Learning a strategy for preference elicitation in conversational recommender systems

Conference or Workshop Item

Accepted Version

Makarova, A., Shahzad, M. ORCID: https://orcid.org/0009-0002-9394-343X, Hong, X. ORCID: https://orcid.org/0000-0002-6832-2298 and Lester, M. ORCID: https://orcid.org/0000-0002-2323-1771 (2024) Learning a strategy for preference elicitation in conversational recommender systems. In: International Joint Conference on Neural Networks (IJCNN), 30 Jun - 5 Jul 2024, Yokohama, Japan. doi: 10.1109/ijcnn60899.2024.10650365 Available at https://centaur.reading.ac.uk/116044/

To link to this article DOI: http://dx.doi.org/10.1109/ijcnn60899.2024.10650365

Learning a Strategy for Preference Elicitation in Conversational Recommender Systems

Aleksandra Makarova, Muhammad Shahzad, Xia Hong, Martin Lester
Department of Computer Science
University of Reading, UK

***Abstract*—This paper delves into the information elicitation aspect of Conversational Recommender Systems (CRS), presenting an innovative method of selecting chatbot questions that result in the highest information gain when reconstructing the preference profile of a user, which allows one to achieve high-quality recommendations after a small number of conversational interactions. The proposed system comprises a Recommendation Module and a Preference Elicitation Module. The Recommendation Module leverages a Long Short-Term Memory (LSTM) network with an Attention mechanism and is optimised to reconstruct the preference profiles of new users based on limited information gathered through dialogue. The Preference Elicitation Module is trained using a reinforcement learning technique known as bot-play, where the Questioner Bot proactively prompts the Answerer Bot to provide item and attribute ratings, leveraging the reduction in the Recommendation model's loss as a reward signal. This enables the model to learn an optimal questioning strategy, thereby maximising the accuracy of the representation of the user profile and the relevance of recommendations. The experimental results demonstrate the ability of the Recommendation component to learn item-attribute mappings, enabling the Questioner Bot to make accurate rating predictions with only a limited number of answered questions. Moreover, the trained Preference Elicitation policy model consistently outperforms the baseline model across both synthetic and real-world datasets, showcasing its ability to minimise the number of conversational turns required to achieve accurate recommendations.**

***Index Terms*—conversational recommender systems, neural recommender systems, bot-play, information elicitation**

I. Introduction

Recommender Systems (RS) help users to choose among a large number of potential options by presenting a narrow selection of those most likely to be relevant. In recent decades, globalisation has enabled retailers to offer a vast range of products to their customers. An abundance of options, particularly combined with high choice complexity (e.g. the number of distinguishing attributes is large), may lead to user frustration and poor sales [1]. Traditionally, this problem has been mitigated by human advisors. However, the widely recognised 'death of the high street' trend, in which traditional shops close and consumers shop online, creates the need for online recommendation systems that are commercially viable, scalable and automated.

In the last two decades this need has been addressed by traditional RS, a family of algorithms designed to personalise user experience and increasingly adopted by e-commerce businesses [2, 3]. RS present users with the most relevant options by leveraging the user profiles and representations they maintain. User representations refer to the way the system models or understands individual users based on their preferences, behaviors, and interactions with the system. These representations typically include information about a user's past choices, likes, dislikes, and any other relevant data that helps the system make recommendations tailored to that specific user. User representations are essentially profiles or models that capture a user's characteristics and preferences, allowing the RS to offer personalised recommendations. However, traditional RS cannot replicate the experience of talking to a human advisor, as they lack interactivity, which leads to such problems as cold start, where existing data is not sufficient to compute a recommendation, and taste drift, where a user's requirements change over time, and recommendations computed based on historical data become irrelevant.

Intelligent conversational systems have high potential in commercial settings. A combination of RS with intelligent conversational algorithms has resulted in the emergence of the Conversational Recommender Systems (CRS), a subset of RS that are interactive. CRS build upon traditional offline algorithms by introducing interactivity, which results in a capability for active human-in-the-loop learning. CRS allow users to communicate their preferences, then utilise this information to update user representations in real time. CRS can identify gaps in user representations and proactively request information from users to compute more appropriate recommendations. This closely mimics a conversation between two humans, particularly where natural language is used as an interface. This property makes CRS a perfect use case for applying a range of reinforcement learning (RL) techniques for a number of reasons.

Firstly, CRS involve continuous interaction with an external environment (a user), which ensures that rewards can be easily operationalised and fed back to the CRS algorithm in real time. This frees reinforcement learning driven CRS from rigid assumptions that characterise systems driven by handcrafted rules [4] or trained using supervised learning only [5]. For example, various collaborative filtering and predictive algorithms assume that a certain dialogue flow is optimal for all users, or that recommendations should mimic behaviour that users exhibit naturally. Instead, the system behaviour can be tailored to directly optimise performance metrics meaningful in real world settings, such as dialogue success rate [6, 7], including business performance indicators like conversion and user satisfaction [4].

Secondly, reinforcement learning allows CRS to overcome such critical problems as cold start or user taste drift [6] by leveraging active exploration techniques, converging to an optimal set of recommendations over the course of the conversation with a user. For instance, when a user provides negative feedback regarding an item or an attribute, the CRS can update its rating of all related items and attributes, avoiding further exploration of the same space [7]. Even in complete absence of any a priori information, reinforcement learning systems can efficiently elicit user preferences and produce relevant recommendations in just a few conversational turns [8].

Thirdly, reinforcement learning techniques can be totally data driven, which makes them easily transferable to new application scenarios. In contrast, non-adaptive systems can exhibit unreliable and wildly different performance on new datasets [7].

Previous work. Lately, a novel RL approach for training CRS, known as bot-play, has been gaining attention. In bot-play both parties of the dialogue, Seeker and Recommender, are models capable of improving over the course of cooperative training [9, 10]. Bot-play is an RL protocol where two (or more) models are trained concurrently, with outputs of one model serving as inputs to the other model and vice versa. In the case of CRS, bot-play is cooperative (in contrast to adversarial learning), so the same reward is commonly shared between the two models. There are several attempts to apply bot-play to CRS domain described in the literature [9, 10].

Li et al. [10] introduces a model based on conversational critiquing approach, implemented using an interactive user interface. The proposed model consists of three main components: a recommender model for item ranking, a justification module for predicting rationales (mined form user reviews) behind recommendations, and an interactive critiquing function that allows users to modify rationales and influence future recommendations. This enables users to refine their preferences, creating a dynamic and adaptable recommendation system. The model is trained through self-supervised bot-play, where the authors employ a rule-based recommendation seeker and a pre-trained expert model that includes recommendation and justification modules. Expert and seeker models are allowed to converse, with the goal of recommending the goal item, which enables expert model fine-tuning.

Kang et al. [9] describes a natural language-based CRS pre-trained on the GoRecDial dataset, which consists of dialogues produced by a goal-driven game data collection protocol, where Amazon Mechanical Turk workers earned points for making appropriate movie recommendations. The algorithm is trained to rank a set of five items rather than retrieve a recommendation from a large set of options. The authors recognise that this is a notable limitation of the algorithm, as it does not scale and cannot be used in a real world scenario. Additionally, the algorithm does not have a separate recommender module, and learns target movies directly from conversational data, which has a limited number of movie mentions. In a real-world system, having a separate pre-trained recommender component is highly beneficial (e.g., [5]).

None of the above papers focus on the capability of CRS to proactively glean preference information from users. In contrast, beyond the domain of CRS, cooperative bot-play has been applied by Das et al. [11] in order to optimise a conversational agent specifically for information elicitation. They proposed an algorithm which is trained on a natural dataset that is centered around the task of accurately describing an image to a ‘blind’ conversational agent and fine-tuned using bot-play. The algorithm consists of QBot and ABot agents. The ABot has access to the image, while the QBot does not. The reward is given to both agents and is defined as a function of accuracy of the internal representation of the image generated by the QBot bot. This game setup incentivises QBot to generate questions that are most efficient at reducing uncertainty, and ABot to provide accurate answers to the questions. We believe that any CRS should use its interactivity to elicit information from users in a way that allows it to optimise recommendations in a way that increases their relevance. The aim of this paper is to devise an efficient strategy for extracting valuable information from users within the vast and diverse realm of potential item features. To achieve this, we propose a novel CRS algorithm that harnesses its interactive capabilities to optimise recommendation relevance, aiding users in making good selections.

The key components of the proposed system are the Questioner bot (QBot), Answerer bot (ABot), recommendation network, and policy network. QBot proactively elicits preference information from ABot and updates its internal representation of the ABot’s ratings. ABot retrieves the item and attribute preferences from the user’s history in response to QBot’s questions. The attention-based recommendation network predicts item ratings based on observed preferences, and the policy network guides QBot’s behaviour. The collaborative data matrix enriched with item attribute data is used to pre-train the recommendation component, and the change in the recommendation model loss in QBot’s internal representation is used as a reward to train the policy model.

Contributions. First, a novel algorithm is proposed that directly addresses the need to develop an efficient active learning strategy for a conversational recommendation system, a problem that does not pertain to classical, non-interactive recommenders. This paper describes the first attempt to apply bot-play as choice of a training approach in order to optimise the quality of recommendations in CRS by effectively eliciting user preference information during dialogue. Experimental results that are based on both simplified synthetic and natural datasets showcase the ability of the system to reduce the number of conversational turns required to achieve accurate recommendations compared to the baseline. The proposed approach can be applied on top of supervised pre-training of the encoder-decoder model, which can result in a conversational system that has high practical value in the real world.

Second, a novel recommendation component architecture is proposed that represents a user through their preference history, and so can fully support a cold start scenario at the beginning of a dialogue with a new user. Furthermore, this

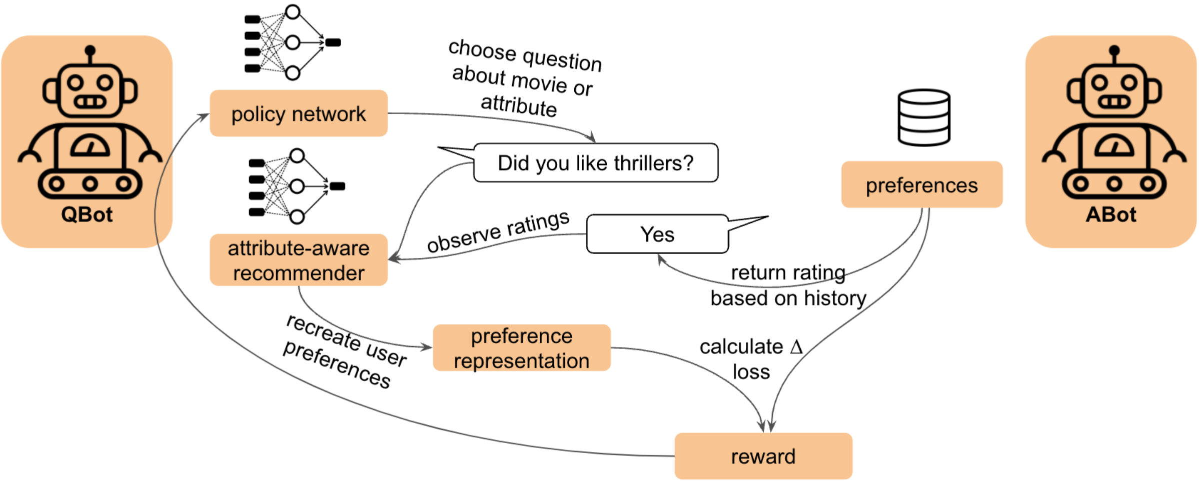
QBot
policy network
choose question about movie or attribute
Did you like thrillers?
attribute-aware recommender
observe ratings
Yes
preferences
return rating based on history
ABot
recreate user preferences
preference representation
calculate Δ loss
reward

Fig. 1. Proposed bot-play driven Conversational Recommender System architecture. QBot elicits preference information from ABot by asking a question about a movie or an attribute, inputs ABot's answers into its recommender module and updates preference predictions. Change in quality of these predictions, defined as recommender loss, is fed to QBot's policy network as a reward, which allows it to learn the best questioning strategy.

algorithm has a useful property of showing a consistent and steady output accuracy improvement as new item-rating inputs are added to the model, which allows use of the model as the source of reward in the bot-play scenario. Results show that the recommendation component can successfully learn mappings between items and attributes, which allows the CRS to make accurate rating predictions based on a limited set of answered questions.

II. Learning Preference Elicitation through Bot-Play

A. System Overview

We now present our novel self-supervised bot-play approach to CRS. This approach allows us to directly optimise QBot agent behaviour for eliciting valuable information efficiently and providing relevant recommendations. Two bots, QBot and ABot, converse with each other about ABot's preferences with the goal of generating accurate recommendations as early as possible in the conversation. The key components of the system include (see Figure 1):

1) QBot, whose goal is to proactively elicit preference information from ABot by asking about item and attribute preferences in order to maximise recommendation quality;
2) ABot, which acts deterministically by retrieving the item and attribute preferences from the user's history in response to QBot's question;
3) Recommendation network ϕ, which predicts item ratings based on the observed preferences of ABot; and
4) Policy network π, which guides the behaviour of QBot and is optimised stochastically using the ϵ-greedy REINFORCE algorithm in the process of bot-play and is rewarded for providing good quality recommendations, defined as decrease in loss of the recommendation network ϕ.

A collaborative data matrix filled with ratings is made available to the QBot in order to pre-train the Recommendation component. A vector of ratings for an additional user (not included in the matrix) is visible only to the ABot.

ABot's action set is comprised of possible questions, including item titles and attribute names. A recent study of natural human conversations about item preferences revealed that besides sharing a couple of examples of items that they liked or disliked, humans rely heavily on generalisations and descriptions of item attributes [12]. Furthermore, eliciting information about individual item titles can be extremely inefficient, considering that most users may not interact with the majority of items. For instance, the MovieLens dataset [13] includes 9066 distinct movies, while a median user in that dataset has only rated for 71 movies, which constitutes a mere 0.8%. Therefore, it is crucial to extend the framework in order to introduce item attributes alongside the item IDs into all of the individual system components to enable QBot to elicit preference information efficiently.

During the dialogue, QBot requests information about specific item titles and attributes from the ABot's 'history' (vector of item ratings), and ABot responds with a rating. Following that, QBot updates its internal representation of the ABot's ratings for all movies based on the conversation history and collaborative data matrix. The change in the recommendation model loss in QBot's internal representation is used as a reward to train the QBot's Policy model.

This approach is inspired by the bot-play training protocol developed by Das et al. [11] for an image guessing game. To allow us to apply this approach to the recommendation scenario, we have replaced: (1) the image — with a representation of user-item ratings; (2) the regression network — with a recommendation module; and (3) reward — with a metric that reflects the improvement in usefulness of the recommendation.

B. Recommendation Module

We propose a novel architecture for a recommender module that utilizes an item-to-item approach in order to predict item ratings. This approach enables the module to make predictions for new users whose user ID was not previously observed and included in the model training data. A Long Short-Term Memory (LSTM) network with an attention mechanism is used to predict, for a user u, the full vector of user's ratings v_u, for k movies from a shuffled partially observed history of ratings s_u. The attention mechanism allows the network to learn the relationships between inputs and outputs directly based on embedded indexes of items and independently of the order of inputs (which can be arbitrary).

The recommendation network ϕ is parameterised by Θ_{rec}. At each timestep t, it receives an item index $a_t \in A$, where A is the set of item titles, and a rating v_a. The rating v_a is one-hot encoded and can belong to one of the three categories that capture a combination of implicit and explicit ratings: (a) liked, (b) disliked, (c) not rated. It embeds the index (embeddings are learned in the process of optimisation), concatenates the embedding with the rating and passes the resulting vector to an LSTM unit, which learns an encoded representation of a user's history. At each timestep, this representation is decoded by passing the LSTM hidden state through two dense hidden layers. The encoded state is connected to decoder outputs with the attention mechanism. Outputs of the decoder and the attention layer are concatenated and passed through the final dense layer with a sigmoid nonlinearity function.

The network is trained through backpropagation using the Adam optimisation algorithm to minimise cross-entropy (CE) loss $\min L(\Theta_{rec})$ for each timestep t:

$$L(\Theta_{rec})_t = -E[v_u \otimes \mathbf{log}(\phi(s_{u,t})) - (1-v_u) \otimes \mathbf{log}(1-\phi(s_{u,t})], \tag{1}$$

where $\phi(s_{u,t})$ is the output of the recommendation neural network for a given user's state $s_{u,t}$, which in turn is defined as the history of ratings of a user u at timestep t and consists of concatenated item indexes and ratings $s_t = (a_t, v_a)$. **log** denotes vectorized logarithm. The full set of item ratings v_u, drawn from all users' histories, serves as ground truth for the model. Items that do not have a rating in a data set for a given user are masked during the calculation of CE loss.

The recommendation network is trained separately on the full user history. Its hidden layers serve as a bottleneck that forces the network to learn to predict ratings of unobserved items based on correlated observed items. Therefore, a new observation of an item-rating pair reduces the loss most when it is uncorrelated with previous observations and information gain is maximised.

During dialogue, at each dialogue timestep t, predictions of the pre-trained recommendation network are calculated based on the inputs collected through dialogue. The between-timestep difference in loss of the recommendation network ΔL serves as a reward observed by the policy network π, as a decrease in recommendation loss L indicates improvement in quality of predictions.

$$r_t = \Delta L = L_{t-1} - L_t \tag{2}$$

C. Information Elicitation Module

We extend the approach of the VisDial framework introduced by [11] to conversational recommenders by training the policy network through goal-driven self-supervised bot-play. First, a recommendation module is trained, which serves the same purpose as a feature regression network in VisDial. The between-timestep difference in loss of the recommendation network is viewed as a proxy for information gain and is ultimately used to calculate the reward observed by the policy network π.

Next, the policy network π parameterised by Θ_{policy} is optimised stochastically using the REINFORCE algorithm during bot-play. The policy network guides the behaviour of QBot, while ABot acts deterministically. The policy network is a feedforward neural net, which receives the following inputs:

1) item ratings known before the current timestep, which represent the state s_{t-1};
2) index of the item that the QBot asked about a_t.

The model is optimised using the observed reward r_t, operationalised as the reduction of the recommendation network loss ΔL compared with the previous conversational turn (see Equation 3). The policy network learns to predict distribution over items that QBot asks about (QBot's action space A, which can also be seen as its vocabulary) through maximising the REINFORCE value function: $\max J(\Theta_{policy})$, where

$$J(\Theta_{policy}) = E[\log \pi(a_t|s_{t-1})r_t] \tag{3}$$

The REINFORCE algorithm maximises the product of the log-likelihood of the selected action a_t and the reward r_t, thus pushing up the probabilities of the actions associated with the highest expected rewards. Therefore, the policy network $\pi(a_t, r_t, s_{t-1})$ learns to assign higher probabilities to those item indexes whose ratings, if obtained, will be associated with higher information gain.

The optimisation is done stochastically through bot-play using an ϵ-greedy algorithm [14]. The bot-play itself is implemented using the ParlAI framework for training dialogue models [15]. Firstly, the vector of the user's item ratings v_u is transformed in accordance with the agent's vocabulary. ABot's vocabulary consists of three phrases ('liked', 'disliked' and 'haven't seen'), while QBot's vocabulary consists of the list of available item titles A. At the initialisation of the dialogue, ABot observes a rating history of a random user v_u (drawn from the dataset). QBot is blinded to this information. At each conversational turn, QBot utters an item title a_{ut}, drawn from the set of item titles according to the policy network $\pi(a_t|s_{t-1})$. The ABot observes the question and utters the rating of the item v_{ut} deterministically, based on the user's history v_u. The QBot observes its own utterance, as well as the response of the ABot, and appends the information to its internal representation of ABot's rating history s_{t-1}. Following that, the recommendation network ϕ is used to compute the loss $L(s_t)$, given the observed history of ratings. The pseudocode is provided in Algorithm 1.

Algorithm 1 Bot-Play Training Procedure

```
Require: user $u$, items $A$, ratings $v_u$, number of conversational rounds $T$, exploration probability $\epsilon$, recommendation function $\phi$
Ensure: policy network $\pi$ parameters $\Theta_{policy}$ are optimised

Randomly initialise policy network $\pi$ parameters $\Theta_{policy}$
for $t = 1, 2, \ldots, T$ do
    generate probability distribution over actions $p(A|s_{t-1}) = \pi(s_{t-1})$
    if $\epsilon >$ uniformRandom(0,1) then
        question $a_t$ = randomChoice($A$)
    else
        question $a_t$ = randomChoice($A$, $p(A|s_{t-1})$)
    end if
    QBot utters $a_t$
    ABot observes $a_t$
    ABot utters rating $v_a$
    QBot observes ($a_t$, $v_a$) and updates state $s_t = s_{t-1} \cup (a_t, v_a)$
    compute predicted ratings $\widehat{v}_t = \phi(s_t)$
    compute CE loss
        $L_t = -\sum_{j=1}^{k}[v_j \log(\widehat{v}_{tj}) - (1 - v_j) \log(1 - \widehat{v}_{tj})]$
    compute reward $r_t = L_{t-1} - L_t$
    compute REINFORCE value functions $J_t = E[\log \pi(a_t|s_{t-1}) r_t]$
    update $\Theta_{policy\ t} = \Theta_{policy\ t-1} - \nabla \log \pi(a_t|s_{t-1}) r_t$
end for
return $\Theta_{policy}$
```

III. Experiments

A. Datasets

The first dataset is generated artificially with the goal of making the development, debugging and evaluation of the algorithm as transparent as possible. The dataset consists of Boolean values that represent item ratings assigned by n users to k movies. The matrix is complete, meaning that all users have ratings for all items. Perfect collinearity is introduced in this dataset in order to evaluate the ability of the algorithm to select items that maximise information gain. Specifically, the item set consists of $k/2$ independent clusters, containing two perfectly correlated items each. Thus, if the conversational agent observes the rating of one of the items in a cluster, asking about the second item would not allow it to further improve the accuracy of its internal representation of the user's tastes.

The second dataset follows a similar format, however, it is generated based on real user ratings. In order to emulate a realistic and human-like information exchange between the ABot and the QBot, it has been enriched with item attributes. Firstly, the MovieLens dataset [13] was used as a source of collaborative data. The number of distinct movies was limited to the 100 most frequently rated titles, and the number of users was limited to those who had at least a quarter of ratings available (37,139 users in total). Secondly, the dataset was enriched with several movie attribute categories: genre, cast & crew and plot keywords. The movie metadata was obtained from Kaggle.com and is based on the TMDB and GroupLens datasets.

The cast & crew feature space was constrained to contain only the most well-known actors (top 50) and movie directors (top 50) in order to reduce the dimensionality of the input feature matrix. Cast & crew popularity p_i was not directly available in the dataset and was approximated, for each movie that the cast member appeared in $j = 1, 2, \ldots, k$, by dividing movie vote count c_j (as a proxy for movie popularity) by the cast member's order of appearance in the credits o_{ji} (as a proxy for their importance to the success of the movie j) and summing over the movies M.

$$p_i = \sum_{j=1}^{k} \frac{c_j}{(o_{ji} + 1)} \tag{4}$$

It has been shown that in natural unscripted human conversations, users do not just rely on easily available movie metadata (e.g., year, genre); rather, they often use less well defined properties such as story, plot, characters, acting and attributes like violence [12]. To address this gap, a set of common keywords mined from movie plots has been added to the dataset.

B. Recommender Component Performance

1) Recommender Component Evaluation: Good performance of the recommendation network is a prerequisite of successful training of QBot's policy network. The recommendation network is trained to predict the full set of ratings based on the partial set. Therefore, at the last timestep of the recurrent network, it is expected to have approximately zero cross-entropy recommendation network loss ΔL, which would confirm that the network has learned to correctly map inputs to outputs based on movie indexes and that intermediate bottleneck layers do not result in information loss. Indeed, the model trained on MovieLens dataset achieves loss of 0.07, as can be seen in Figure 2.

Furthermore, loss ΔL would be expected to decrease as a function of the number of inputs, as more and more of a user's history is revealed through conversation. Looking at the curves shown in Figure 2, it can be seen that at each timestep entropy is reduced. However, information gain gradually slows down as a function of number of timesteps, which suggests that the model can successfully learn to extrapolate from the incomplete available data. This observation is confirmed by the fact that the model can successfully predict ratings of movies that have not been passed as model inputs. Figure 2(b) shows that prediction accuracy consistently improves with the

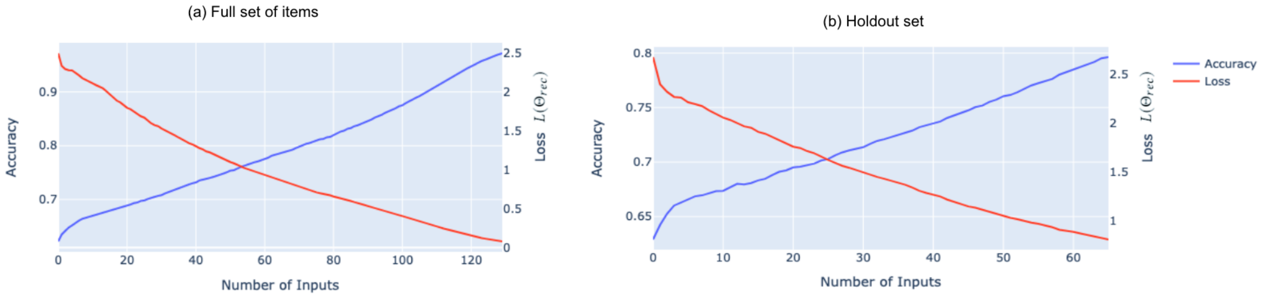

Fig. 2. **Recommendation network evaluation.** Plot shows recommendation network loss ΔL and predicted rating accuracy by number of movie-rating inputs given to the model. These metrics consistently improve with every input, whether some of the output items have been passed to the model as inputs or not.

increase in the number of inputs where sets of input and output movies do not overlap, improving average prediction accuracy from the baseline 0.6 to 0.8.

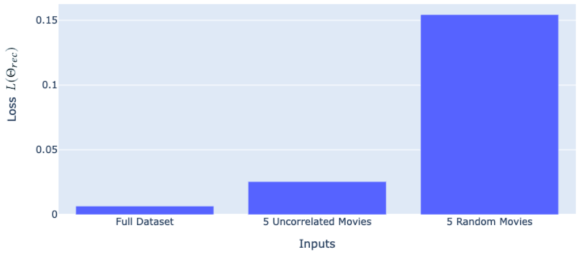

Fig. 3. Recommendation network loss ΔL as a function of entropy in input data. Plot shows recommendation model achieves lowest loss when inputs are representative of all preference clusters.

2) Relationships between Item Correlations and Recommender Loss.: To sense check the model, it can be observed how the recommendation model behaves in the scenario when some movies are perfectly correlated. Perfect collinearity has been introduced in the synthetic dataset. The dataset contains five clusters, each consisting of two perfectly correlated rating vectors. Figure 3 compares loss ΔL in three scenarios: (1) when the full user's history is observed, (2) when half of the movies are observed, each drawn from a separate cluster, and (3) when half of the movies are observed, but the movies are drawn at random. It can be seen that the second scenario results in near perfect loss (0.02), which is similar to the first scenario, which suggests that the model has successfully learned to recreate the full information from a partially observed history. In contrast to that, the third scenario, where information is incomplete, results in high loss, as expected.

The natural dataset drawn from MovieLens does not have perfectly correlated movies, however, it does have a distinct rating cluster formed by different episodes of "Star Wars" (see Figure 4). Figure 5 shows that the loss is considerably lower when predicting the rating given to "Episode IV" if either "Episode V" or "Episode VI" are observed, compared to loss of the rating prediction based on other movies in the dataset.

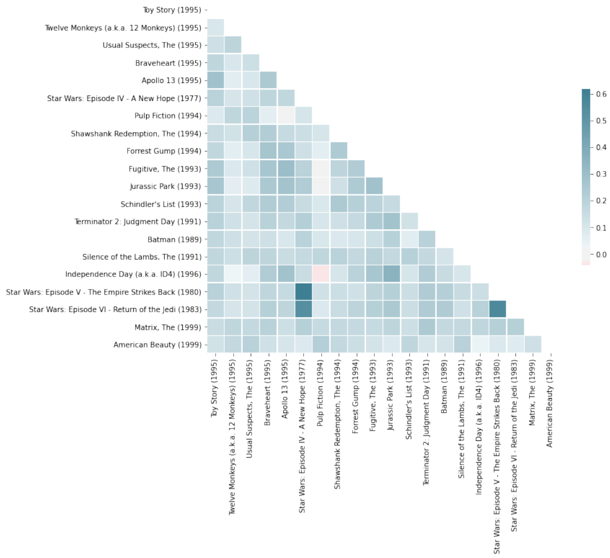

Fig. 4. Pairwise Pearson's correlation coefficients between pairs of rating vectors corresponding to movies drawn from the MovieLens dataset. Plot shows that distinct preference clusters occur in natural datasets.

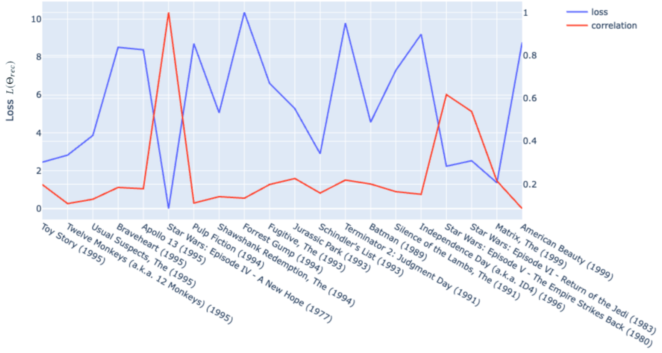

Fig. 5. Pairwise Pearson's correlation coefficients between pairs of rating vectors corresponding to movies drawn from the MovieLens dataset. The plot shows that inputs that are highly correlated with outputs lead to accurate rating predictions.

3) Evaluation of Attribute Enriched Recommendations.: Previous research has shown that movie attributes play a significant role in natural conversations surrounding films. To enable QBot and ABot to use this in their interactions, the dataset has been enriched with attribute ratings, including actor, director and genre, which were derived from movie ratings. The graph in Figure 6 illustrates the model's performance in predicting movie ratings solely based on attribute ratings. As demonstrated, the model has effectively learned the relationships between attributes and items, and leverages this information to make predictions, albeit with slightly lower accuracy than item-to-item predictions.

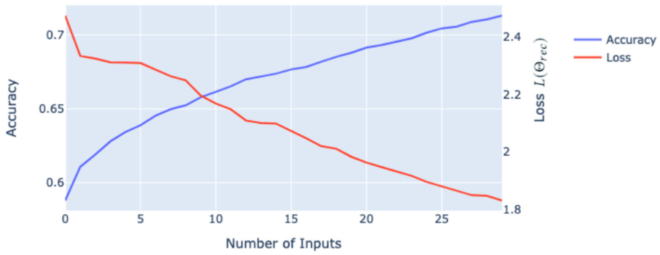

Fig. 6. **Attribute enriched recommendation.** Plot shows recommendation network loss ΔL and predicted rating accuracy by number of attribute-rating inputs given to the model. The model improves item rating predictions with each attribute input.

In order to verify the meaningfulness of model mappings between attributes and individual movies, the ratings of attributes have been manipulated and resulting changes in predictions have been measured, which allows identification of the most affected items. Observe that, where the only known attribute of a movie is that it features the actor Russel Crowe, positive ratings of the actor improve predicted ratings for other movies in which he stars, such as "A Beautiful Mind," "Gladiator," and "L.A. Confidential," which aligns with the expected outcomes (see Table I).

TABLE I
TOP 10 IMPROVED MOVIE RATINGS WHEN ATTRIBUTE "RUSSELL CROWE" WAS MANIPULATED

Title	Rating Difference
"The Silence of the Lambs"	+1.00
"Apollo 13"	+0.94
"The Princess Bride"	+0.94
"The Godfather"	+0.92
"The Mask"	+0.92
"A Beautiful Mind"	+0.91
"Gladiator"	+0.89
"Jurassic Park"	+0.88
"Clear and Present Danger"	+0.83
"L.A. Confidential"	+0.82

C. Policy Network Performance

We constructed QBot to learn an effective questioning strategy that would allow it to uncover as much information as possible about the user in the smallest possible number of conversational turns. Therefore, it is expected that the policy network will learn which questions would be the most impactful given the current state (already observed history), so as to minimise the loss of the recommendation network ΔL at each conversational turn.

As seen in Figure 7(a), on a synthetic dataset, on average, the policy network has learned to recreate the full vector of user's rating in five conversation turns, which corresponds to five uncorrelated movie clusters. In contrast, a random policy does not consistently achieve zero loss even in 10 conversational turns.

On the MovieLens-based dataset (Figure 7(b)), the policy network has learned to significantly reduce loss in the initial conversational turns by asking questions that are most predictive of the full rating vector. However, in contrast with the synthetic dataset, this effect diminishes over conversational turns, which is likely related to the fact that correlations between individual items in the MovieLens based dataset are much weaker than in the synthetic dataset (see Figure 5) and it is not possible to accurately recreate the full set of user's ratings from a small subset of ratings.

IV. CONCLUSIONS

To summarise, a novel CRS system design has been introduced based on a goal-driven self-supervised bot-play approach. A neural recommender module architecture has been proposed that is optimised for cold-start dialogue and for training a policy-based Information Elicitation module to select questions that result in lower recommendation loss. Item attributes are introduced into all of the individual system components alongside item IDs.

The recommendation component learns mappings between items and attributes, which allows the QBot to make accurate rating predictions from a limited set of answered questions. On both synthetic and real world datasets, the trained policy model is more efficient at recreating user preferences than the random baseline. On the synthetic dataset, the learned policy minimised the recommendation model's error in the smallest theoretically possible number of conversational turns. On the MovieLens-based dataset, the policy network learned to reduce loss significantly compared to the baseline, although the effect is less pronounced due to weaker item intercorrelations.

The most significant contribution of our work is the proposed algorithm that directly addresses the need to develop an efficient active learning strategy for a conversational recommendation system, a problem that does not pertain to classical non-interactive recommenders.

The described architecture can be extended to encompass multimodal information obtained from users in natural language by connecting it to a Large Language Model (LLM) based interface. This would enable development of a highly flexible system that would be able to 'understand' and elicit a wide spectrum of information from users. It is important to note that the current approach focuses on the accuracy of the representation of user's tastes. In future work, it is important to consider the novelty aspect of the recommendation — in a real world scenario it is important to train the agent to discover items that the user has not consumed before.

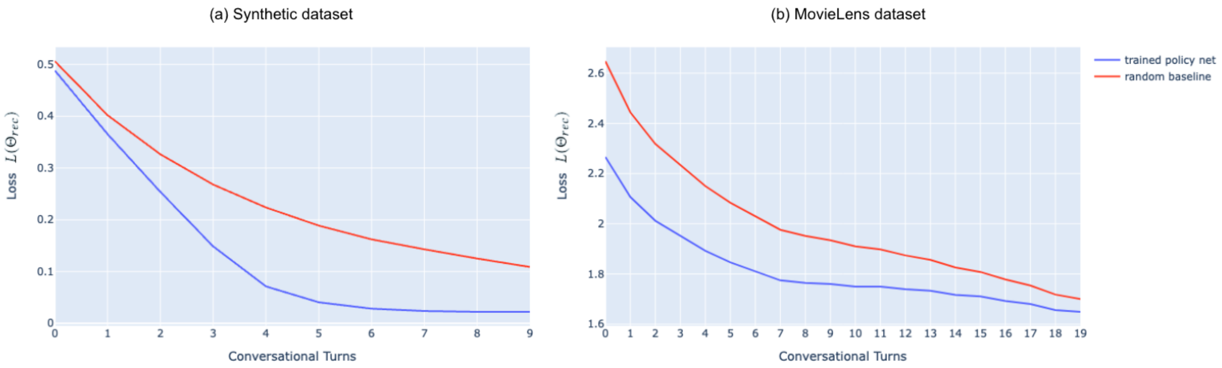

Fig. 7. **Information Elicitation module evaluation.** Plot shows recommendation network loss ΔL by number of conversational turns for random policy and policy trained through bot-play. Policy network has successfully learned which preference questions lead to improvement in prediction accuracy earlier in a dialogue.

References

[1] R. Greifeneder, B. Scheibehenne, and N. Kleber, "Less may be more when choosing is difficult: Choice complexity and too much choice," *Acta psychologica*, vol. 133, no. 1, pp. 45–50, 2010.

[2] A. A. Chandrashekara, R. K. M. Talluri, S. S. Sivarathri, R. Mitra, P. Calyam, K. Kee, and S. Nair, "Fuzzy-based conversational recommender for data-intensive science gateway applications," in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 4870–4875.

[3] P. Kucherbaev, A. Psyllidis, and A. Bozzon, "Chatbots as conversational recommender systems in urban contexts," in *Proceedings of the International Workshop on Recommender Systems for Citizens*, 2017, pp. 1–2.

[4] T. Mahmood, F. Ricci, and A. Venturini, "Learning adaptive recommendation strategies for online travel planning," *Information and Communication Technologies in Tourism 2009*, pp. 149–160, 2009.

[5] R. Li, S. Ebrahimi Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal, "Towards deep conversational recommendations," *Advances in neural information processing systems*, vol. 31, 2018.

[6] I. Vendrov, T. Lu, Q. Huang, and C. Boutilier, "Gradient-based optimization for bayesian preference elicitation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 06, 2020, pp. 10 292–10 301.

[7] S. Li, W. Lei, Q. Wu, X. He, P. Jiang, and T.-S. Chua, "Seamlessly unifying attributes and items: Conversational recommendation for cold-start users," *ACM Transactions on Information Systems (TOIS)*, vol. 39, no. 4, pp. 1–29, 2021.

[8] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 815–824.

[9] D. Kang, A. Balakrishnan, P. Shah, P. Crook, Y.-L. Boureau, and J. Weston, "Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue," *arXiv preprint arXiv:1909.03922*, 2019.

[10] S. Li, B. P. Majumder, and J. McAuley, "Self-supervised bot play for transcript-free conversational recommendation with rationales," in *Proceedings of the 16th ACM Conference on Recommender Systems*, 2022, pp. 327–337.

[11] A. Das, S. Kottur, J. M. Moura, S. Lee, and D. Batra, "Learning cooperative visual dialog agents with deep reinforcement learning," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2951–2960.

[12] F. Radlinski, K. Balog, B. Byrne, and K. Krishnamoorthi, "Coached conversational preference elicitation: A case study in understanding movie preferences," in *SIGDIAL Conferences*, 2019.

[13] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *Acm transactions on interactive intelligent systems (tiis)*, vol. 5, no. 4, pp. 1–19, 2015.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] A. H. Miller, W. Feng, A. Fisch, J. Lu, D. Batra, A. Bordes, D. Parikh, and J. Weston, "Parlai: A dialog research software platform," *arXiv preprint arXiv:1705.06476*, 2017.