Model-Free Preference Elicitation

Carlos Martin[2†] , **Craig Boutilier**[1] , **Tuomas Sandholm**[2,3,4,5] , **Ofer Meshi**[1]
[1]Google Research
[2]Carnegie Mellon University
[3]Strategy Robot, Inc.
[4]Optimized Markets, Inc.
[5]Strategic Machine, Inc.
{cgmartin, sandholm}@cs.cmu.edu, {cboutilier, meshi}@google.com

Abstract

In recommender systems, preference elicitation (PE) is an effective way to learn about a user's preferences to improve recommendation quality. Expected value of information (EVOI), a Bayesian technique that computes expected gain in user utility, has proven to be effective in selecting useful PE queries. Most EVOI methods use probabilistic models of user preferences and query responses to compute posterior utilities. By contrast, we develop model-free variants of EVOI that rely on function approximation to obviate the need for specific modeling assumptions. Specifically, we learn user response and utility models from existing data (often available in real-world recommender systems), which are used to estimate EVOI rather than relying on explicit probabilistic inference. We augment our approach by using online planning, specifically, Monte Carlo tree search, to further enhance our elicitation policies. We show that our approach offers significant improvement in recommendation quality over standard baselines on several PE tasks.

1 Introduction

Recommender systems (RSs) play a vital role in making massive amounts of online content accessible to users in domains such as ecommerce, news, movies, videos, and others [Schafer *et al.*, 1999; Sarwar *et al.*, 2000; Herlocker *et al.*, 2004; Ekstrand *et al.*, 2011; Abel *et al.*, 2011; Hallinan and Striphas, 2016; Linden *et al.*, 2003; Pal *et al.*, 2020; Covington *et al.*, 2016]. Such systems typically leverage past user interactions to learn about a user's preferences and improve their future recommendations. However, in many cases, information about such preferences is lacking, for example, with new users who have little interaction history, or when privacy constraints limit the use of past interactions, a problem known as *cold-start* [Lam *et al.*, 2008; Bobadilla *et al.*, 2012]. Moreover, past interactions may not represent a user's taste accurately, for instance, when accounts are shared with others or when preferences change.

Rather than relying solely on passively observing user behavior, an RS can use *preference elicitation (PE)* to ask users questions about their preferences, thus increasing user agency by allowing them to directly communicate their preferences [Keeney and Raiffa, 1976; Salo and Hämäläinen, 2001; Rashid *et al.*, 2008]. Various types of queries can be used for PE, for example, those involving individual items ("Do you like movie X?") or item comparisons ("Do you prefer movie X to Y?"). In this work, we focus on *attribute-based queries* (e.g., "Do you like science fiction movies?").

The challenge, then, is *how to select queries*. In general, this can be viewed as a sequential decision problem, since the value of a query (hence, the optimal query) at any point will be influenced by the value of the *ultimate recommendation* to the user, and may also be influenced by *subsequent* queries and responses [Boutilier, 2002; Holloway and III, 2003]. Despite this, most query selection approaches in the literature are *myopic*, applying one-step lookahead with various criteria. Among these are information gain methods [Rokach and Kisilevich, 2012; Zhao *et al.*, 2018; Canal *et al.*, 2019], entropy-based methods [Abbas, 2004], polyhedral/volumetric methods [Iyengar *et al.*, 2001; Toubia *et al.*, 2003], ellipsoidal methods [Salo and Hämäläinen, 2001], and minimax-regret methods [Boutilier *et al.*, 2004, 2006].

In this work, we focus on the use of *expected value of information (EVOI)* as our query selection criterion. EVOI scores queries based on the expected improvement in recommendation quality that would result by incorporating the user's query response [Howard and Matheson, 1984; Guo and Sanner, 2010; Viappiani and Boutilier, 2010; Vendrov *et al.*, 2020]. As a Bayesian method, EVOI typically requires maintaining a probabilistic user model and implementing posterior updates given user responses. However, such posterior update is intractable even for simple distributions, such a multivariate Gaussian, requiring approximation [Cohn *et al.*, 2014; Doshi-Velez *et al.*, 2012; Guo and Sanner, 2010].

To address these limitations, we propose a *model-free approach* that obviates the need to make specific assumptions regarding the form of the user distribution. Specifically, we demonstrate the feasibility of training predictive models that capture user response behavior and recommendation utility directly from observational data. These models subsequently serve for query selection, eliminating the need for explicit distributional assumptions. Beyond this, we extend the typi-

[†] Part of the work done while at Google.

cal *myopic* usage of EVOI by incorporating multi-step planning to consider the value of information associated with a sequence of queries. We do this using online *Monte Carlo tree search (MCTS)* [Coulom, 2007].

The rest of the paper is organized as follows. We formulate the PE problem in Section 2 and briefly discuss related work in Section 3. We review the typical method for myopic EVOI-based query selection in Section 4, and the model-based approach to predicting user responses and utilities in Section 5. In Section 6, we describe our model-free technique in general terms, and outline the novel use of various neural network architectures we adopt to this end, while in Section 7, we detail our non-myopic approach to query selection using MCTS. In Section 8, we describe our experimental setting and present our experimental results. We conclude with Section 9 and suggest directions for future research.

2 Problem Formulation

We assume an RS that can recommend items from set $\mathcal{X}$, but can also engage in PE by posing queries to a user about their preferences. We let $\mathcal{Q}$ be the set of queries from which the RS can choose, and let $\mathcal{R}_q$ be the set of possible user responses to any $q \in \mathcal{Q}$. In our formulation, we assume that $\mathcal{Q}$ is finite and, for each $q \in \mathcal{Q}$, $\mathcal{R}_q$ is finite. The RS's interaction with a user proceeds as follows. Given the prior queries and user responses, the RS poses a query $q \in \mathcal{Q}$, to which the user then provides a response $r \in \mathcal{R}_q$. This process repeats for T steps (where T is a finite bound on the number of RS queries) or until the user decides to terminate the process. We include a special response token $r_{\text{stop}} \in \mathcal{R}_q$ for all $q \in \mathcal{Q}$ to reflect user-initiated interaction termination. We assume that users are sampled from some prior distribution, for example, as determined by a dataset of user-RS interactions.

A *history* is a sequence of at most T query-response pairs (q, r) such that $r \in \mathcal{R}_q$ and r_{stop} occurs only (if at all) in the final pair. A history is called *terminal* if and only if it ends with r_{stop} or has length T.[1] Let $\mathcal{H}^T$ be the set of terminal histories, $\mathcal{H}^P$ be the set of partial (i.e., non-terminal) histories, and $\mathcal{H} = \mathcal{H}^T \cup \mathcal{H}^P$ be the set of all histories.

At the end of the session with terminal history $h \in \mathcal{H}^T$, the RS recommends an item $x \in \mathcal{X}$ to the user, who derives some (scalar) utility from x, which we denote by $\text{EU}(x|h)$. We wish to design an RS that poses queries and issues recommendations that have high utility to the user. Since we wish to focus on the *elicitation* aspect of this problem, we assume that the item recommendation policy is fixed and can be treated as a black box $\pi_{\text{item}} : \mathcal{H}^T \rightarrow \mathcal{X}$. Given this fixed policy π_{item}, we can associate utility with any terminal history $h \in \mathcal{H}^T$ in the obvious way, $\text{EU}(h) = \text{EU}(\pi_{\text{item}}(h)|h)$.[2] We discuss item utility further in Section 5. Here, we only assume that π_{item} returns high-value or near-optimal items conditioned on its history. In what follows, we assume that π_{item} can be applied to partial histories $h \in \mathcal{H}^P$ in the obvious way. We note that π_{item} (and π_{query} defined below) will generally depend on additional user context and prior history—we ignore this to streamline the presentation, but nothing below hinges on this simplification.

We represent queries in an embedding space $\mathbb{R}^d$. For attribute-based queries, this means attributes that are similar (e.g., "Thriller" and "Horror") are more likely to be close together in embedding space than attributes that are different (e.g., "Comedy" and "War"). For convenience, queries and items are embedded in the same space, and both are computed using *collaborative filtering (CF)* techniques. We will provide more details in Section 8.

Our problem is how to choose optimal queries via a PE policy $\pi_{\text{query}} : \mathcal{H}^P \rightarrow \mathcal{Q}$. Since this is a sequential decision problem [Boutilier, 2002; Holloway and III, 2003], in principle, we could apply model-free *reinforcement learning (RL)*, with the set of queries $\mathcal{Q}$ as the action space and histories $\mathcal{H}^P$ as the state space. However, in our setting we can leverage the specific structure of the problem to derive a more efficient approach. Specifically, we exploit the fact that each episode consists of queries followed by user responses, and a final recommendation which determines user utility. We also use multi-step planning at execution time, which has been shown to improve performance in many settings [Tamar *et al.*, 2016; Guez *et al.*, 2018, 2019; Farquhar *et al.*, 2018; Oh *et al.*, 2017; Silver *et al.*, 2016a]. This is because it is prohibitively complex to do such reasoning for all paths in advance (or to learn a purely reactive policy that can perform as well), but one can often afford to do focused search starting from nodes on the path that are *actually reached*.

3 Related Work

Considerable work on PE has been done in multi-criteria decision analysis, management science, operations research and artificial intelligence, including for RSs. We distinguish two broad classes of work. Our work focuses on the types of *content RSs* in which users often seek many recommendations over time, and that exploit generalization across users, as exemplified by CF and related methods [Salakhutdinov and Mnih, 2007; He *et al.*, 2017; Yang *et al.*, 2020a]. These settings tend to be lower stakes (e.g., movies, music). By contrast, PE has been more widely studied in settings where a user tends to make a single decision and is prepared to invest more time engaging in with a PE process [Salo and Hämäläinen, 2001; Toubia *et al.*, 2003; Boutilier *et al.*, 2004; Dubus *et al.*, 2009; Boutilier, 2013], and generalization across users is less valuable. That said, we do draw on notions like EVOI that are more commonly applied in one-shot, higher-stakes settings.

PE in the latter settings is typically approached in a myopic fashion, with several approaches for query optimization considered in the literature. Among these, *maximum information gain* uses a distribution over user preferences, selecting the query whose expected response maximizes some measure of information [Rokach and Kisilevich, 2012; Zhao *et al.*, 2018; Canal *et al.*, 2019], as do related entropy-based

[1] The RS itself could decide to terminate PE itself before T steps and move to a recommendation; but for ease of exposition, we use a fixed maximum horizon T.

[2] In fact, our formulation supports a more general notion of utility, not necessarily item-based. The only requirement is that the utility is derived from the interaction between the user and the RS, which takes h as input to guide its recommendations.

methods [Abbas, 2004]. Other approaches include polyhedral/volumetric methods [Iyengar *et al.*, 2001; Toubia *et al.*, 2003], ellipsoidal algorithms [Salo and Hämäläinen, 2001], coverage maximization [Meshi *et al.*, 2023], and minimax-regret-based techniques [Boutilier *et al.*, 2004, 2006; Braziunas and Boutilier, 2010; Boutilier, 2013], while Bourdache *et al.* [2019] propose a Bayesian logistic-regression technique, and Chajewska *et al.* [1998] cast PE as a classification problem using decision trees.

Several approaches have been developed for PE, that like ours, optimize the *sequence* of queries. These methods have tended to focus on *offline* computation of full PE policies (in contrast to our online method). Early approaches formulated the PE problem as a partially observable Markov decision process (MDP) [Boutilier, 2002; Holloway and III, 2003], taking a Bayesian approach to modeling uncertainty over a user's preferences. More recently, Vayanos *et al.* [2020] proposed mathematical programming models for multi-turn PE assuming polyhedral uncertainty; these methods are computationally intense, but may be well suited to one-off, high-stakes decisions. Within the setting of CF models for content RSs, various techniques have been developed that use PE to improve the quality of recommendations. Some of this work focuses on *onboarding* new users to help resolve the cold-start problem [Lam *et al.*, 2008; Bobadilla *et al.*, 2012]. Other research deals more directly with in-session interactive rating [Boutilier *et al.*, 2003; Zhao *et al.*, 2013].

The problem of asking good elicitation questions has also been addressed more recently for questions generated by a language model. Rao and III [2018] build a neural network model to rank clarification questions, inspired by expected value of perfect information. However, they do not handle multi-turn question sequences. Yu *et al.* [2019] tackle interactive classification by selecting questions using entropy minimization, while Aliannejadi *et al.* [2019] study the problem of asking good clarifying questions using an information retrieval framework. Kuleshov and Ellis [2023] introduce an inference-time algorithm that helps *large language models (LLMs)* infer user preferences by posing informative questions. It uses a probabilistic model whose conditional distributions are defined by prompting an LLM, and returns questions that minimize expected entropy and model change.

With respect to complexity of using EVOI, it has been noted that EVOI-based query selection is difficult in part because of computationally demanding Bayesian inference for individual query response updates, which itself typically requires approximations [Cohn *et al.*, 2014; Doshi-Velez *et al.*, 2012]. Guo and Sanner [2010] note that Bayesian PE methods that maintain a belief distribution over utility functions and update beliefs using a realistic query confusion model are a natural way to handle noise, although exact inference in these Bayesian models is often intractable.

4 Expected Value of Information

As noted above, while computing optimal queries can be cast as a sequential decision problem [Boutilier, 2002; Holloway and III, 2003], most approaches select queries *myopically* according to some criterion. One criterion which has been used successfully in the past is EVOI [Chajewska *et al.*, 2000; Guo and Sanner, 2010; Viappiani and Boutilier, 2010; Vendrov *et al.*, 2020]. To define (myopic) EVOI, we first define the *posterior expected utility (PEU)* of a query $q \in \mathcal{Q}$ given a history $h \in \mathcal{H}^P$ as

$$\text{PEU}(q|h) = \textstyle\sum_{r \in \mathcal{R}_q} P(r|h,q)\text{EU}(h\|(q,r)) \; . \qquad (1)$$

Here, $h\|(q,r)$ denotes h with (q,r) appended, $\text{EU}(h)$ is the *expected utility* under h, and $P(r|h,q)$ is the probability of the user responding with r given the history h and query q. PEU measures the utility of a query by taking an expectation over all possible user responses. This is effectively one-step lookahead in the query space.

The EVOI of a query q given h is

$$\text{EVOI}(q|h) = \text{PEU}(q|h) - \text{EU}(h) \; . \qquad (2)$$

The query that maximizes $\text{PEU}(q|h)$ also maximizes $\text{EVOI}(q|h)$ since $\text{EU}(h)$ does not depend on q. EVOI measures the improvement in expected utility offered by q relative to not asking any query. It serves not only as a means for ranking potential queries, but also as a useful stopping criterion for elicitation (e.g., when it is sufficiently small).

5 Model-Based Prediction

In this section we review a generic model-based approach to the prediction of user responses and utilities. We assume a user is represented as a distribution over user embeddings $\mathcal{U}$, where $\mathcal{U} \subseteq \mathbb{R}^d$ is the space of possible embeddings, which capture any relevant latent state (e.g., item utilities) as in, say, standard CF approaches [Salakhutdinov and Mnih, 2007]. The *user response model* specifies the probability $P_u(r|u,q)$ of $u \in \mathcal{U}$ responding to $q \in \mathcal{Q}$ with $r \in \mathcal{R}_q$, while the *user utility model* $v(x,u)$ specifies the utility of item $x \in \mathcal{X}$ to $u \in \mathcal{U}$. Notice that the response model P_u is different than $P(r|h,q)$ from Section 4 in that it depends directly on the user rather than the history h. This can be viewed as an application of Bayesian inference: $P(r|h,q) = \int P(u|h)P_u(r|u,q)du$. (Here, $P(u|h,q) = P(u|h)$ because q is an action the RS takes, and is assumed not to give us any additional information about the user.)

In model-based PE, the RS starts with a prior belief $P(u)$. At each step of a session, the RS poses a query q to which the user responds with $r \in \mathcal{R}_q$. Given the induced *history* $h \in \mathcal{H}$, the RS updates its user belief using Bayes' rule to obtain posterior $P(u|h)$. If the utility function is linear, that is, $v(x,u) = u \cdot x$, then the expected utility of recommending $x \in \mathcal{X}$ at that point is

$$\mathbb{E}_{u|h} v(x,u) = \mathbb{E}_{u|h}(u \cdot x) = \left(\mathbb{E}_{u|h} u\right) \cdot x = \bar{u} \cdot x \; , \qquad (3)$$

where $\bar{u} = \mathbb{E}_{u|h} u$. If $h \in \mathcal{H}^T$ is terminating, we assume the item with highest expected utility is recommended, defining the history's utility $\text{EU}(h) = \max_{x \in \mathcal{X}} \bar{u} \cdot x$.

If the posterior is a multivariate Gaussian, then $\bar{u}$ is its mean. However, posterior update is intractable even in this simple case. It can be approximated using the Laplace approximation [Kass *et al.*, 1991], or one can sample users from the posterior using *Markov chain Monte Carlo (MCMC)*

[Metropolis *et al.*, 1953; Hastings, 1970] and use these samples to estimate the required expectations.

These approaches are *model-based* because they explicitly model a user u via a distribution $P(u|h)$. They also assume a specific form of utility, e.g., linear, as in Vendrov *et al.* [2020]. However, the linearity and Gaussianity assumptions might be too restrictive to capture realistic user preferences. For example, the mean of the user distribution might not capture a user's preferences well when the user has multiple interests in the item space. Moreover, even with these simplifications, posterior computation is still intractable. It is these restrictive assumptions and computational challenges we address next.

6 Model-Free Prediction

Our model-free approach obviates the need for the strict modeling assumptions and computational demands of model-based methods, which were outlined in Section 5. It is motivated by real-world settings, where an existing PE policy is deployed and can be used to collect training data for new policies [e.g., Meshi *et al.*, 2023]. We assume access to a dataset of *episodes*, each containing a full history h and the utility of the ultimate RS recommendation x, as measured by user satisfaction with x (through user surveys, proxy measures like engagement, etc.). With this data, we can fit a function approximator to *directly* predict both $P(r|h,q)$ and $\mathrm{EU}(h)$, which can be used to compute PEU (Equation 1). Compared to Bayesian approaches, this obviates the need to model the full posterior $P(u|h)$, replacing model-based posterior computation using Bayesian inference with model-free prediction. This avoids posterior approximations and unnecessary, restrictive assumptions about user utility.

We formalize the prediction and function approximation problems as follows. Let $\mathcal{R} = \bigcup_{q\in\mathcal{Q}} \mathcal{R}_q$ denote the set of possible responses. A *response prediction model* $f_{\mathrm{r}} : \mathcal{H} \times \mathcal{Q} \to \triangle\mathcal{R}$ takes as input a history and a query and outputs a distribution over responses.[3] A *utility prediction model* $f_{\mathrm{v}} : \mathcal{H} \to \mathbb{R}$ maps histories to expected values/utilities. For response prediction, we convert its input (h, q) to a *pseudo-history* $h\|(q, r_{\text{pseudo}})$, where r_{pseudo} is a *pseudo-response* token that is not in $\mathcal{R}$. This ensures that both response and utility prediction have the same input form (h), which allows us to treat both using the same model architectures.

We consider five distinct neural network architectures to learn response and utility predictions. Each takes a history (or pseudo-history) as input. The history's queries are converted to *query embeddings* in $\mathbb{R}^d$, where the embeddings can be either learned or given to the model.

Affine. This architecture sums the query embeddings associated with each possible response, concatenates the results, and passes the resulting vector to an affine layer:

$$\mathrm{affine}\big(\mathrm{cat}\big(\{\textstyle\sum_t [r_t = r]\mathrm{embedding}(q_t)\}_{r\in\mathcal{R}}\big)\big), \quad (4)$$

where $\{x_t\}_t$ denotes the sequence whose entry at t is x_t, and $[\cdot]$ is the indicator function.

Recurrent. This applies a *recurrent neural network (RNN)* [Rumelhart *et al.*, 1986; Werbos, 1988] to the history. We use the *gated recurrent unit (GRU)* [Cho *et al.*, 2014]:

$$\mathrm{RNN}(\{\mathrm{embedding}(q_t)\|\mathrm{onehot}(r_t)\}_t), \quad (5)$$

where $\mathrm{onehot}(k)$ is the kth standard basis vector. RNNs are permutation sensitive and can therefore learn temporal dependencies if they exist. We use GRUs, which tend to be competitive with most RNNs [Yang *et al.*, 2020b].

DeepSets. Murphy *et al.* [2018] introduce *Janossy pooling*, a unifying framework for methods that learn strictly permutation-invariant functions (or suitable approximations). Janossy pooling considers all possible permutations π of its input elements. Each permutation is separately passed through a fixed permutation-sensitive function ϕ, and the outputs are aggregated by averaging (or some other global pooling operation). A second network is then applied to predict the final output. This guarantees permutation invariance without imposing restrictions on any network component. Instead of dealing with all (factorially many) permutations, one may consider only k-ary interactions. *DeepSets* [Zaheer *et al.*, 2017], a special case of Janossy pooling with $k = 1$, applies an encoder to each element of the input, aggregates the outputs through a pooling operation (such as summation), and applies a decoder to the result. In our case, the input elements are the queries concatenated with their respective responses. For the encoder and decoder, we use fully-connected *multi-layer perceptrons (MLPs)*:

$$\mathrm{decoder}(\textstyle\sum_t \mathrm{encoder}(\mathrm{embedding}(q_t)\|\mathrm{onehot}(r_t))). \quad (6)$$

Attention. This architecture uses multi-head self-attention [Vaswani *et al.*, 2017], which works as follows. The attention module accepts a sequence of queries Q, keys K, and values V, and computes the following sequence:

$$\mathrm{attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}V\right). \quad (7)$$

The *multi-head attention (MHA)* module receives input sequences X, Y, and Z, and has several heads, each of which linearly transforms the input sequences into queries, keys, and values. It then applies the attention module to each, concatenates the results, and applies a linear transformation:

$$\begin{aligned}\mathrm{MHA}(X, Y, Z) &= \mathrm{concat}(\mathrm{head}_1, \ldots, \mathrm{head}_n)W^O,\\ \text{where}\quad \mathrm{head}_i &= \mathrm{attention}(XW_i^Q, YW_i^K, ZW_i^V).\end{aligned} \quad (8)$$

In our case, we perform *self*-attention on the query-response history, where each element of the sequence is the query embedding concatenated with its corresponding response:

$$X = Y = Z = \{\mathrm{embedding}(q_t)\|\mathrm{onehot}(r_t)\}_t\,. \quad (9)$$

We use LayerNorm [Ba *et al.*, 2016], specifically, pre-layer normalization (Pre-LN) [Xiong *et al.*, 2020] to normalize activations.

Multisets. This architecture converts the history to table of *counts* $|\mathcal{Q}| \times |\mathcal{R}| \to \mathbb{N}$ for each query-response pair, flattens this table, and passes it to a feed-forward network:

$$\mathrm{MLP}(\textstyle\sum_t \mathrm{onehot}(q_t) \otimes \mathrm{onehot}(r_t)), \quad (10)$$

where $a \otimes b$ denotes the outer product of vectors a and b.

[3] Any additional context or side information about the user can also be input, but we do not assume any such information here.

To each of these architectures, we append a fully-connected layer that outputs a scalar utility prediction $\hat{u} \in \mathbb{R}$ and a vector of probabilities $\hat{p} \in \mathbb{R}^{|\mathcal{R}|}$ for response prediction. For the utility prediction, which is a regression problem, we use a squared error loss. For the response prediction, which is a classification problem, we use a cross entropy loss.

7 Multi-Step Planning

Instead of implementing one-step lookahead with respect to user responses, as in standard myopic approaches to EVOI usage, we can perform multi-step lookahead search to better detemerine the value of a (conditional) *sequence* of PE queries or PE *policy*, in the spirit of non-myopic PE approaches [Boutilier, 2002; Holloway and III, 2003]. We focus on the use of online search, were we employ algorithms such as *depth-limited search (DLS)* and *Monte Carlo tree search (MCTS)* [Coulom, 2007]. The latter allows the search tree to grow asymmetrically toward more promising paths. The user response model is used at the search tree's chance nodes, and the utility model is used at its leaves. "Classic" myopic EVOI is equivalent to DLS with a one-step lookahead.

Recent MCTS variants use value function approximation to guide the search [Silver *et al.*, 2016b, 2017, 2018; Danihelka *et al.*, 2022]. Stochastic MuZero [Antonoglou *et al.*, 2022], which extends MuZero [Schrittwieser *et al.*, 2020] to plan with a stochastic model, is especially appropriate in our setting, since user responses are stochastic. In our experiments, we use the open-source implementation found in DeepMind's JAX library mctx[4] [Babuschkin *et al.*, 2020].

The MCTS Algorithm. MCTS iteratively grows a search tree. Each node of this tree is either a *decision node*, corresponding to a point at which the agent chooses an *action*, or a *chance node*, corresponding to a point at which the environment chooses an *outcome*. The root of the tree is a decision node. In our setting, decision nodes represent points at which the RS chooses a query $q \in \mathcal{Q}$, and chance nodes denote the user selecting a response $r \in \mathcal{R}_q$. Thus, the path to a decision node is a history $h \in \mathcal{H}$, while the path to a chance node is a history-query pair $(h, q) \in \mathcal{H} \times \mathcal{Q}$. A decision node h contains, for each query $q \in \mathcal{Q}$, a visit count $N(h, q)$, value estimate $Q(h, q)$, and prior probability $P(h, q)$.

The algorithm proceeds over B iterations, where B is the *simulation budget*. Each iteration of the algorithm consists of three phases: *selection*, *expansion*, and *backpropagation*.

Selection. Starting from the root, the algorithm traverses the tree until it reaches a leaf edge. At a decision node h, it selects the query q that maximizes the upper confidence bound [Silver *et al.*, 2016a]: $\mathrm{UCB}(h, q) = Q(h, q) + P(h, q)\frac{\sqrt{1+\sum_{q' \in \mathcal{Q}} N(h,q')}}{1+N(h,q)}\left(c_1 + \log \frac{\sum_{q' \in \mathcal{Q}} N(h,q')+c_2+1}{c_2}\right)$. Here, c_1 and c_2 are constants that control the relative importance of the value estimates and prior probabilities; we use the same values as Antonoglou *et al.* [2022]. Once q is selected, the chance node (h, q) becomes the new node, at which a response $r \in \mathcal{R}_q$ is sampled according to a black-box response model $P(r|h, q)$ (an input to the algorithm). Once r is selected, the decision node $h \| (q, r)$ becomes the new node.

Expansion. When a leaf edge is reached, a new node is added to the search tree. Its value estimate is computed from a black-box value model (an input to the algorithm). In our setting, this is a prediction of the user utility of the item $\pi_{\text{item}}(h)$ that would be recommended by the RS if the episode were to end at that point. If the new node is a decision node, its lookup table is initialized with $N(h, q) = 0$ and $Q(h, q) = 0$ for each query $q \in \mathcal{Q}$. Optionally, we can also specify a prior probability $P(h, q)$ for each action, which can reflect prior knowledge about which queries are better than others. In our case, we use a uniform prior.

Backpropagation. The value estimate v of the newly-added edge, which is determined by the utility prediction model, is propagated up the tree to the root. At each decision node $h \in \mathcal{H}$ on this path, the algorithm updates the statistics for each query $q \in \mathcal{Q}$ as follows: $Q(h, q) \leftarrow \frac{N(h,q)Q(h,q)+v}{N(h,q)+1}$, $N(h, q) \leftarrow N(h, q) + 1$.

After B iterations, the size of the search tree is $B + 1$, and we select an action from the root. To do this, we use the same strategy as MuZero [Schrittwieser *et al.*, 2020] and Stochastic MuZero [Antonoglou *et al.*, 2022] to sample an action at the root, as implemented in Google DeepMind's JAX library mctx [Babuschkin *et al.*, 2020]. This strategy samples actions from the root according to their visitation counts.

8 Experiments

We empirically compare the performance of our model-free approach (Section 6) to the model-based approach (Section 5) in terms of the expected utility to the user of the recommended item at the end of an episode. To do this, we use three datasets.

MovieLens 1M. The MovieLens 1M dataset [Harper and Konstan, 2016] consists of movie ratings by users as well as a set of genres for each movie.[5] For this dataset, the RS's PE queries to the user take the form "Do you like genre X?", and the user responds with either "yes" (1) or "no" (0).

MovieLens 25M. The MovieLens 25M dataset [Harper and Konstan, 2016] is analogous to the MovieLens 1M dataset but with a greater number of ratings, tags, and items. Here RS's PE queries are again binary (yes/no), but querying movie tags rather than genres. (Examples of tags include: 'post-apocalyptic', 'World War II', and 'alternate reality'.)

Amazon Reviews. The 2018 Amazon review dataset [Ni *et al.*, 2019] contains product reviews and metadata from Amazon. We use the "CDs and Vinyl" dataset, with RS binary queries focusing on music primary genres and first-level sub-genres. (Examples of subgenres include: 'piano blues', 'electronica', and 'baroque pop'.) Dataset statistics are shown in Table 1. ("Attributes" refers to genres, tags, or genres/subgenres in the three datasets as described above). We use the 100 most common attributes and 1000 most common items for each dataset.

[4] https://github.com/google-deepmind/mctx

[5] The genres are action, adventure, animation, children's, comedy, crime, documentary, drama, fantasy, film-noir, horror, musical, mystery, romance, sci-fi, thriller, war, and western.

Dataset	Users	Items	Attributes	Ratings
ML 1M	6,041	3,953	18	1,000,209
ML 25M	162,542	209,172	1,129	25,000,095
Amazon	1,944,316	412,325	511	4,543,369

Table 1: Dataset statistics.

To simulate user responses and utilities, we use CF to embed users, items, and queries in a joint embedding space of dimension 50. We use probabilistic matrix factorization [Salakhutdinov and Mnih, 2007] to computing embeddings, solving the optimization problem:

$$\begin{aligned}\operatorname*{argmin}_{U,X,Q} w_1\mathcal{L}_R(U^\top X, R) + w_2\mathcal{L}_A(X^\top Q, A) \\ +w_3\|U\|_2^2 + w_4\|X\|_2^2 + w_5\|Q\|_2^2 \,.\end{aligned} \tag{11}$$

where U, X, Q, R, A are matrices representing user embeddings, item embeddings, query embeddings, user-item ratings, and item-attribute association, respectively. As a preprocessing step, we rescale ratings to the unit interval using min-max normalization. We use $\mathcal{L}_R(x, y) = (\sigma(x) - y)^2$ as a regression loss for ratings, where σ is the logistic sigmoid function. Our classification loss for attributes, $\mathcal{L}_A$, uses sigmoid binary cross-entropy loss. The last three terms reflect L_2 regularization. We solve for U, X, Q using gradient descent. As shown in Figure 1, we obtain low regression and classification errors.

Given user, item, and query embeddings, we generate synthetic episodes by sampling queries from a default PE policy and user response model. The system and user interact for $T = 10$ rounds, or until the user stops the process. At that point, the RS recommends an item and observes the user's utility for that item. In practice, trajectories will be obtained from a deployed policy interacting with real users and utility will be measured using some proxy [e.g., Meshi *et al.*, 2023].

For simplicity, we employ a random policy for data generation, where the query is selected uniformly at random. In practice, a deployed elicitation policy will have a lower coverage of the query space, but here we use a random policy for illustration. The response of user u to query q is either the special stop response r_{stop} with a fixed probability (we use 0.1 at each time step), or sampled from a Bernoulli distribution with parameter $\sigma(u \cdot q)$. Given recommended item $x \in \mathcal{X}$, user u's utility is the dot product of the corresponding user and item embedding $u \cdot x$—we use a linear utility for simplicity, but, as noted above, our approach supports non-linear utilities, or more complex evaluation of user value.

We assume a Bayesian recommender which uses a multivariate Gaussian to represent users, with an MCMC-based approximation for posterior update $P(u|h)$. To recommend an item, it computes $\bar{u}$ and returns the best item, $\operatorname{argmax}_{x\in\mathcal{X}} \bar{u} \cdot x$. Our approach is general and uses the recommender as a black box. Therefore, any recommender which recommends items based on histories h can be incorporated. Each episode consists of the history h together with the user utility of the recommended item. We emphasize that the user embeddings are *only* used for generating the synthetic dataset. Neither the RS, nor its elicitation module, have access to the user embeddings—they see only histories h.

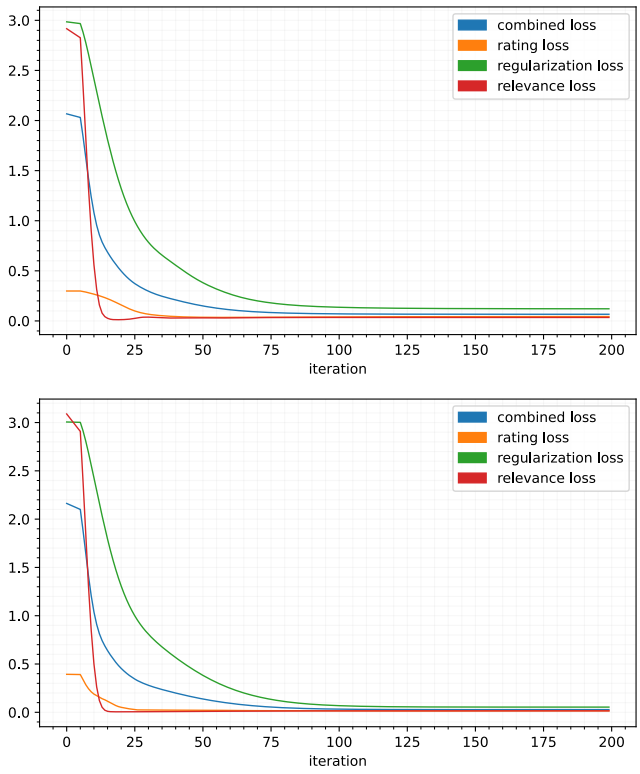

Figure 1: Embedding losses for MovieLens 1M (top) and Amazon reviews (bottom) over the course of fitting. Each iteration is a gradient descent optimization step for the embeddings.

Model Details. All of our models use a hidden layer of size 1024 where applicable, and four heads for the attention model. We use the Radam optimizer [Liu *et al.*, 2019] (based on Adam [Kingma and Ba, 2014]) with a learning rate of 10^{-5} and weight decay 10^{-3}. We train using 100 epochs, a batch size of 32, 20 trials, and 10% of the dataset for validation. Where needed by the network, we use *recitified linear unit (ReLU)* activation functions [Fukushima, 1975].

For model-based response and utility prediction models, we sample users from the posterior and compute expectations. To do this, we use MCMC, which constructs a Markov chain that has the desired distribution $P(u|h)$ as its stationary distribution. We use *Hamiltonian Monte Carlo (HMC)* [Duane *et al.*, 1987], specifically the implementation in TensorFlow Probability [Abadi *et al.*, 2015] for JAX [Bradbury *et al.*, 2018]. HMC uses the derivatives of the density being sampled to generate efficient transitions spanning the posterior. It numerically integrates a Hamiltonian dynamics (we use the leapfrog method) to propose new points in the sample space and then performs a Metropolis acceptance step.

Results. We first examine the performance of the various neural network architectures, outlined in Section 6, in learning to predict responses and utilities on the MovieLens 1M dataset. Figure 2 shows the utility and response loss on the validation set for the different architectures over the course

of training. Solid lines show the mean across trials, with bands indicating a 0.95 confidence interval. The latter is computed using *bias-corrected and accelerated (BCa)* bootstrapping [Efron, 1979, 1987]. We see that the DeepSets architecture performs best with respect to the sum of both losses.

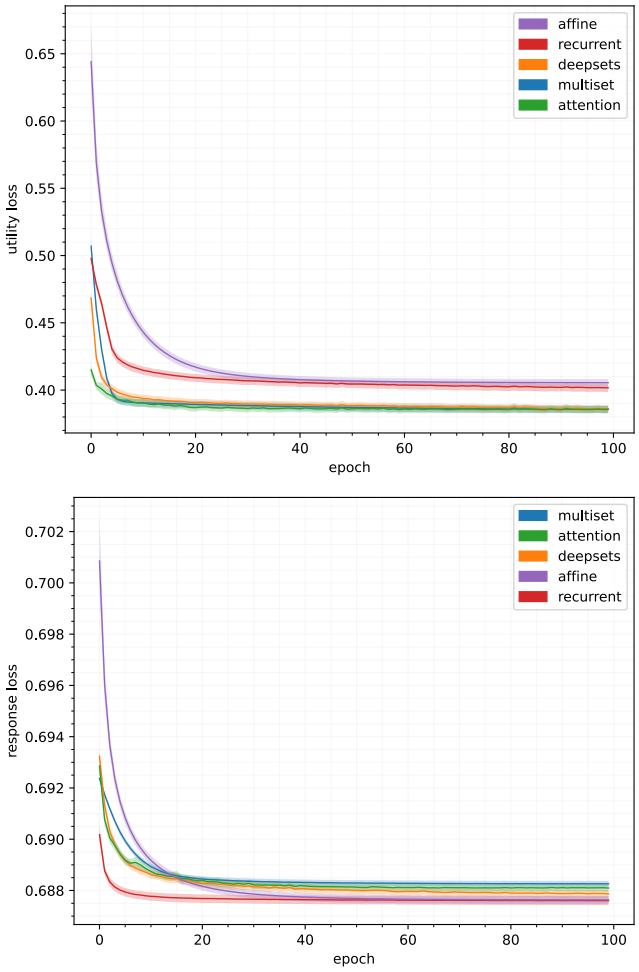

Figure 2: Losses on the episode validation set for MovieLens 1M over the course of training for various architectures. Legend labels are ordered according to the loss of the final epoch.

We next combine DeepSets, the best-performing architecture, with different algorithms (DLS or MCTS with various depths and simulation budgets, respectively) to create different elicitation *policies*. We evaluate the resulting policy by running it against synthetic users and recording the resulting episode utilities. Results are shown in Figure 3. Here, dots show means across trials and error bars show a 0.95 confidence interval for this mean. In these plots: the "mb-" prefix denotes "model-based," "mf-" denotes "model-free," the notation"mcts:n" denotes MCTS with simulation budget of n, and "dls:n" denotes DLS with depth n. We also include a random query policy as a very simple baseline. We use 3000 episodes for each policy, and execute them on a single NVIDIA A100 SXM4 40GB GPU. We show run times in Figure 4. We include various simulation budgets for MCTS.

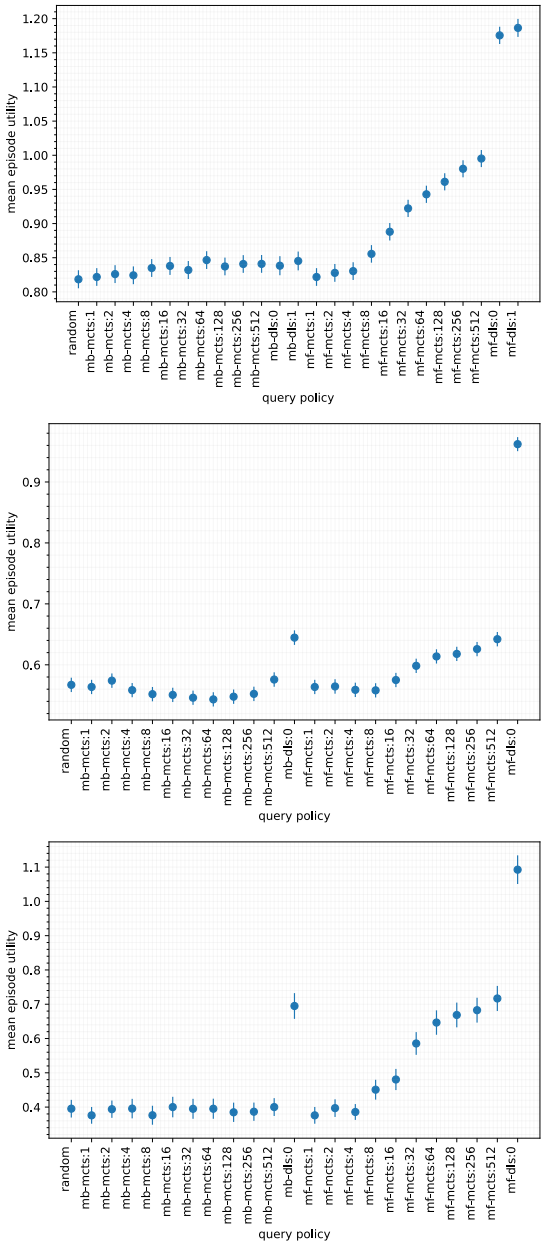

Figure 3: Mean episode utility of each query policy. Top to bottom: MovieLens 1M, MovieLens 25M, Amazon reviews.

Our results show that the model-free policies perform comparably or better than the model-based policies. Specifically, for DLS or MCTS with a given simulation budget, the model-free policy is usually both faster than its model-based counterpart and also attains a higher mean episode utility. We also observe that DLS outperforms MCTS with respect to utility, but scales poorly in terms of computational cost. In fact, DLS becomes infeasible beyond depth 0 (i.e., myopic selection), except with the MovieLens 1M dataset, which has the smallest number of queries. In settings where compute budgets are constrained and a large query space (like the full set of Movielens 25M tags) makes multi-step DLS intractable,

MCTS will generally outperform DLS. Furthermore, on devices where hardware parallelization is limited (e.g., on CPUs rather than GPUs), DLS cannot evaluate its search tree leaves in parallel, and must process them sequentially like MCTS does. DLS, even for single-step lookahead (that is, 0 depth), requires traversing all queries, and is therefore costly in terms of runtime when the query space is large. Depending on the simulation budget, MCTS might attain lower performance, but does not require traversing all queries, and can therefore potentially be faster, which puts it on the Pareto frontier of performance versus runtime cost.

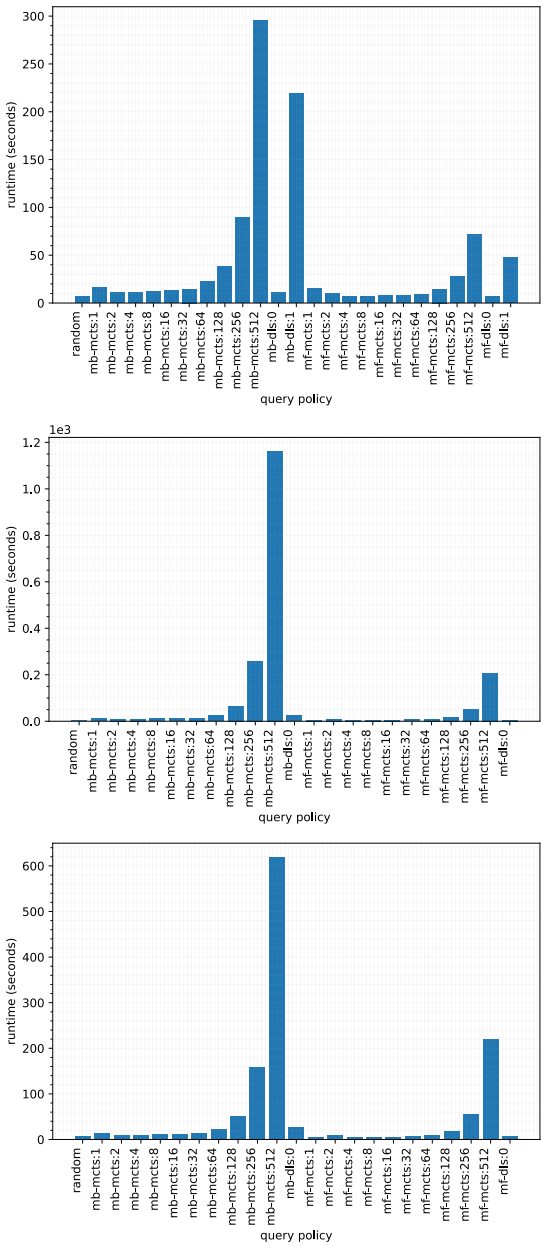

Figure 4: Runtime of each query policy. Top to bottom: MovieLens 1M, MovieLens 25M, Amazon reviews.

These results suggest that our model-free approach yields significant improvements in recommendation quality, as measured by the utility of the recommended items to users. The model-free methods also exhibit much lower runtimes than model-based techniques This is due to the fact that, while each response or utility prediction of the model-based approach requires performing Bayesian inference and sampling from the posterior, the model-free approach requires only a forward pass through a neural network.

9 Conclusions and Future Research

We have proposed a model-free approach to preference elicitation that avoids the simplistic, restrictive modeling assumptions typical of model-based methods, and instead leverages function approximation to learn the quantities needed for PE. We explored multiple network architectures and two (classes of) planning algorithms, demonstrating significant improvements in recommendation quality with respect to a natural baseline.

A summary of the process is as follows. We generated user, tag, and item embeddings from the dataset, then generated trajectories (queries, responses, utilities) given a query policy (in a real application, trajectories would be induced by a deployed policy interacting with users). Next, we ran a model to predict user responses and utilities, which was used inside planning algorithms (DLS and MCTS) to create a planning-based query policy. Finally, we evaluated this policy by measuring the mean item recommendation utility of newly-generated trajectories.

In future work, we plan to explore the use of more informed, but less diverse, query policies for generating trajectories for training. We also hope to analyze the relationship between the quality of predictive models and the performance of the induced query policies. Another important direction is training in an online fashion (as in AlphaZero and MuZero) with simulated users to learn more accurate value/policy functions. Finally, we hope to test an approach that first learns an environment dynamics model (user responses and item recommendation utilities) from offline data, and then applies *online* reinforcement learning against this learned model. We conjecture an *ensemble* of such models (or an uncertainty-aware model, such as a Bayesian neural network) will help prevent overfitting the policy to a learned model that is different from the true environment.

Acknowledgements

We thank Chih-wei Hsu and Yinlam Chow from Google Research for useful discussions. Tuomas Sandholm's and part of Carlos Martin's research is supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876, National Science Foundation grants RI-2312342 and RI-1901403, ARO award W911NF2210266, and NIH award A240108S001.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

Ali Abbas. Entropy methods for adaptive utility elicitation. *IEEE Transactions on Systems, Science and Cybernetics*, 2004.

Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Analyzing user modeling on Twitter for personalized news recommendations. In *ACM Conference on User Modeling, Adaptation and Personalization (UMAP)*, 2011.

Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K Hubert, and David Silver. Planning in stochastic environments with a learned model. In *International Conference on Learning Representations (ICLR)*, 2022.

Jimmy Lei Ba, Jamie Kiros, and Geoffrey Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, et al. The DeepMind JAX Ecosystem, 2020.

Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Jesús Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-based systems*, 2012.

Nadjet Bourdache, Patrice Perny, and Olivier Spanjaard. Incremental elicitation of rank-dependent aggregation functions based on Bayesian linear regression. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Craig Boutilier, Richard S Zemel, and Benjamin Marlin. Active collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

Craig Boutilier, Tuomas Sandholm, and Rob Shields. Eliciting bid taker non-price preferences in (combinatorial) auctions. In *AAAI Conference on Artificial Intelligence*, 2004.

Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 2006.

Craig Boutilier. A POMDP formulation of preference elicitation problems. In *AAAI Conference on Artificial Intelligence*, 2002.

Craig Boutilier. Computational decision support: Regret-based models for optimization and preference elicitation. In *Comparative Decision Making: Analysis and Support Across Disciplines and Applications*. Oxford University Press, 2013.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, et al. JAX: composable transformations of Python+NumPy programs, 2018.

Darius Braziunas and Craig Boutilier. Assessing regret-based preference elicitation with the UTPREF recommendation system. In *ACM Conference on Economics and Computation (EC)*, 2010.

Gregory Canal, Andy Massimino, Mark Davenport, and Christopher Rozell. Active embedding search via noisy paired comparisons. In *International Conference on Machine Learning (ICML)*, 2019.

Urszula Chajewska, Lise Getoor, Joseph Norman, and Yuval Shahar. Utility elicitation as a classification problem. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 1998.

Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *AAAI Conference on Artificial Intelligence*, 2000.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*, 2014.

Robert Cohn, Satinder Singh, and Edmund Durfee. Characterizing evoi-sufficient k-response query sets in decision problems. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Computers and Games*, 2007.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *ACM Conference on Recommender Systems (RecSys)*, 2016.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with Gumbel. In *International Conference on Learning Representations (ICLR)*, 2022.

Finale Doshi-Velez, Joelle Pineau, and Nicholas Roy. Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs. *Artificial Intelligence*, 2012.

Simon Duane, Anthony Kennedy, Brian Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 1987.

Jean-Philippe Dubus, Christophe Gonzales, and Patrice Perny. Multiobjective optimization using GAI models. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 1979.

Bradley Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association (JASA)*, 1987.

Michael Ekstrand, John Riedl, Joseph Konstan, et al. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 2011.

Gregory Farquhar, Tim Rocktaeschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreeC: Differentiable tree planning for deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 1975.

Arthur Guez, Theophane Weber, Ioannis Antonoglou, Karen Simonyan, et al. Learning to search with MCTSnets. In *International Conference on Machine Learning (ICML)*, 2018.

Arthur Guez, Mehdi Mirza, Karol Gregor, Rishabh Kabra, et al. An investigation of model-free planning. In *International Conference on Machine Learning (ICML)*, 2019.

Shengbo Guo and Scott Sanner. Real-time multiattribute Bayesian preference elicitation with pairwise comparison queries. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Blake Hallinan and Ted Striphas. Recommended for you: The Netflix prize and the production of algorithmic culture. *New Media & Society*, 2016.

F Maxwell Harper and Joseph Konstan. The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2016.

W Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 1970.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *ACM Web Conference*, 2017.

Jonathan Herlocker, Joseph Konstan, Loren Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 2004.

Hillary Holloway and Chelsea White III. Question selection for multiattribute decision-aiding. *European Journal of Operational Research (EJOR)*, 2003.

Ronald Howard and James Matheson, editors. *Readings on the Principles and Applications of Decision Analysis*. Strategic Decision Group, 1984.

Vijay Iyengar, Jon Lee, and Murray Campbell. Q-Eval: Evaluating multiple attribute items using queries. In *ACM Conference on Economics and Computation (EC)*, 2001.

Robert Kass, Luke Tierney, and Joseph Kadane. Laplace's method in bayesian analysis. *Contemporary Mathematics*, 1991.

Ralph Keeney and Howard Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, 1976.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Volodymyr Kuleshov and Kevin Ellis. Active preference inference using language models and probabilistic reasoning. *arXiv:2312.12009*, 2023.

Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *ACM International Conference on Ubiquitous Information Management and Communication (ICUIMC)*, 2008.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Distributed Systems Online*, 2003.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv:1908.03265*, 2019.

Ofer Meshi, Jon Feldman, Li Yang, Ben Scheetz, Yanli Cai, Mohammadhossein Bateni, Corbyn Salisbury, Vikram Aggarwal, and Craig Boutilier. Preference elicitation for music recommendations. In *International Conference on Machine Learning 2023 Workshop The Many Facets of Preference-Based Learning*, 2023.

Nicholas Metropolis, Arianna Rosenbluth, Marshall Rosenbluth, Augusta Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 1953.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv:1811.01900*, 2018.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing*, 2019.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Aditya Pal, Chantat Eksombatchai, Yitong Zhou, Bo Zhao, Charles Rosenberg, and Jure Leskovec. PinnerSage: Multi-modal user embedding framework for recommendations at Pinterest. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Sudha Rao and Hal Daumé III. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. *arXiv:1805.04655*, 2018.

Al Mamunur Rashid, George Karypis, and John Riedl. Learning preferences of new users in recommender systems: An information theoretic approach. *ACM SIGKDD Explorations Newsletter*, 2008.

Lior Rokach and Slava Kisilevich. Initial profile generation in recommender systems using pairwise comparison. *IEEE Transactions on Systems, Man, and Cybernetics*, 2012.

David Rumelhart, Geoffrey Hinton, and Ronald Williams. Learning representations by back-propagating errors. *Nature*, 1986.

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2007.

Ahti Salo and Raimo P Hämäläinen. Preference ratios in multiattribute evaluation (PRIME)—elicitation and decision procedures under incomplete information. *IEEE Transactions on Systems, Science and Cybernetics*, 2001.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recom-

mender system-a case study. Technical report, University of Minnesota, 2000.

J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *ACM Conference on Economics and Computation (EC)*, 1999.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 2020.

David Silver, Aja Huang, Chris Maddison, Arthur Guez, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

David Silver, Aja Huang, Chris Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, et al. Mastering the game of Go without human knowledge. *Nature*, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 2018.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

Olivier Toubia, Duncan I Simester, John R Hauser, and Ely Dahan. Fast polyhedral adaptive conjoint estimation. *Marketing Science*, 2003.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Phebe Vayanos, Yingxiao Ye, Duncan McElfresh, John Dickerson, and Eric Rice. Robust active preference elicitation. *arXiv:2003.01899*, 2020.

Ivan Vendrov, Tyler Lu, Qingqing Huang, and Craig Boutilier. Gradient-based optimization for Bayesian preference elicitation. In *AAAI Conference on Artificial Intelligence*, 2020.

Paolo Viappiani and Craig Boutilier. Optimal Bayesian recommendation sets and myopically optimal choice query sets. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2010.

Paul Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1988.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, et al. On layer normalization in the transformer architecture. In *International Conference on Learning Representations (ICLR)*, 2020.

Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H Chi. Mixed negative sampling for learning two-tower neural networks in recommendations. In *ACM Web Conference*, 2020.

Shudong Yang, Xueying Yu, and Ying Zhou. LSTM and GRU neural network performance comparison study: Taking yelp review dataset as an example. In *International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*, 2020.

Lili Yu, Howard Chen, Sida Wang, Tao Lei, and Yoav Artzi. Interactive classification by asking informative questions. *arXiv:1911.03598*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep Sets. *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Conference on Information and Knowledge Management (CIKM)*, 2013.

Zhibing Zhao, Haoming Li, Junming Wang, Jeffrey Kephart, Nicholas Mattei, Hui Su, and Lirong Xia. A cost-effective framework for preference elicitation and aggregation. *arXiv:1805.05287*, 2018.