Sequential Recommendation Systems Using LLM with Affordance-Based Deep Q-Learning

Lynn Cherif
Lynn.Cherif@mail.mcgill.ca

Edwin Meriaux
Edwin.Meriaux@mail.mcgill.ca

Vraj Patel
Vraj.Patel@mail.mcgill.ca

Abstract

In this paper, we explore the impact of using affordance-based deep reinforcement Q-learning for sequential item recommendation on the Amazon Beauty dataset using natural language representations from a pre-trained FastText and LLama model. Although computational challenges were met using Llama embeddings of the item names, results using the FastText embeddings and affordance-based deep Q-learning show promise to match the state-of-the art on sequential recommendation on this dataset using large language models. As well, the affordance-based model exhibits significantly faster and more stable learning than its vanilla deep Q-learning counterpart.

1 Introduction

Recommender systems are commonly to recommend songs, movies, and products to be purchased. One of the largest difficulties in such models is understand the possible items to be recommend and the users' history of previous items. This is why LLMs have been used to generalize the previous actions into the current state and the possible recommenations into the next state (Zhang et al., 2021)(Geng et al., 2023)(Liu et al., 2023). This dependency on LLMs come from the large number of parameters and data that LLMs such as LLama or ChatGPT are trained on, it is hypothesized that LLMs have "world knowledge" that can be used to make recommendations.

An important recommendation task is sequential recommendation: given a user's past item interaction history, what item should be recommended to them next? Sequential recommendation can be naturally formulated as a Markov Decision Process (MDP). Therefore, methods such as reinforcement learning (RL) are naturally suited to such challenges and have shown promise on sequential recommendation (Chandak et al., 2020). Thus far, LLMs have not been able to demonstrate strong generalization abilities without fine-tuning or extensive prompt-tuning (Liu et al., 2023)(Wang and Lim, 2023).
To this end, this project seeks to explore the possibility of matching or improving the data-efficiency and generalization abilities of natural-language-based sequential recommendation systems by combining language representations from a language model or an LLM, and RL techniques. More specifically, sequential recommendation is considered on the Amazon Beauty dataset, where results by several supervised fine-tuned LLMs such as Llama, Alpaca, and ChatGPT are reported (Liu et al., 2023). As such, in this project, the item names are embedded using either a pre-trained FastText model (Bojanowski et al., 2017) on the 1M Wiki-news dataset, or extracted embeddings from either the first or last hidden layer preceding the projection layer in a pre-trained Llama 2 model (Touvron et al., 2023). The state space of the RL environment consists of the concatenated sequences of n-gram embedded items corresponding to users' item-interaction histories (Chandak et al., 2020) (Zhang et al., 2021). The action space consists of the set of items available in the Amazon Beauty dataset. Two RL models are explored: vanilla Deep Q-learning (DQN), and an affordance-based DQN (ADQN) that leverages product category meta-data as prior knowledge to aid the training of the RL agent and potentially its generalization capacities. The results for sequential recommendation obtained with the DQN and ADQN on the Amazon Beauty dataset are compared to those reported by the supervised-fine-tuned ChatGPT, LLama 2, and Alpaca reported in (Liu et al., 2023). The overall approach is shown in Figure 1. [1]

[1]code for this project can be found here: https://drive.google.com/drive/folders/1n1uGJdSjCpRg9tdfB9ci276Ct0FQI8a2?usp=sharing

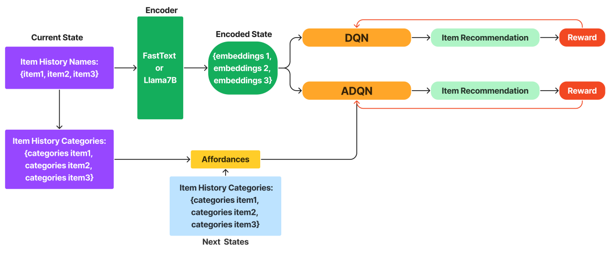
Current State
Item History Names: {item1, item2, item3}
Encoder
FastText or Llama7B
Encoded State
{embeddings 1, embeddings 2, embeddings 3}
DQN
ADQN
Item Recommendation
Item Recommendation
Reward
Reward
Item History Categories: {categories item1, categories item2, categories item3}
Affordances
Item History Categories: {categories item1, categories item2, categories item3}
Next States

Figure 1: Recommender system workflow

2 Related work

Due to LLMs' promise to adequately represent the state of a user history with natural language alone and give recommendations accordingly, three general trends in LLM-based recommender systems have been observed: recommender system specific LLM foundation models (Geng et al., 2023), supervised fine-tuning LLMs (Zhang et al., 2021) (Liu et al., 2023), and prompt-engineering (Wang and Lim, 2023). Additionally, several studies have been conducted to evaluate deep RL techniques on sequential recommendation tasks (Xian et al., 2019) (Chen et al., 2023), as well as combining natural language representations with deep RL (Lin et al., 2023). However, to the best of our knowledge, there have not been any studies that leverage the notion of affordances (Gibson, 1977) (Khetarpal et al., 2020) in a deep RL model to attempt to improve the data efficiency and generalization capacities for natural-language-based sequential recommendation systems.

3 Methodology

In this project, n-gram sequences of user-item interaction histories from the Amazon Beauty dataset are embedded and used as input states to the RL algorithm for item recommendation. three types of item name embeddings were extracted: one using a pre-trained FastText model, and two using pre-trained Llama 2 model. As for the RL algorithms, a vanilla Deep Q-Learning (DQN) model and an affordance-based Deep Q-Learning (ADQN) that leverages product category meta-data as prior knowledge, were implemented. The results are reported in terms of hit rate (HR), normalized discounted cumulative gain (NDCG), and cumulative reward over episodes.

3.1 Dataset

The parsed Amazon Beauty dataset parsed into a sequential recommendation task is used from (Liu et al., 2023). This data comprises of variable lengths user-item interaction histories, and are pre-split into training, validation, and testing sets of approximately 22k item interaction histories, respectively. In order to adapt the data to a RL setting, the user-item interaction histories were parsed into n-grams, specifically tri-grams, where the last item is considered the target item to recommend. Tri-grams were selected based on a recency relevance hypothesis, and to manage computational resources. Every word in every item name is embedded using a respective embedding type, and then concatenated to form a state. After parsing the data into tri-grams, the training state space consisted of 43k states, and the testing state space of 133k. The action space consists of all the 230k beauty products. Every product has associated metadata, which includes its relevant categories (e.g., beauty, skin care, body, make up, nail polish). [2]

3.2 FastText Embeddings

Embeddings using a pre-trained FastText model on the 1M Wiki-news dataset were considered as an initial simple baseline in this project. Unlike, pre-trained word2vec models (Mikolov et al., 2013), FastText is able to generate embeddings for out-of-vocabulary words. Indeed, FastText is based on the skipgram model, where each word is represented as a bag of character n-grams. A vector representation is associated to each character n-gram, and words are represented as the sum of these representations (Bojanowski et al., 2017). FastText is particularly fast and light-weight, allowing ease of computational resource management. To respect the model's pre-training requirements, every word is embedded into a vector of size 300, and every item name consisted of a fixed 256 entries to accomodate for varying item name lengths. An item name consisting of less than 256 words are padded with zero embedding vectors. The item names were not preprocessed by removing stop-words, lemmatizing, or stemming, and every word is considered a token. This is purposely done to have an equivalent comparison to the Llama 2 embeddings, where raw natural text is input. Additionally, the item names mainly consisted of nouns, and rarely observed stop-words.

[2]Amazon Beauty dataset `https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/#sample-metadata`

3.3 Llama 2 Embeddings

Llama is a pre-trained LLM with 7B parameters and a context length of 4096(Touvron et al., 2023). It uses input text to output chat responses to the user. To understand such inputs it needs to embed the received text. There are two types of embeds: one from the first hidden layer, and a second from the last hidden layer preceding the projection layer. Llama was modified for the sake of this project to simply just output its embedded text. The issue with this is the size of the data. There are about 250k products in our train and test sets combined. This required over 150 GB of embeded stored data. This took very long to process (it was all processed) but in the time constraints it was unfeasible to use. FastText was faster and simpler to use and so it was used even though considerable time was spent working on Llama and embedding all the needed text.

3.4 Reinforcement Learning

RL is a subfield of machine learning that focuses on training an artifically intelligent agent by interacting with its environment to accumulate rewards. DQN and ADQN were trained in an offline manner (i.e., using the logged state transitions from the Amazon Beauty dataset). The model is limited to the start states that exist in the dataset. A reward of 1 is attributed upon choosing an action that transitions the agent into a state that exists in the logged data, and a reward of 0 otherwise. To aid training and avoid continuously transitioning into nonexistent states, the episode consists of one timestep.

3.4.1 DQN

DQN combines Q-Learning and a deep neural networks (DNN). Q-learning aims to find an optimal state-action value function Q^* that can adequately estimate the return of an action in a given state in order to construct a policy π that maximizes rewards (Equation 1).

$$\pi^*(s) = \text{argmax}_a \, Q^*(s,a) \tag{1}$$

The DNN is used to represent the combinatorial state-space and approximate the optimal policy function π^* through the Q function by optimizing the temporal difference error δ (Equation 3) using the Huber Loss (Equation 4), given the training update rule 2 obeying the Bellman equation. s denotes a the current state, s' the next state, a denotes the action, r denotes the reward, π denotes the policy function, γ is a discount factor, and B denotes a batch sampled from the replay buffer. Since our episodes are defined as one-step, $\gamma = 0$.

$$Q^\pi(s,a) = r + \gamma Q^\pi(s', \pi(s')) \tag{2}$$

$$\delta = Q(s,a) - (r + \gamma max_a Q(s',a)) \tag{3}$$

$$\mathcal{L}(\delta) = \begin{cases} \frac{1}{2}\delta & \text{for } |\delta| \leq 1 \\ |\delta| - \frac{1}{2} & \text{otherwise} \end{cases} \quad \mathcal{L} = \frac{1}{|B|} \sum_{(s,a,s',r)\in B} \mathcal{L}(\delta) \tag{4}$$

A replay buffer of size N is a queue that stores previously seen transitions to aid the model's learning. In this project, $N = 2000$. The DNN used consists of two hidden layers of size 300 using the `ReLU` activation function, and linear output layer of size of the action space. A learning rate of 10^{-3} is used, along with an Adam optimizer set to its default settings in PyTorch. To allow better learning, the ϵ-greedy method is used trade between the agent's exploration and exploitation behavior. If a sampled number α from a uniform distribution in $[0, 1)$ is $\alpha \geq \epsilon_{threshold}$, a greedy action is taken (i.e., the action that the policy network predicts to have the highest value), otherwise, a random action from the action space is explored. The epsilon threshold decays over the number of episodes, and follows the schedule in Equation 5, where $\epsilon_{start} = 0.9$, $\epsilon_{end} = 0.05$, and $\epsilon_{decay} = 1000$. All hyperparameters were chosen by trial and error (e.g., learning rate), due to computational limitations (e.g., replay buffer size), and/or based on recommendations found online (e.g., epsilon decay). [3]

$$\epsilon_{threshold} = \epsilon_{end} + (\epsilon_{start} - \epsilon_{end}) \exp(-\frac{episodes}{\epsilon_{decay}}) \tag{5}$$

3.4.2 ADQN

Affordances (Gibson, 1977) refer to the feasible actions in a given situation (e.g., water affords swimming for a fish). The notion of affordances has been formalized in RL (Khetarpal et al., 2020) and proven to improve planning efficiency and accuracy by only considering affordable actions in the environment at a given a state. Integrating the notion of affordances to DQN is thus hypothesized to aid the model's training efficiency and generalization capacity. Given a state, affordable actions are

[3] https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

heuristically computed based on the normalized category overlap NCO between the items contained in the current state and the next potential state if the considered action were to be taken. The NCO is defined in Equation 6, where s corrsponds to the current state, and s' to the next state. An action is considered affordable if $NCO \geq 0.5$. The set of computed affordances is then used to mask the the values of the unaffordable actions in the DQN before selecting the greedy action to take. Similarly, random actions are only sampled from the set of affordable actions at a given state. The same DQN architecture and hyperparameters are used otherwise.

$$NCO = \frac{|categories(s) \ \cup \ categories(s')|}{|categories(s)| + |categories(s')|} \tag{6}$$

3.5 Evaluation Metrics

To benchmark and compare our results to those in (Liu et al., 2023), two metrics that are mainly used in evaluating the quality of recommender systems were considered: the top-K hit rate (HR@K) and the top-K normalized discounted cumulative gain (NDCG@k). Additionally, the cumulative reward over episodes is reported.

3.5.1 HR@K

HR@K evaluates whether an item is in the top-K recommended items(Liu et al., 2023). In our case, HR@K is based on whether the correct item is in the top-K predicted Q values by the policy network.

3.5.2 NDCG@K

NDCG@K is similar to HR@K but weights the position of the answer within the top K. This evaluates both the model's ability to predict the recommendation and how highly it was ranked (Wang et al., 2013).

3.6 Experimental Setup

3.6.1 Train Settings

Given the significantly large action space of 230k items, during training, RL models can struggle to obtain informative learning signals if incorrect transitions are consistently shown (e.g., due to consistently taking random actions during exploration, and a network without knowledge yet). Thus, warm-starting by showing the model a number of correct transition tuples (s, a, s', r), i.e., transitions that can grant a reward of 1, has the potential to aid in this regard. As such, three training settings are explored:

- Case 1: 0 warm-start episodes, 10k ϵ-greedy episodes
- Case 2: 1k warm-start episodes, 9k ϵ-greedy episodes
- Case 3: 10k warm-start episodes, 10k ϵ-greedy episodes

Although validation on the development should be done to determine the best model, we were unable to perform validation experiments with our computational resources. This is mainly caused by the affordances computations during ADQN, where affordances are currently computed in $O(N)$. Ideally, an affordance classifier should be pre-trained and used for inference during the training and validation of the RL models. All sources of randomness were seeded with 0.

3.7 Test Settings

For every training case, the fully trained model is evaluated on the Amazon Beauty test dataset. However, as the tri-gram parsed test dataset resulted in over 133k states, it was not computationally tractable to perform inference on all test states with the ADQN. Hence, 20k test states were randomly sampled with a seed of 0 and evaluated instead.

4 Results and Discussion

4.1 Train Results

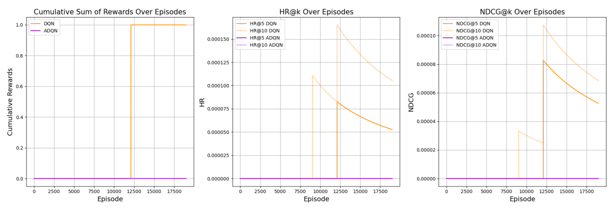

Figure 2: Train Data Case 1

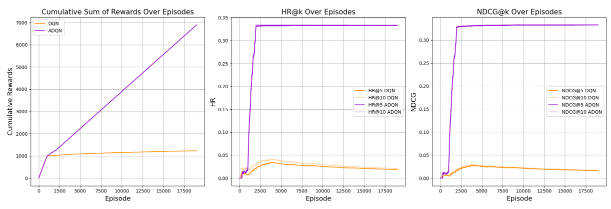

Figure 3: Train Data Case 2

As seen in Figures 2, 3, and 4, the ADQN trained on Case 2 outperformed all the other methods

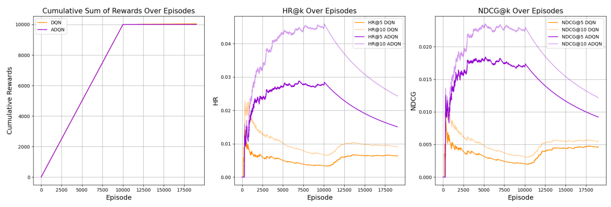

Figure 4: Train Data Case 3

overall in terms of HR@5, HR@10, NDCG@5, NDCG@10 and cumulative reward over episodes. It is clear that Case 2 ADQN learn significantly faster and is more stable than Case 2 DQN, as well as the other methods. It is also more accurate, as can be seen with the overlapping HR@K and NDCG@K curves. Both ADQN and DQN learned best with a combination of warm-starts, random exploration, and greedy actions. Indeed, the other two training cases do not have this balance of elements. Case 1 does not have any warm-start episodes, which limits the relevance of the learning signal. As for Case 3, the model is likely overfitting the warm-start episodes as it is not performing any random actions due to already reaching ϵ_{end} by the ten-thousandth episode. It could thus be hypothesized that injecting exploration during warm-starts can help regularize the RL model. In addition, the poor performance and performance drop in cases 1 and 3, respectively, could also be due to a mis-learned representation of the environment by the network. This could be caused by the FastText embeddings, as it is likely that many of the words in the item names are out-of-vocabulary with respect to the Wiki news corpus. Another possibility could be the relatively weak capacity of the neural network, having only two hidden layers. It is also worth noting that the HR@10 and NDCG@10 being greater than HR@5 and NDCG@5 is expected as one is a subset of the other, and that the HR@K and NDCG@K would be similar.

4.2 Test Results

The best results from the test set mirror the train set, as noted in Table 1 where the best model is the Case 2 ADQN. However, the model does seem to overfit as the values are an order of magnitude lower than the train results. In future work, this could be mitigated by using the development set as previously described. Nevertheless, the results from the Case 2 ADQN approximately match those reported for the Llama-7B with supervised fine-tuning, and are about two-thirds as good as the current state-of-the-art reported on the ChatGLM-6B with supervised fine-tuning, as seen in Table 2. Given that our models were trained significantly less data than the supervised-fine-tuned LLMs presented in (Liu et al., 2023), it is expected that Case 2 ADQN could be able to match the state-of-the-art given more training episodes, and perhaps a larger network and the LLama 2 embeddings (type 2 being of particular interest). Indeed, the training dataset contains over 40k states with 250k possible actions, but the training was only done over 20k states.

	Case 1		Case 2		Case 3	
	DQN	ADQN	DQN	ADQN	DQN	ADQN
HR5	0	0.0011	0.0064	0.0198	0.0025	0.0169
HR10	0	0.0013	0.0068	0.0298	0.0028	0.0308
NDCG5	0	0.0005	0.0058	0.0138	0.0025	0.0106
NDCG5	0	0.0006	0.0060	0.0171	0.0025	0.0150

Table 1: Test results of DQN And ADQN on 20k randomly sampled test states

	ChatGLM-6B w/ SFT	LLAMA-7B w/ SFT	Alpaca w/ SFT
HR5	0.0403	0.0277	0.0325
HR10	0.0455	0.0280	0.0325
NDCG5	0.0349	0.0280	0.0330
NDCG5	0.0366	0.0277	0.0327

Table 2: State-of-the-art models using supervised fine-tuning (SFT) and top LLM models on all test data

5 Conclusion

This paper studies whether embedded text of users' item interaction histories can be used to represent states in deep reinforcement learning models to accurately recommend next products to users. We explore an affordance-based and vanilla DQN. We show that with limited training, our affordance-based DQN model is promising with a limited amount of data and training, and could compete with current state of the art systems on sequential recommendation using supervised fine-tuned LLMs given more training episodes. This shows that the combination of natural language representations and RL is promising to improve the data-efficiency and generalization for sequential recommendation tasks.

6 Statement of Contribution

Lynn Cherif: Worked on the finding and processing the dataset, Fasttext embedding, buildling DQN model, building affordance model, train/test evaluation and report; **Edwin Meriaux:** Worked on the finding and processing the dataset, LLAMA embeding, buildling DQN model, train/test evaluation and report; **Vraj Patel:** Worked on the data preprocessing model and report.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information.

Yash Chandak, Georgios Theocharous, Chris Nota, and Philip S. Thomas. 2020. Lifelong learning with a changing action set.

Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. 2023. Deep reinforcement learning in recommender systems: A survey and new perspectives. *Knowledge-Based Systems*, 264:110335.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2023. Recommendation as language processing (rlp): A unified pretrain, personalized prompt predict paradigm (p5).

James J. Gibson. 1977. The theory of affordances. In John Bransford Robert E Shaw, editor, *Perceiving, acting, and knowing: toward an ecological psychology*, pages pp.67–82. Hillsdale, N.J. : Lawrence Erlbaum Associates.

Khimya Khetarpal, Zafarali Ahmed, Gheorghe Comanici, David Abel, and Doina Precup. 2020. What can i do here? a theory of affordances in reinforcement learning.

Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. 2023. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*.

Junling Liu, Chao Liu, Peilin Zhou, Qichen Ye, Dading Chong, Kang Zhou, Yueqi Xie, Yuwei Cao, Shoujin Wang, Chenyu You, and Philip S. Yu. 2023. Llmrec: Benchmarking large language models on recommendation task.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models.

Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A theoretical analysis of ndcg type ranking measures. In *Conference on learning theory*, pages 25–54. PMLR.

Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 285–294.

Yuhui Zhang, Hao Ding, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. 2021. Language models as recommender systems: Evaluations and limitations. In *NeurIPS 2021 Workshop on I (Still) Can't Believe It's Not Better*.