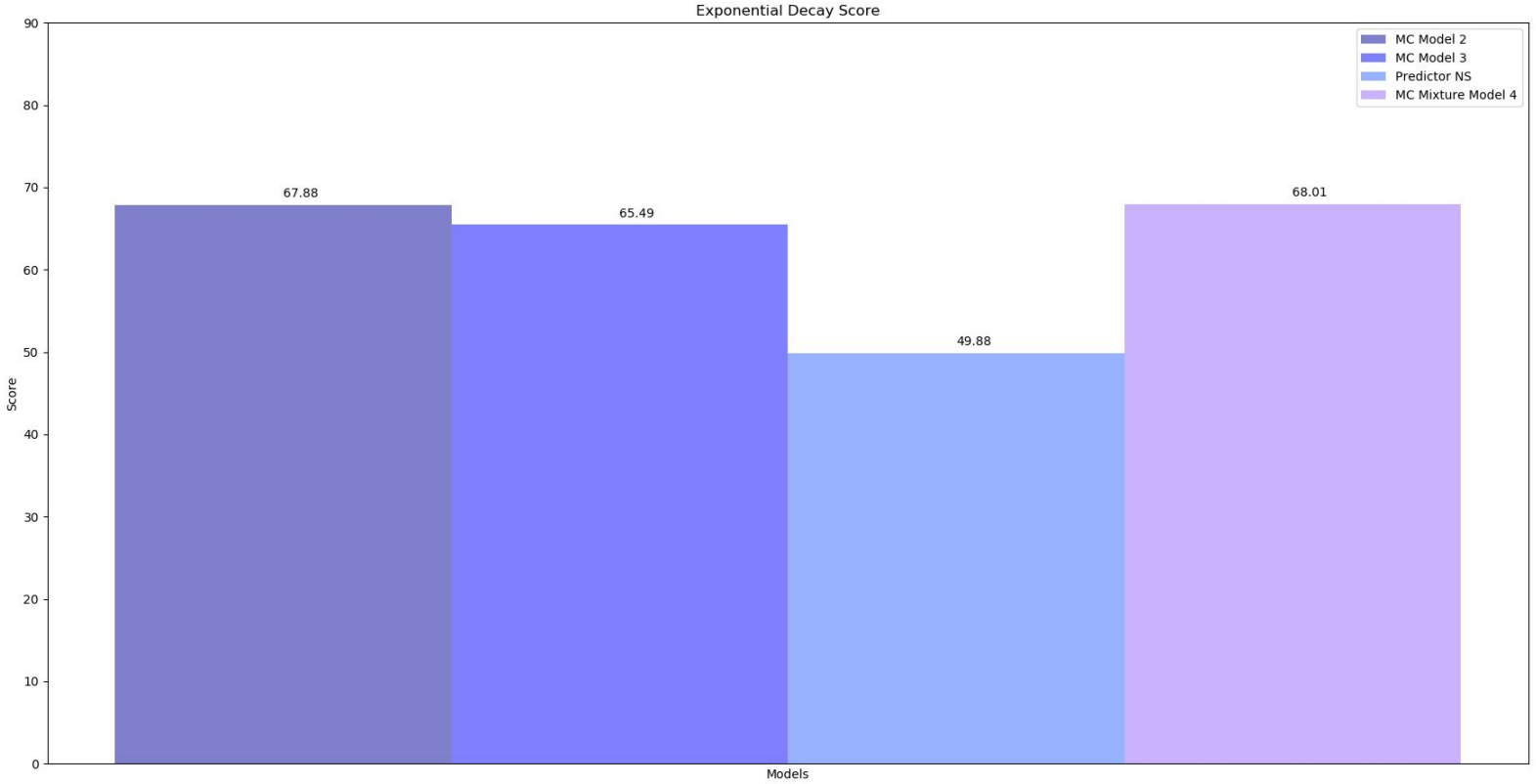

Avg Exponential Decay Score vs Number of items in each state
Score
K
0
10
20
30
40
50
60
70
80
90
1
2
3
4
5
6
7
8
9
10
50.57
67.88
65.49
69.15
69.44
69.38
69.30
69.28
69.28
69.28

Comparison with other models

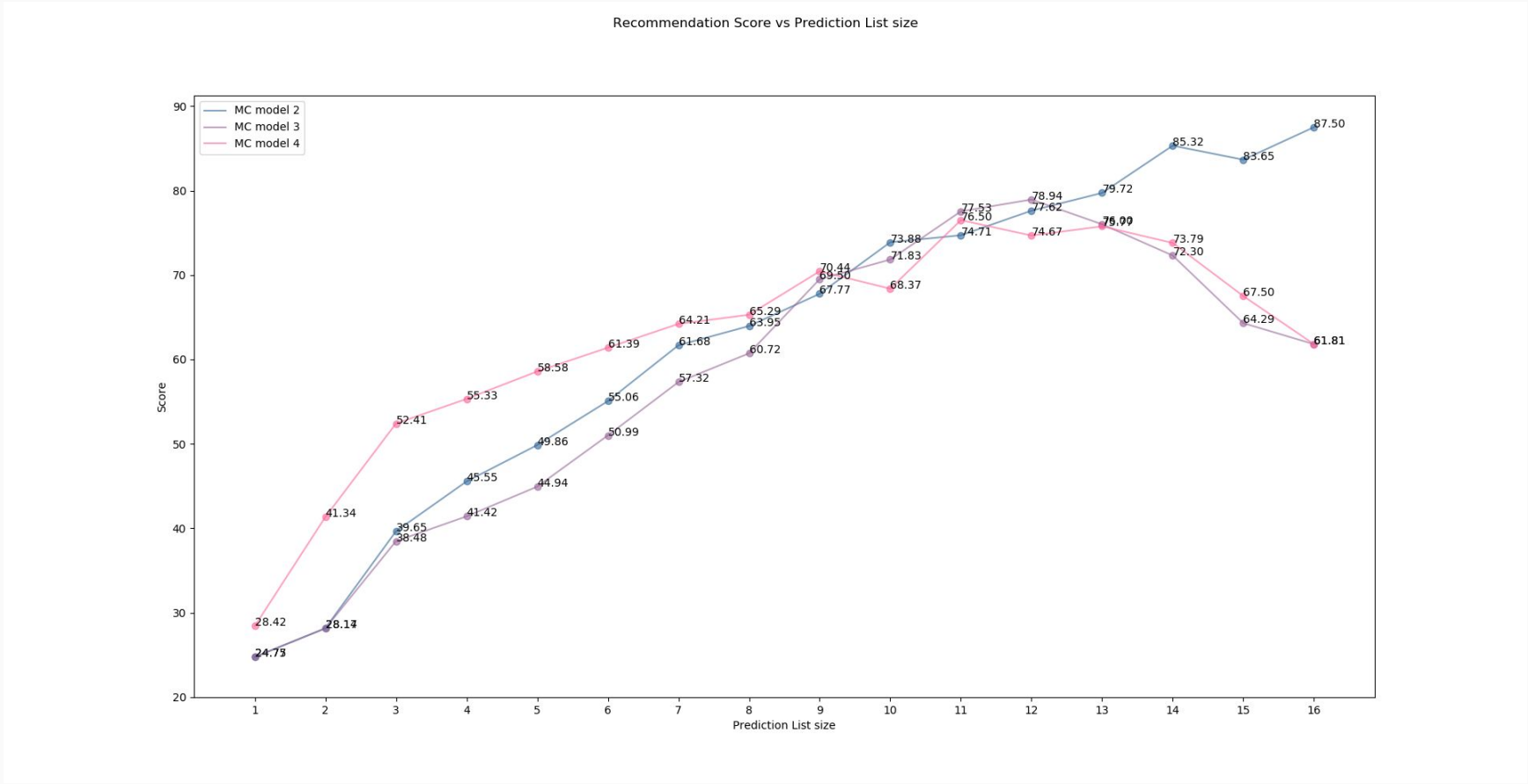
Comparison with random game recommender

- A random game recommender selects randomly a product out of the 20 items. So probability that it recommended the product will be in his top 10 list is 0.5 which means the recommendation score and decay score will be around 50%.
- Our model got a recommendation score of around 70% which means the user’s most relevant product is in the top 10 around 70% of the time.

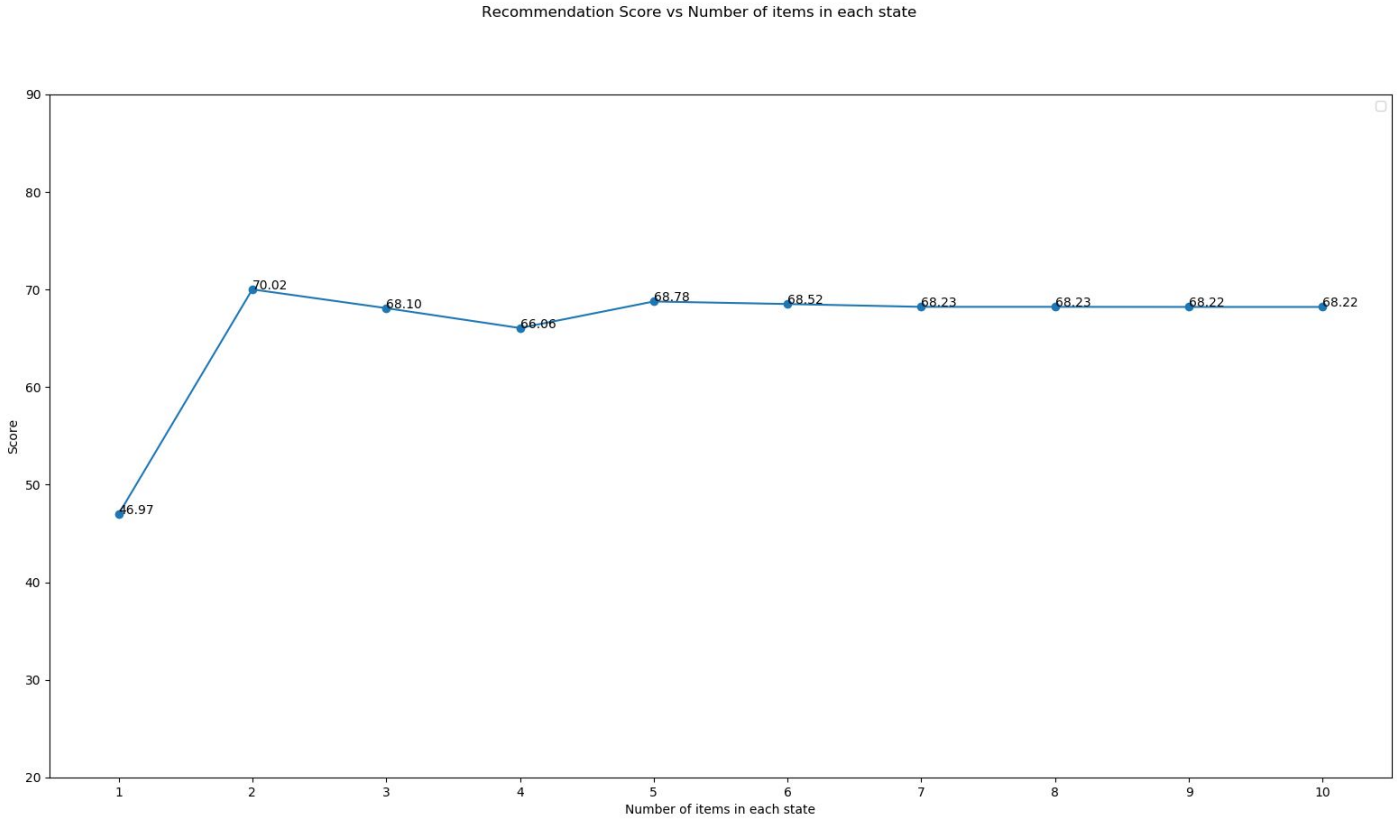
Decay score comparison with various models



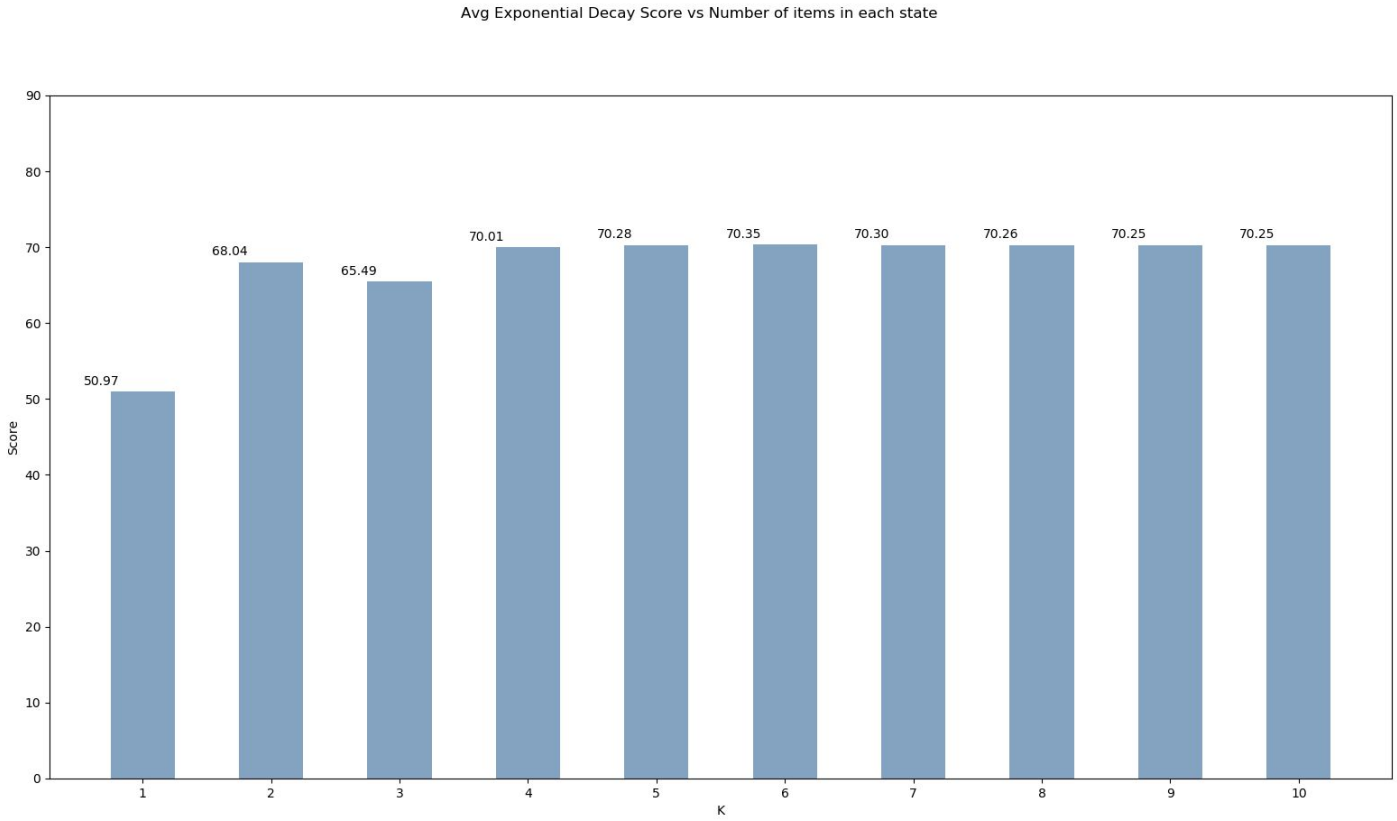
Recommendation score comparison with various models



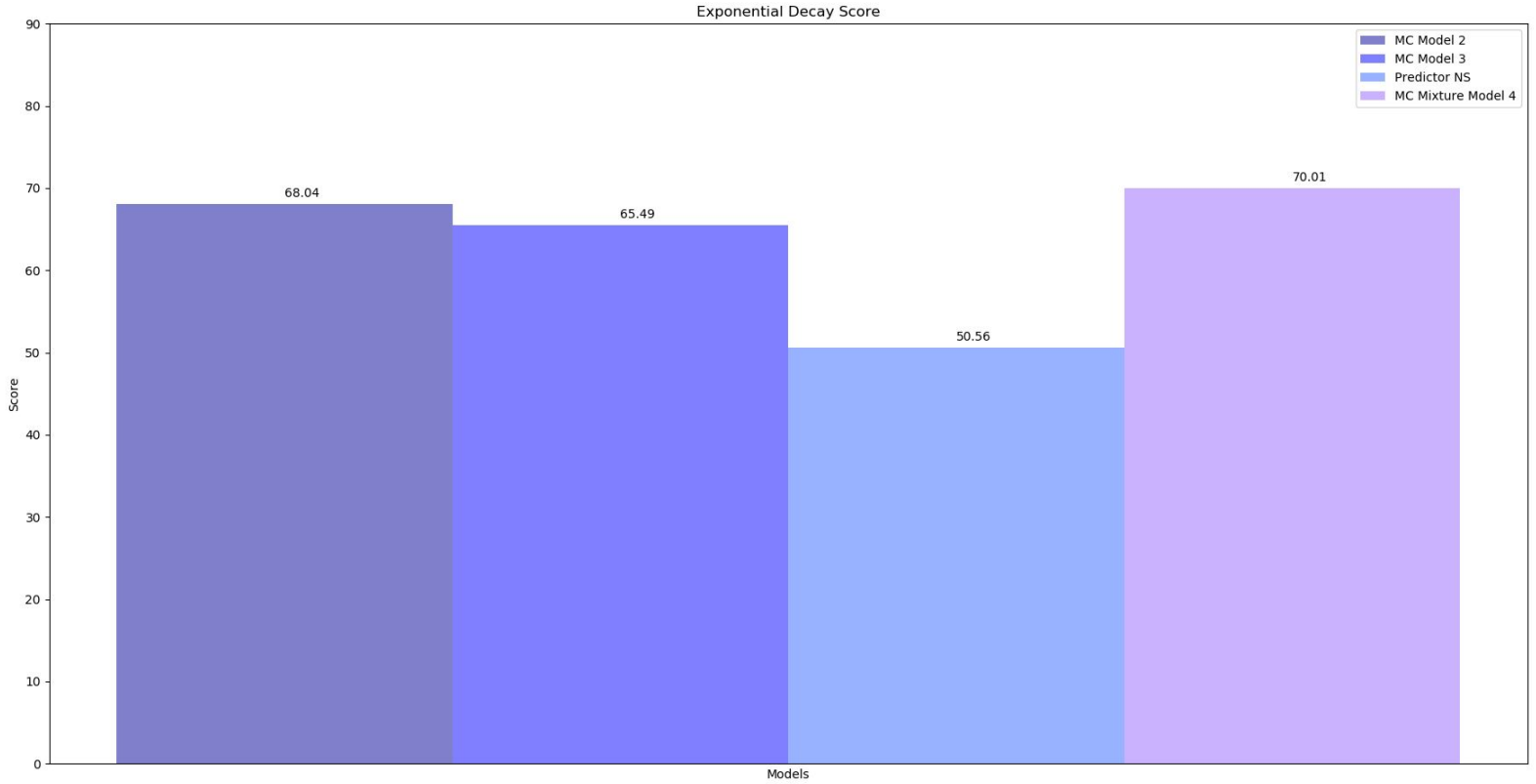
On a larger data set (almost 100 times larger)



On a larger data set (almost 100 times larger)



Decay score comparison with various models
On a larger data set (almost 100 times larger)



Demo

Future work

Exploring Composite Items

- We plan to follow the paper: “**Constructing and Exploring Composite items**”
- We understand that the paper is about an item being strengthened in value by using satellite items which are closer or compatible with it while also allowing multiple items to be processed at once.
- Rather than having an action involving a single item we can have it as a composite set of items.
- Another approach could be to use composite sets as a way to prune the search space when performing learning algorithms as we can check only those actions which are a part of the composite sets of all the items in the state as they can be assumed to be the only relevant ones.

Exploring Composite Items

- Use the summarization algorithm mentioned in the paper to prune out any redundant states
- Explore visual effect optimization mentioned in the paper to only keep states that are meaningful by nature using a predefined ordering. This also helps prune out the large state space by taking only those that make tangible sense.

Deep RL and Collaborative Filtering

- We have started looking into the collaborative filtering approach as mentioned by Prof. PK Reddy.
- We are also planning to follow the paper: “**Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling**” by Feng Liu et al
- Deep RL would be a huge improvement to our model as it greatly improves the scalability and also it produces an easy framework to model a complex problem. To find a good policy we could use valued-based methods like Q-learning to measure how good an action is in a particular state or policy-based methods to directly find out what actions to take under different states without knowing how good the actions are.

Thank You