SLATEQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets

Eugene Ie[1*†], **Vihan Jain**[1†], **Jing Wang**[1†],
Sanmit Narvekar[2‡], **Ritesh Agarwal**[1], **Rui Wu**[1], **Heng-Tze Cheng**[1],
Tushar Chandra[1] and **Craig Boutilier**[1*]
[1]Google Research, [2]Department of Computer Science, University of Texas at Austin
{eugeneie,cboutilier}@google.com

Abstract

Reinforcement learning (RL) methods for recommender systems optimize recommendations for long-term user engagement. However, since users are often presented with slates of multiple items—which may have interacting effects on user choice—methods are required to deal with the combinatorics of the RL action space. We develop SLATEQ, a decomposition of value-based temporal-difference and Q-learning that renders RL tractable with slates. Under mild assumptions on user choice behavior, we show that the long-term value (LTV) of a slate can be decomposed into a tractable function of its component item-wise LTVs. We demonstrate our methods in simulation, and validate the scalability and effectiveness of decomposed TD-learning on YouTube.

1 Introduction

Practical recommender systems largely focus on *myopic* prediction—estimating a user's *immediate* response to a recommendation—without considering the long-term impact on subsequent user behavior. This can be limiting: modeling a recommendation's stochastic impact on the future affords opportunities to trade off user engagement in the near-term for longer-term benefit (e.g., by probing a user's interests, or improving satisfaction). As a result, recommender systems research has increasingly turned to the sequential nature of user behavior using temporal models, such as hidden Markov models and recurrent neural networks [Rendle *et al.*, 2010; Wu *et al.*, 2017], and long-term planning using *reinforcement learning (RL)* techniques (e.g., [Gauci *et al.*, 2018; Choi *et al.*, 2018; Zhao *et al.*, 2018]). However, RL for recommendation has largely been confined to restricted domains due to the complexities of deploying such models at scale.

One challenge in many recommenders is that *multiple items are recommended to a user simultaneously*, sometimes called a recommendation *slate*. This induces an RL problem with a large combinatorial action space. Recent approaches to RL with such combinatorial actions [Sunehag *et al.*, 2015; Metz *et al.*, 2017] make inroads into this problem, but are unable to scale to problems of the size encountered in large, real-world recommenders, in part because of their generality. In this work, we develop a new *slate decomposition* technique called SLATEQ that estimates the *long-term value (LTV)* of a slate of items by directly using the estimated LTV of the *individual items on the slate*. This decomposition takes advantage of the specifics of user choice behavior, but makes minimal assumptions about user choice.

Specifically, we first show how the SLATEQ decomposition can be incorporated into *temporal difference (TD)* learning algorithms, such as SARSA and Q-learning, so that LTVs can be learned at the level of individual items despite the fact that items are always presented to users in slates. This is critical for both generalization and exploration efficiency. We then turn to optimization, i.e., constructing slates that maximize LTV, a required component of policy improvement (e.g., in Q-learning) at training time, and for selecting optimal slates at serving time. Despite the combinatorial (and fractional) nature of the underlying optimization problem, we show that it can be solved in polynomial-time by a two-step reduction to a linear program (LP). We also show that simple top-k and greedy approximations, while having no theoretical guarantees, work well in practice.

Finally, we demonstrate our approach with both offline simulation experiments and an online live experiment on the YouTube video recommendation system. We show that our techniques are scalable and offer significant improvements in user engagement over myopic recommendations. The live experiment also demonstrates how our methodology supports the relatively straightforward deployment of TD and RL methods that build on the learning infrastructure of extant myopic systems. Further details can be found in an expanded, related article [Ie *et al.*, 2019].

2 Related Work

Recommender systems have typically relied on collaborative filtering (CF) techniques [Konstan *et al.*, 1997; Breese *et al.*, 1998; Salakhutdinov and Mnih, 2007]. These exploit user feedback on a subset of items (either explicit, e.g., ratings, or implicit, e.g., consumption) to directly estimate user preferences for unseen items. Increasingly, recommenders have moved beyond explicit preference prediction to capture more nuanced aspects of user behavior, for instance, how they respond to specific recommendations, such

*Contact Authors
†Authors Contributed Equally
‡Work done while at Google LLC

as pCTR (predicted click-through rate), degree of engagement (e.g., dwell/watch/listen time), ratings, social behavior (e.g., comments, sharing), etc. [van den Oord *et al.*, 2013; Covington *et al.*, 2016; Cheng *et al.*, 2016].

Early attempts to formulate recommendation as an RL problem include an MDP model for shopping recommendation [Shani *et al.*, 2005] and Q-learning for page navigation [Taghipour *et al.*, 2007]), but were limited to very small-scale settings (100s of items, few thousands of users). More recently, biclustering has been combined with RL algorithms [Choi *et al.*, 2018], while several commercial applications are reported in [Gauci *et al.*, 2018; Chen *et al.*, 2019], the latter developing a scalable, off-policy policy-gradient approach (though it does not compute explict LTVs or model slate effects). Zhao *et al.* [2018] consider actor-critic-based RL in slate recommendation, tested in simulation on an e-commerce data set. While similar in motivation to our approach, it does not explicitly address action-space combinatorics.

Accounting for slates in recommenders is common [Deshpande and Karypis, 2004; Viappiani & Boutilier, 2010; Le and Lauw, 2017] and introduces interesting modeling questions and computational issues due to the combinatorics of slates themselves. Swaminathan *et al.* [2017] develop off-policy evaluation and optimization with inverse propensity scores for slate interactions, while Jiang *et al.* [2019] use of VAEs to model the item distribution and generate slates.

Constructing optimal recommendation slates generally depends on *user choice behavior*. *Choice modeling* is widely studied in econometrics, psychology, statistics, operations research and marketing [Louviere *et al.*, 2000]. Probably the most common models of user choice are the *multinomial logit (MNL)* model and its extensions. For instance, the *conditional logit (CL)* model is justified under specific independence and extreme value assumptions [McFadden, 1974]. The MNL and CL models are instances of a more general conditional choice format we use below, in which a user i selects item $j \in A$ with unnormalized probability $v(x_{ij})$, where v is some function of a user-item feature vector x_{ij}:

$$P(j|A) = \frac{v(x_{ij})}{\sum_{\ell \in A} v(x_{i\ell})}. \tag{1}$$

In the case of CL, $v(x_{ij}) = e^{\tau u(x_{ij})}$, where u is a utility function. Such models are used to capture consumer choice or other user behavior in various domains. *Cascade models* [Joachims, 2002; Craswell *et al.*, 2008] have proven popular as a means of explaining user browsing behavior through (ordered) lists of recommendations, search results, etc., and are especially effective at capturing positional bias.

Designing tractable RL approaches for *combinatorial actions*—of which slate recommendations are an example—is itself quite challenging. *Sequential DQN* [Metz *et al.*, 2017] decomposes k-dimensional actions into a sequence of atomic actions, but trades off the exponential size of the action space with a corresponding exponential increase in the size of the state space. *Slate MDPs* [Sunehag *et al.*, 2015] model slates of *primitive actions*, and use DQN to learn the value of item slates, and a greedy procedure to construct slates. However, these approaches either require that primitive actions can be executed in isolation (which is not feasible in slate recommenders), or that one maintains an explicit Q-function over slates (hence, failing to address the combinatorics of representation, exploration and generalization).

3 An MDP Model for Slate Recommendation

In this section, we develop a *Markov decision process (MDP)* model for content recommendation with *slates*. We consider a recommender charged with presenting a slate to a user, from which the user selects zero or more items for consumption (e.g., listening to selected music tracks). Once items are consumed, the user can return for additional slate recommendations or terminate the session. The user's response to an item may have multiple dimensions, e.g., degree of engagement (such as consumption time), quality of that engagement, subsequent engagement beyond the recommender's direct control, rating feedback, sharing behavior, etc. We treat degree of engagement as the reward without loss of generality.

Session optimization can be modeled as a MDP with states $\mathcal{S}$, actions $\mathcal{A}$, reward function R and transition kernel P, with discount factor γ. The key components are:

(i) States $\mathcal{S}$ reflect user state, capturing both (relatively static) user features (e.g., demographics, interests) and relevant user history or past behavior (e.g., past recommendations, items consumed, degree of engagement).

(ii) Actions $\mathcal{A}$ are possible recommendation slates. We assume a fixed catalog of items $\mathcal{I}$, so actions are subsets $A \subseteq \mathcal{I}$ s.t. $|A| = k$, where k is the slate size.

(iii) Transition probability $P(s'|s, A)$ reflects the probability of the state becoming s' when action A is taken at s.

(iv) Reward $R(s, A)$ is the expected reward of a slate, which measures the expected degree of user engagement with items on slate A.

A *(stationary, deterministic) policy* $\pi : \mathcal{S} \to \mathcal{A}$ dictates the action to be taken at any state. Its value function V^π and action-value (or Q-) function, Q^π are given by:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V^\pi(s'), \tag{2}$$

$$Q^\pi(s, A) = R(s, A) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, A) V^\pi(s'). \tag{3}$$

The optimal policy π^* maximizes expected value $V(s)$ uniformly over $\mathcal{S}$, and its value—the optimal value function V^*—is given by the fixed point of the Bellman equation:

$$V^*(s) = \max_{A \in \mathcal{A}} R(s, A) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, A) V^*(s'). \tag{4}$$

$$Q^*(s, A) = R(s, A) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, A) V^*(s'). \tag{5}$$

The optimal policy satisfies $\pi^*(s) = \arg\max_{A \in \mathcal{A}} Q^*(s, A)$.

When transition and reward models are provided, optimal policies and value functions can be computed using a variety of methods [Puterman, 1994], though generally these require approximation in large state/action problems. With sampled data, RL methods such as *TD-learning SARSA* and *Q-learning* can be used (see Sutton & Barto [1998] for an overview). Given observed transitions and rewards as training data of the form (s, A, r, s', A'), the Q-function is updated as one of (where $\alpha^{(t)}$ is a learning rate):

$$Q^{(t)}(s, A)$$
$$\leftarrow \alpha^{(t)}[r + \gamma Q^{(t-1)}(s', A')] + (1 - \alpha^{(t)}) Q^{(t-1)}(s, A); \quad (6)$$
$$\leftarrow \alpha^{(t)}[r + \max_{A'} \gamma Q^{(t-1)}(s', A')] + (1 - \alpha^{(t)}) Q^{(t-1)}(s, A). \quad (7)$$

SARSA, Eq. (6), is *on-policy* and estimates the value of the data generating policy π, i.e., the TD-prediction problem on state-action pairs.[1] However, if the policy has sufficient exploration or other forms of stochasticity (as is common in large recommenders), acting greedily w.r.t. Q^π, and using the data so-generated to train a new Q-function, will implement a policy improvement step, and with repetition will converge to the optimal Q-function. Q-learning, Eq. (7), is *off-policy* and directly estimates the optimal Q-function. Unlike SARSA, Q-learning requires that one compute optimal slates A' at training time, not just at serving time.

4 SlateQ: Slate Decomposition for RL

One key challenge in the formulation above is the combinatorial nature of the action space, consisting of all $\binom{|\mathcal{I}|}{k} \cdot k!$ (ordered) k-sets over $\mathcal{I}$. This poses two difficulties for RL. First, the sheer size of the action space makes sufficient *exploration* impractical; and *generalization* of Q-values across slates is challenging without some compressed representation. Second, is the combinatorial optimization problem of finding a slate with maximum Q-value. Without structural assumptions or approximations, this problem cannot meet the real-time latency requirements of production recommender systems (often on the order of tens of milliseconds).

We make two assumptions about the interplay between system dynamics and user choice behavior to develop SlateQ, a model that allows the Q-value of a slate to be *decomposed into a combination of the item-wise Q-values of its constituent items*. We first develop the decomposition itself, then show how it can make slate optimization tractable.

4.1 Slate Decomposition of Q-values

We treat selection of no item from the slate as the selection of the *null item* $\perp$, which is an (implicit) item on every slate. We first specify two assumptions about user choice behavior of items from slates that allow the SlateQ decomposition.

- **Single choice (SC):** A user consumes a *single* item from each slate (which may be the null item $\perp$).
- **Reward/transition dependence on selection (RTDS):** The realized reward (user engagement) $R(s, A)$ depends (perhaps stochastically) *only on the item* $i \in A$ *consumed by the user*. Similarly, the state transition $P(s'|s, A)$ depends only on the consumed $i \in A$.

Assumption **SC** implies that users select only singletons $B \subseteq A$ where $|B| = 1$. This hold in the conditional choice (including CL) and cascade models described in Sec. 2. While limiting in some settings, in our application (see Sec. 6), users consume one content item at a time. Returning to the slate for a second item is a separate event (with a new state).[2] The **RTDS** assumption is realistic in many recommenders, especially w.r.t. immediate reward. The transition assumption holds in recommenders where it is a user's *direct* interaction with items that drives user utility, overall satisfaction, new interests, etc. But it may be treated as a simplifying assumption in other recommenders where unconsumed slate impressions themselves create, say, future curiosity. **RTDS** allows us to express rewards and state transitions as follows:

$$R(s, A) = \sum_{i \in A} P(i|s, A) R(s, i), \quad (8)$$
$$P(s'|s, A) = \sum_{i \in A} P(i|s, A) P(s'|s, i). \quad (9)$$

Our decomposition of (on-policy) Q-functions for a fixed data-generating policy π relies on an *item-wise auxiliary function* $\overline{Q}^\pi(s, i)$, which represents the LTV of a user consuming an item i, i.e., the LTV of i conditional on it being clicked/selected. Under **RTDS**, this function is independent of the slate A from which i was selected. We define:

$$\overline{Q}^\pi(s, i) = R(s, i) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, i) V^\pi(s'). \quad (10)$$

We immediately have, using **SC**:

Proposition 1. $Q^\pi(s, A) = \sum_{i \in A} P(i|s, A) \overline{Q}^\pi(s, i)$.

Proof.

$$Q^\pi(s, A) = R(s, A) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, A) V^\pi(s') \quad (11)$$
$$= \sum_{i \in A} P(i|s, A) R(s, i)$$
$$+ \gamma \sum_{i \in A} P(i|s, A) \sum_{s' \in \mathcal{S}} P(s'|s, i) V^\pi(s') \quad (12)$$
$$= \sum_{i \in A} P(i|s, A) [R(s, i) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, i) V^\pi(s')]$$
$$= \sum_{i \in A} P(i|s, A) \overline{Q}^\pi(s, i). \quad (13)$$

Here Eq. (12) follows immediately from **SC** and **RTDS** and Eq. (13) follows from the definition of $\overline{Q}^\pi$. $\square$

This simple result gives a *complete decomposition* of slate Q-values into Q-values for individual items. Hence, the combinatorial challenges disappear if we can learn $\overline{Q}^\pi(s, i)$ using TD methods. Fortunately, a simple Q-update fits the bill. Given a consumed item i at s with observed reward r, a transition to s', and next slate $\pi(s') = A'$, we update $\overline{Q}^\pi$ as:

$$\overline{Q}^\pi(s, i) \leftarrow \alpha (r + \gamma \sum_{j \in A'} P(j|s', A') \overline{Q}^\pi(s', j)) \quad (14)$$
$$+ (1 - \alpha) \overline{Q}^\pi(s, i).$$

Notice that this update assumes knowledge of the user choice model $P(i|s, A)$. Models such as MNL, CL, and *cascade* are

[1]SARSA is often used to refer to the on-policy *control* method that includes making policy improvement steps. We use it simply to refer to the TD-method based on SARSA updates as in Eq. (6).

[2]Our decomposition can be extended readily to accurately model user selection of multiple items by assuming conditional independence of item-choice probabilities given A.

easily learned from user response data, independent of LTV. Indeed, most recommenders use models that predict *click-through rates (CTR)* for items while ignoring slate interactions. We can use the scores or logits of an existing pCTR model, v, as a proxy for relative appeal of items to the user in state s in any of our models.

Our decomposed SLATEQ update facilitates more compact Q-value models, *using items as action inputs rather than slates*. This in turn allows for greater generalization and data efficiency. Critically, while SLATEQ learns item-level Q-values, it can be shown to converge to the correct *slate Q-values* under standard assumptions:

Proposition 2. *Under standard assumptions on learning rate schedules and exploring starts [Sutton and Barto, 1998], and the assumptions on user choice probabilities, state transitions, and rewards stated in the text above,* SLATEQ*—using update (14) and definition of slate value (13)—will converge to the true slate Q-function $Q^\pi(s, A)$ and support greedy policy improvement of π.*

Proof. (Brief sketch.) Standard proofs of convergence for TD(0) apply directly for Q^π, except for the introduction of the direct *expectation* over user choices, i.e., $\sum_{j\in A'} P(j|s', A')$, rather than samples. But the explicit expectation does not impact convergence (see, e.g., analysis of *expected SARSA* [Van Seijen *et al.*, 2009]). W.r.t. exploration, if the choice model allows some item j to have $P(j|s, A) = 0$ for *any slate A with* $\pi(s) > 0$ in some state s, item j at s won't be sampled under π (this is problematic for learning Q^π, not estimating V^π). Thus exploration must account for the choice model (e.g., by sampling all slates, or by configuring exploratory slates that ensure each j is sampled). For most choice models, all items have non-zero choice probability, so standard action-level exploration conditions apply. □

Modifying Eq. (10) to obtain $\overline{Q}(s, i)$—the *optimal* (off-policy) conditional-on-click item-wise Q-function—requires only the replacement of $V^\pi(s')$ with $V^*(s')$. Likewise, extending the decomposed update Eq. (14) to full Q-learning needs only to introduce the usual maximization:

$$\begin{aligned}\overline{Q}(s,i) \leftarrow &\alpha(r+\gamma \max_{A'\in\mathcal{A}} \sum_{j\in A'} P(j|s',A')\overline{Q}(s',j)) \\ &+(1-\alpha)\overline{Q}(s,i).\end{aligned} \tag{15}$$

Analogs of Props. 1 and 2 easily extend to this case.

4.2 Slate Optimization of Q-values

We now address the combinatorial *LTV slate optimization problem*, i.e., finding a slate with maximum expected Q-value from $\mathcal{A}$, the space of all $\binom{|\mathcal{I}|}{k} \cdot k!$ possible (ordered) k-sets over $\mathcal{I}$, given a specific choice model. This is required during training with Q-learning, and serving time when executing the induced greedy policy w.r.t. any Q-function (including when engaging in policy improvement with SARSA). Here we focus on the general conditional choice model (Eq. (1), of which CL and MNL are special cases). In this model, the ordering of items in a slate does not impact choice probabilities, so the action space consists of $\binom{|\mathcal{I}|}{k}$ (unordered) k-sets.

Exact Optimization. We can formulate the *LTV slate optimization problem* as follows:

$$\max_{\substack{A\subseteq\mathcal{I}\\|A|=k}} \sum_{i\in A} P(i|s,A)\overline{Q}(s,i). \tag{16}$$

Under general conditional choice Eq. (1), including MNL and CL, we can express this as a *fractional mixed-integer program (MIP)*, with binary variables $x_i \in \{0,1\}$ for each item $i \in \mathcal{I}$ indicating whether i occurs in slate A:

$$\max \sum_{i\in\mathcal{I}} \frac{x_i v(s,i)\overline{Q}(s,i)}{v(s,\perp)+\sum_j x_j v(s,j)} \tag{17}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} x_i = k; \quad x_i \in \{0,1\},\ \forall i \in \mathcal{I}. \tag{18}$$

This is a simplified variant of a classic product-line (or assortment) optimization problem [Chen and Hausman, 2000; Schön, 2010]. It can be shown that the binary indicators can be relaxed (see, e.g., Chen & Hausman [2000]) to obtain the following fractional linear program (LP):

$$\max \sum_{i\in\mathcal{I}} \frac{x_i v(s,i)\overline{Q}(s,i)}{v(s,\perp)+\sum_j x_j v(s,j)} \tag{19}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} x_i = k; \quad 0 \le x_i \le 1,\ \forall i \in \mathcal{I}. \tag{20}$$

The constraint matrix in this relaxed problem is totally unimodular, so the optimal solution is integral and standard nonlinear optimization methods can be used. However, since it is a fractional LP, it is directly amenable to the Charnes-Cooper [1962] transformation and can be recast directly as a (non-fractional) LP. To do so, we introduce an additional variable t that (implicitly) represents the (inverse) choice weight of the selected items $t = (v(s,\perp) + \sum_j x_j v(s,j))^{-1}$, and auxiliary variables y_i that represent the products $x_i \cdot (v(s,\perp) + \sum_j x_j v(s,j))^{-1}$, giving the following LP:

$$\max \sum_i y_i v(s,i)\overline{Q}(s,i) \tag{21}$$

$$\text{s.t. } tv(s,\perp) + \sum_i y_i v(s,i) = 1 \tag{22}$$

$$t \ge 0; \quad \sum_i y_i \le kt. \tag{23}$$

The optimal solution $(\mathbf{y}^*, t^*)$ to this LP yields the optimal x_i assignment in the fractional LP Eq. (19) via $x_i = y_i^*/t^*$. This in turn gives the optimal slate in the original fractional MIP Eq. (17): item i is on the slate if $y_i^* > 0$. Hence:

Observation 3. *LTV slate optimization, Eq. (16), under the general conditional choice model, Eq. (1), and fixed slate size k, is solvable in polynomial-time in the number of items $|\mathcal{I}|$.*

Thus full Q-learning with slates using the SLATEQ decomposition imposes at most a small polytime overhead relative to item-wise Q-learning despite its combinatorial nature. Moreover, production systems typically restrict the items to be ranked using a separate retrieval policy, so the set of items in the LP is usually much smaller than the complete set $\mathcal{I}$.

Top-k and Greedy Optimization. While the exact maximization of slates under the conditional choice model can be accomplished in polytime using $\overline{Q}$ and the item-score function v, we may wish to avoid solving an LP at serving time. A natural heuristic for constructing a slate is to simply add the k items with the highest score: in this case, we insert items into the slate in decreasing order of their "unnormalized expected LTV" $v(s,i)\overline{Q}(s,i)$. We call this *top-k optimization*. This incurs only an $O(\log(\mathcal{I}))$ overhead relative to the $O(\mathcal{I})$ time required for maximization with item-wise Q-learning.

Top-k is limited since, when considering the item to add to the Lth slot (for $1 < L \leq k$), item scores are not updated to reflect the previous $L-1$ items already added to the slate. *Greedy optimization*, instead of scoring each item *ab initio*, updates item scores w.r.t. the current partial slate. Specifically, given $A' = \{i_{(1)}, \ldots i_{(L-1)}\}$ of size $L-1 < k$, the Lth item is that with maximum marginal value:

$$\arg\max_{i \notin A'} \frac{v(s,i)\overline{Q}(s,i) + \sum_{\ell<L} v(s,i_{(\ell)})\overline{Q}(s,i_{(\ell)})}{v(s,i) + v(s,\perp) + \sum_{\ell<L} v(s,i_{(\ell)})}.$$

We compare top-k and greedy optimizations with the LP solution in our offline simulation experiments below.

Under general conditional choice, neither top-k nor greedy find the optimal slate, per the following counterexample:

Item	Score ($v(s,i)$)	Q-value
Null	1	0
a	2	0.8
b_1, b_2	1	1

The null item is always on the slate. Items b_1, b_2 are identical w.r.t. their behavior. We have $V(\{a\}) = 1.6/3$, greater than $V(\{b_i\}) = 1/2$. Both top-k and greedy will place a on the slate first. However, $V(\{a, b_i\}) = 2.6/4$, whereas the optimal slate $\{b_1, b_2\}$ is valued at $2/3$. So for slate size $k = 2$, neither top-k nor greedy find the optimal slate.

It is not hard to show that the expected value of a slate, when viewed as a set function, is neither submodular nor monotone, which prevents the application of standard analyses of greedy algorithm [Nemhauser *et al.*, 1978; Feige, 1998]. Moreover, we can show that top-k can perform arbitrarily poorly in general.

Observation 4. *The approximation ratio of the top-k algorithm for slate construction is unbounded.*

The following example demonstrates this.

Item	Score ($v(s,i)$)	Q-value
$\perp$	ε	0
a	ε	1
b	1	ε

Suppose we have $k = 1$. Top-k scores item b higher than a, creating the slate with value $V(\{b\}) = \varepsilon/(1+\varepsilon)$, while the optimal slate has value $V(\{a\}) = 1/2$.

Algorithm Variants. With multiple slate optimization methods at our disposal, many variants of our RL algorithms exist depending on the optimization method used during training and serving. Given a trained SlateQ model, we can apply that model to *serve* users using either top-k, greedy or the LP-based optimal method to generate recommended slates. Below we use the designations TS, GS, or OS to denote these serving protocols, respectively. These designations apply equally to (off-policy) Q-learned models, (on-policy) SARSA models, and even (non-RL) *myopic* models.[3]

During Q-learning, slate optimization is also required at *training time* to compute the maximum successor-state Q-value (Eq. 15). This can also use either of the three optimization methods, which we designate by TT, GT, and OT, respectively. This designation is not applicable when training a myopic model or SARSA (since it is on-policy). Thus for example, QL-OT-OS (resp., QL-OT-TS) refers to Q-learning with optimal training and optimal (resp., top-k) serving, while MYOP-GS refers to myopic recommendation with greedy serving. In our experiments we consider two other baselines: Random, which recommends random slates from the feasible set; and *full-slate Q-learning (FSQ)*, which is a standard, non-decomposed Q-learning method that treats each slate *atomically (i.e., holistically)* as a single action. The latter is a useful baseline to test whether the SlateQ decomposition provides leverage for generalization and exploration.

Other Choice Models. SlateQ is not limited to the choice models (e.g., CL) used above; but the optimization problem varies with the specific model. We discuss additional models (e.g., cascade) in the expanded paper [Ie *et al.*, 2019].

5 Empirical Evaluation: Simulation

We assess the SlateQ decomposition in a large-scale, live traffic experiment in Sec. 6. In this section we evaluate the quality of SlateQ on simulated problems using some of the algorithms above. We construct a *simulation environment* since most public datasets (e.g., MovieLens [Harper & Konstan, 2016]) are point-wise, static, and not designed for evaluating multi-step user-recommender interactions. Simulation for evaluating RL is useful even when live experiments are viable, since evaluation on live traffic is expensive and can introduce uncontrollable confounding effects.

The simulated environment works as follows (please see the expanded, related article [Ie *et al.*, 2019] for a more detailed description). We have a set of *documents* D representing content items to be recommended, each reflecting a mixture of *topics* T. Each $d \in D$ is a topic vector $\mathbf{d} \in [0,1]^{|T|}$; in our experiments, each d has only a single topic $T(d)$. Documents are drawn from distribution P_D. Documents also have an (unobservable) *inherent quality* L_d, drawn randomly from $\mathcal{N}(\mu_{T(d)}, \sigma^2)$, where μ_t is a *topic-specific* mean quality for any $t \in T$ (hence topics vary in average quality). Users $u \in U$ are characterized in part by their interests in topics, ranging from -1 (completely uninterested) to 1 (fully interested), hence are represented by $\mathbf{u} \in [-1,1]^{|T|}$, drawn from prior P_U. User u's *interest in document* d is given by the dot product $I(u,d) = \mathbf{ud}$.

At each interaction, m *candidate* documents are drawn from P_D, from which a slate of size k is selected for recommendation. A user chooses one item from the slate using a simple conditional model per Eq. (1): d_i chosen with probability $I(u,d_i)/\sum_{j\leq k} I(u,d_j)$. A user's *satisfaction* $S(u,d)$

[3]A myopic model is equivalent to a Q-learned model with $\gamma = 0$.

with a selected/consumed d is a function $f(I(u,d), L_d)$ of u's interest and d's quality. We use a simple convex combination $S(u,d) = \alpha I(u,d) + (1-\alpha)L_d$.

Each u has an initial (unobservable) *budget* B_u of time to engage with content during a session. Each chosen d reduces u's budget, with a session ending once B_u reaches 0. The budget decreases by a fixed c (which can vary per item if desired) less a *bonus* that increases with $S(u,d)$; thus, more satisfying items decrease the time remaining in a session at a slower rate. When u consumes d, her interest in topic $T(d)$ is nudged stochastically, biased toward increasing her interest, but with some chance of it decreasing. Thus, this recommender faces a short-term/long-term tradeoff between nudging a user's interests toward topics that tend to have higher quality at the expense of short-term consumption of user budget. Our learning algorithms use the Dopamine framework [Castro *et al.*, 2018]. Each strategy is evaluated over 5000 simulated users for statistical significance. All results are within a 95% confidence interval.

Myopic vs. Non-myopic Recommendations: We first test the quality of *(non-myopic) LTV policies* learned using SLATEQ to optimize engagement ($\gamma = 1$), using a selection of the SLATEQ algorithms (SARSA vs. Q-learning, different slate optimizations for training/serving). We compare these to *myopic scoring (MYOP)* ($\gamma = 0$), which optimizes only for immediate reward, as well as a Random policy. The following table compares several key metrics:

Strategy	Avg. Return (%)	Avg. Quality (%)
Random	159.2	-0.5929
MYOP-TS	166.3 (4.46%)	-0.5428 (8.45%)
MYOP-GS	166.3 (4.46%)	-0.5475 (7.66%)
SARSA-TS	168.4 (5.78%)	-0.4908 (17.22%)
SARSA-GS	172.1 (8.10%)	-0.3876 (34.63%)
QL-TT-TS	168.4 (5.78%)	-0.4931 (16.83%)
QL-GT-GS	172.9 (8.61%)	-0.3772 (36.38%)
QL-OT-TS	169.0 (6.16%)	-0.4905 (17.27%)
QL-OT-GS	173.8 (9.17%)	-0.3408 (42.52%)
QL-OT-OS	174.6 (9.67%)	-0.3056 (48.46%)

The LTV methods (SARSA and Q-learning) using SLATEQ offer overall improvements in average return per user session (percentage improvements relative to Random are shown in parentheses). For instance, relative to the random baseline, QL-OT-GS provides a 105.6% greater improvement than MYOP-GS. The LTV methods all learn to recommend documents of much higher quality than MYOP. This results in a positive impact on overall session length, which explains the improved return per user. We also see that LP-based slate optimization during training (OT) provides improvements over top-k and greedy optimization (TT, GT) in Q-learning when comparing similar serving regimes (e.g., QL-OT-GS vs. QL-GT-GS, and QL-OT-TS vs. QL-TT-TS). Optimal serving (OS) also shows consistent improvement over top-k and greedy serving—and greedy serving (GS) improves significantly over top-k serving (TS)—when compared under the same training regime. We note that the combination of optimal training and top-k or greedy serving performs well, and is especially useful when serving latency constraints are tight, since optimal training is generally done offline.

Finally, optimizing using Q-learning gives better results than on-policy SARSA (i.e., one-step improvement) under comparable training and serving regimes. But SARSA itself has significantly higher returns than MYOP, demonstrating the value of on-policy RL for recommender systems. Indeed, repeatedly serving-then-training (with some exploration) using SARSA would implement natural, continual policy improvement. These results demonstrate, in this simple synthetic recommender system environment, that using RL to plan long-term interactions can provide significant value in terms of overall engagement.

SLATEQ vs. Holistic Optimization: Next we compare the quality of policies learned using the SLATEQ decomposition to FSQ, the non-decomposed Q-learning method that treats each slate atomically as a single action. We set $|T| = 20, m = 10, k = 3$ so that we can enumerate all $\binom{10}{3}$ slates for FSQ maximization. Note that the Q-function for FSQ requires representation of all $\binom{20}{3} = 1140$ slates as actions, which can impede both exploration and generalization. For SLATEQ we test only SARSA-GS. The following table shows our results:

	Avg. Return (%)	Avg. Quality (%)	Avg. pCTR
Random	160.6	-0.6097	0.5800
FSQ	164.2 (2.24%)	-0.5072 (16.81%)	0.5661 (-2.38%)
SARSA-GS	170.7 (6.29%)	-0.5340 (12.41%)	0.7078 (22.06%)

While FSQ, an off-policy Q-learning method, is guaranteed to converge to the optimal slate policy in theory with sufficient exploration, we see that, even when using an *on-policy method* like SARSA, SLATEQ methods perform significantly better, offering a 180% greater improvement over Random than FSQ (despite using no additional model iterations for policy improvement). This is due to the fact that FSQ must learn Q-values for 1140 distinct slates, making it difficult to explore and generalize.[4] These results demonstrate the considerable value of the SLATEQ decomposition.

Improved representations could help FSQ generalize better, but the approach is inherently unscalable, while SLATEQ suffers from no such limitations (see Sec. 6). Interestingly, FSQ does converge quickly to a policy that offers recommendations of greater average quality than SLATEQ, but fails to make a suitable tradeoff with user interest.

Robustness to User Choice: We test the robustness of SLATEQ to changes in the choice model. Instead of the assumed choice model above, users select items from the slate using a simple *cascade model* (e.g., [Joachims, 2002]), where items on the slate are inspected from top-to-bottom with a position-specific probability, and consumed with probability proportional to $I(u,d)$ if inspected. If not consumed, the next item is inspected, etc. Though users act in this fashion, SLATEQ is trained (and served) assuming the *original choice model*. The following table shows the results:

Strategy	Avg. Return (%)	Avg. Quality (%)
Random	159.9	-0.5976
MYOP-TS	163.6 (2.31%)	-0.5100 (14.66%)
SARSA-TS	166.8 (4.32%)	-0.4171 (30.20%)
QL-TT-TS	166.5 (4.13%)	-0.4227 (29.27%)
QL-OT-TS	167.5 (4.75%)	-0.3985 (33.32%)
QL-OT-OS	167.6 (4.82%)	-0.3903 (34.69%)

SLATEQ continues to outperform MYOP, even when the choice model does not accurately reflect the true environment, demonstrating its relative robustness. SLATEQ can be adapted to other choice models, but since any choice model

[4]FSQ also takes roughly 6X the training time of SLATEQ over the same number of events.

will be only an approximation of true user behavior, this form of robustness is important.

Notice that QL-TT and SARSA have inverted relative performance compared to the experiments above. This is due to the fact that Q-learning exploits the (incorrect) choice model to optimize during training, while SARSA, being on-policy, only uses the choice model to compute expectations at serving time. This suggests that an on-policy control method like SARSA (with continual policy improvement) may be more robust than Q-learning in some settings.

6 Empirical Evaluation: Live Experiments

We tested SLATEQ (SARSA-TS specifically) on YouTube with $O(10^9)$ users and $O(10^8)$ items in its corpus. The system is typical of practical recommenders (e.g., [Covington *et al.*, 2016]), with two main components. A *candidate generator* retrieves a small subset (hundreds) of items from a large corpus that best match a user context. A *ranker* scores/ranks candidates using a DNN with both user context and item features. It optimizes a combination of several objectives (e.g., clicks, expected engagement, etc.). The extant recommender's policy is *myopic*, scoring items for the slate using their *immediate* (predicted) expected engagement.

In our experiment, we replace the myopic engagement measure with an *LTV estimate* in the ranker's scoring function. We retain other predictions and incorporate them into candidate scoring as in the myopic model. Our non-myopic recommender maximizes *cumulative* expected engagement, with user trajectories capped at N days. Since homepage visits can be spaced arbitrarily in time, we use time-based rather than event-based discounting to handle credit assignment across large time gaps. Our model extends the myopic ranker to a multi-task feedforward deep network that learns $\overline{Q}(s,i)$, the predicted long-term engagement of item i (conditional on being clicked) in state s, as well as $v(s,i)$ (for pCTR computation). Our model uses the same state and item features as the myopic model to ensure a reliable comparison. The model is trained using TensorFlow in a distributed training setup [Abadi *et al.*, 2015] using stochastic gradient descent. We train on-policy over pairs of consecutive start page visits, with LTV labels computed using Eq. (14), and use top-k optimization for both training and serving—i.e., we test SARSA-TS. The existing myopic recommender (baseline) also builds slates similarly (i.e., MYOP-TS).

We experimented with live traffic for three weeks, treating a small, but statistically significant, fraction of users to recommendations generated by our LTV (SARSA-TS) model. The control is a highly-optimized production machine learning model that optimizes for immediate engagement (MYOP-TS). Fig. 1 shows the percentage increase in aggregate user engagement using LTV over the course of the experiment relative to the control, and indicates that our model outperformed the baseline on the key metric under consideration, consistently and significantly. Fig. 2 shows the change in distribution of cumulative engagement originating from items at different positions in the slate. The results show that the users under treatment have more engaging sessions (larger LTVs) from items ranked higher in the slate compared to users in the control group, which suggests that greedy slate optimization performs reasonably in this domain.

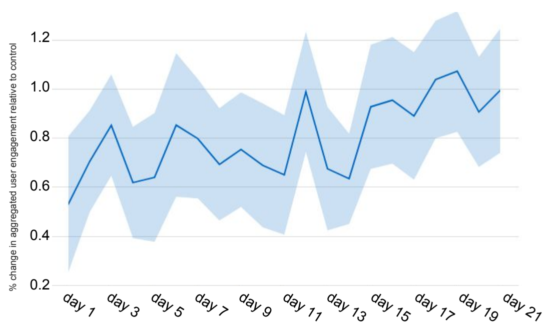

Figure 1: Increase in user engagement over the baseline. Data points are statistically significant and within 95% confidence intervals.

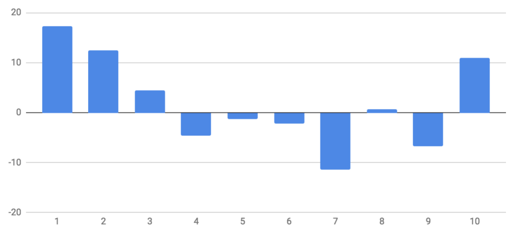

Figure 2: Percentage change in long-term user engagement vs. control (y-axis) across positions in the slate (x-axis). Top 3 positions account for approximately 95% of engagement.

7 Conclusion

We developed SLATEQ, a novel decomposition technique for slate-based RL that allows for effective TD and Q-learning using LTV estimates for individual items. It makes relatively innocuous assumptions, appropriate for many recommender settings, and supports tractable optimization. Our results show SLATEQ to be robust and scalable to large-scale commercial recommender systems like YouTube; they also demonstrate that LTV estimation can improve user engagement significantly in practice. There are a variety of future directions that suggest themselves. While SLATEQ makes only mild assumptions, our methodology can be extended by relaxing these further. Applying SLATEQ to other choice models [Ie *et al.*, 2019], further improving scalability, and releasing our simulation environment to the research community are ongoing directions.

Acknowledgments. Thanks to Larry Lansing for system optimization and the reviewers for helpful feedback.

References

[Abadi *et al.*, 2015] M. Abadi, A. Agarwal, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Breese *et al.*, 1998] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *UAI-98*, 43–52, Madison, WI, 1998.

[Castro *et al.*, 2018] P. Castro, S. Moitra, C. Gelada, S. Kumar, M. Bellemare. Dopamine: A research framework for deep reinforcement learning. `arXiv:1812.06110 [cs.LG]`, 2018.

[Charnes and Cooper, 1962] A. Charnes and W. Cooper. Programming with linear fractional functionals. *Naval Res. Log. Qrt.*, 9(3-4):181–186, 1962.

[Chen and Hausman, 2000] K. Chen and W. Hausman. Mathematical properties of the optimal product line selection problem using choice-based conjoint analysis. *Mgmt. Sci.*, 46(2):327–332, 2000.

[Chen *et al.*, 2019] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, E. Chi. Top-k off-policy correction for a REINFORCE recommender system. *WSDM-19*, 456–464.

[Cheng *et al.*, 2016] H. Cheng, L. Koc, et al. Wide & deep learning for recommender systems. *Deep Learn. for Rec. Sys. Workshop*, 7–10, Boston, 2016.

[Choi *et al.*, 2018] S. Choi, H. Ha, U. Hwang, C. Kim, J. Ha, and S. Yoon. Reinforcement learning-based recommender system using biclustering technique. `arXiv:1801.05532 [cs.IR]`, 2018.

[Covington *et al.*, 2016] P. Covington, J. Adams, and E. Sargin. Deep neural networks for YouTube recommendations. *RecSys-16*, 191–198, Boston, 2016.

[Craswell *et al.*, 2008] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. *WSDM-08*, 87–94, 2008.

[Deshpande and Karypis, 2004] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM TOIS*, 22(1):143–177, 2004.

[Feige, 1998] U. Feige. A threshold of ln(n) for approximating set cover. *JACM*, 45(4):634–652, 1998.

[Gauci *et al.*, 2018] J. Gauci, E. Conti, et al. Horizon: Facebook's open source applied reinforcement learning platform. `arXiv:1811.00260 [cs.LG]`, 2018.

[Harper & Konstan, 2016] F. Harper, J. Konstan. The MovieLens datasets: history & context. *ACM TIIS*, 5:1-19, 2016.

[Ie *et al.*, 2019] E. Ie, V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu, H.-Tze Cheng, M. Lustman, V. Gatto, P. Covington, J. McFadden, T. Chandra, and C. Boutilier. Reinforcement learning for slate-based recommender systems: A tractable decomposition and practical methodology. `arXiv:1905.12767 [cs.LG]`, 2019.

[Jiang *et al.*, 2019] R. Jiang, S. Gowal, T. Mann, and D. Rezende. Beyond greedy ranking: Slate optimization via list-CVAE. *ICLR-19*, New Orleans, 2019.

[Joachims, 2002] T. Joachims. Optimizing search engines using clickthrough data. *KDD-02*, 133–142, 2002.

[Konstan *et al.*, 1997] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *CACM*, 40:77–87, 1997.

[Le and Lauw, 2017] D. Le and H. Lauw. Indexable Bayesian personalized ranking for efficient top-k recommendation. *CIKM-17*, 1389–1398, 2017.

[Louviere *et al.*, 2000] J. Louviere, D. Hensher, and J. Swait. *Stated Choice Methods: Analysis and Application*. Cambridge Univ. Press, 2000.

[McFadden, 1974] D. McFadden. Conditional logit analysis of qualitative choice behavior. In P. Zarembka, ed., *Frontiers in Econometrics*, 105–142. Academic Press, 1974.

[Metz *et al.*, 2017] L. Metz, J. Ibarz, N. Jaitly, and J. Davidson. Discrete sequential prediction of continuous actions for deep RL. `arXiv:1705.05035 [cs.LG]`, 2017.

[Nemhauser *et al.*, 1978] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions. *Math. Prog.*, 14:265–294, 1978.

[Puterman, 1994] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1994.

[Rendle *et al.*, 2010] S. Rendle, C. Freudenthaler, L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. *WWW10*, 811-820.

[Salakhutdinov and Mnih, 2007] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *NIPS-07*, 1257–1264, Vancouver, 2007.

[Schön, 2010] C. Schön. On the optimal product line selection problem with price discrimination. *Mgmt. Sci.*, 56(5):896–902, 2010.

[Shani *et al.*, 2005] G. Shani, D. Heckerman, and R. Brafman. An MDP-based recommender system. *JMLR*, 6:1265–1295, 2005.

[Sunehag *et al.*, 2015] P. Sunehag, R. Evans, G. Dulac-Arnold, Y. Zwols, D. Visentin, and B. Coppin. Deep reinforcement learning with attention for slate Markov decision processes with high-dimensional states and actions. `arXiv:1512.01124 [cs.AI]`, 2015.

[Sutton and Barto, 1998] R. Sutton, A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[Swaminathan *et al.*, 2017] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni. Off-policy evaluation for slate recommendation. *NIPS-17*, 3632–3642, Long Beach, CA, 2017.

[Taghipour *et al.*, 2007] N. Taghipour, A. Kardan, S. Ghidary. Usage-based web recommendations: A reinforcement learning approach. *RecSys-07*, 113–120.

[van den Oord *et al.*, 2013] A. van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. *NIPS-13*, 2643–2651, Lake Tahoe, NV, 2013.

[Van Seijen *et al.*, 2009] H. Van Seijen, H. Van Hasselt, S. Whiteson, and M. Wiering. A theoretical and empirical analysis of expected SARSA. *IEEE Symp. Adap. Dyn. Prog. and RL*, 177–184, 2009.

[Viappiani & Boutilier, 2010] P. Viappiani, C. Boutilier. Optimal Bayesian recommendation sets and myopically optimal choice query sets. *NIPS*, 2352-2360, 2010.

[Wu *et al.*, 2017] C. Wu, A. Ahmed, A. Beutel, A. Smola, and H. Jing. Recurrent recommender networks. *WSDM-17*, 495–503, Cambridge, UK, 2017.

[Zhao *et al.*, 2018] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, J. Tang. Deep reinforcement learning for page-wise recommendations. *RecSys-18*, 95–103, Vancouver, 2018.