Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Selecting Sequences of Items via Submodular Maximization

Sebastian Tschiatschek
ETH Zurich
sebastian.tschiatschek@inf.ethz.ch

Adish Singla
ETH Zurich
adish.singla@inf.ethz.ch

Andreas Krause
ETH Zurich
krausea@ethz.ch

Abstract

Motivated by many real world applications such as recommendations in online shopping or entertainment, we consider the problem of selecting sequences of items. In this paper we introduce a novel class of utility functions over sequences of items, strictly generalizing the commonly used class of submodular set functions. We encode the sequential dependencies between items by a directed graph underlying the utility function. Classical algorithms fail to achieve any constant factor approximation guarantees on the problem of selecting sequences of bounded length with maximum utility. We propose an efficient algorithm for this problem that comes with strong theoretical guarantees characterized by the structural properties of the underlying graph. We demonstrate the effectiveness of our algorithm in synthetic and real world experiments on a movie recommendation dataset.

Introduction

A huge variety of applications involve the selection of sequences of items. Common to many of these applications is that the order of the selected items is not random but that items selected early, influence which items are selected later (Shani, Heckerman, and Brafman 2005). This is illustrated by the following three application scenarios: When online shopping for a new smartphone, users usually select the desired smartphone first and subsequently add corresponding accessories such as cases or headphones (McAuley, Pandey, and Leskovec 2015). When assembling (academic) reading lists, researchers very often work backwards in time, i.e. starting from recent relevant papers they trace back relevant lines of research to their origin (Shahaf, Guestrin, and Horvitz 2012). In entertainment, when watching videos, users typically consume all episodes of a series or if they watch a movie they particularly like, they may watch other movies by the same director produced in the future (Sarwar et al. 2001; Devooght and Bersini 2016).

Very often these sequential dependencies among items are neglected and the utility of a sequence of items is erroneously assumed to not depend on the order of the items, i.e. sequences of items are actually treated as sets of items. A class of widely applicable utility functions for sets of items that has gained much attention is the class of submodular set functions (Krause and Golovin 2014; Bilmes 2015). On the one hand, not considering the order of sequences has the advantages that the search space is exponentially smaller (in the length of the selected sequences) and that many algorithms enjoying strong theoretical guarantees exist, e.g. the greedy algorithm for submodular maximization under cardinality constraints (Nemhauser, Wolsey, and Fisher 1978). On the other hand, however, considering sets instead of sequences results in less expressive models and the inability to exploit the sequential dependencies.

Most existing algorithms cannot be readily applied to select sequences of items. Furthermore, certain tractable approaches for subset selection do not enjoy any theoretical guarantees if extended naively to select sequences, even in restricted scenarios (cf. Algorithms section).

Overview of our approach. We close this important gap by considering an expressive class of utility functions over sequences. This function class strictly generalizes the class of submodular utility functions and enables one to capture ordered preferences among items over arbitrarily long ranges. These ordered preferences are encoded by a directed acyclic graph over the items, where a directed edge between two items encodes that there is an additional utility when selecting the “tail" item before picking the “head" item. Furthermore, our model can capture diminishing returns between edges, i.e. the fore-mentioned additional utility can vary in the context of other edges. For the introduced function class, we propose a provably effective greedy algorithm for selecting sequences of items under sequence length constraints. In contrast to a naive item-based greedy algorithm, the proposed algorithm greedily selects edges in the graph and exploits graph-theoretic properties to determine an optimal sequence of items from a given set of edges.

Our main contributions include:

- We introduce a novel class of functions over sequences of items, strictly generalizing the commonly used class of submodular functions.
- We propose efficient algorithms for maximizing functions from that class and prove approximation guarantees for maximization under sequence length constraints.
- We demonstrate the effectiveness of our approach in synthetic and real world experiments.

Problem Statement

Sequences of items. Let $\mathcal{V} := \{v_1, v_2, \ldots, v_n\}$ be a set of n items a user can choose from. We are interested in selecting sequences of these items, i.e. the order in which the items are selected is important. Let $\Sigma := \{(\sigma_1, \sigma_2, \ldots, \sigma_k) \mid k \in [n], \sigma_1, \ldots, \sigma_k \in \mathcal{V}, \forall i, j \in [k] \colon i \neq j \Rightarrow \sigma_i \neq \sigma_j\}$ be the set of all possible sequences of items (without repetitions) that can be created from $\mathcal{V}$. By $|\sigma|$ we denote the length of the sequence $\sigma \in \Sigma$. For two sequences $\sigma, \pi \in \Sigma$ we denote their concatenation as $\sigma \oplus \pi$.

Ordered preferences and utility function. Commonly, when defining utility functions, the ordering of the items in σ is ignored and one simply assigns a utility to the set of items constituting σ. However, in many applications it is natural that the order of the items effects the actual utility, e.g. in entertainment as illustrated in Figure 1. In this paper, we model these ordered preferences by utilizing a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ whose vertices correspond to the items $\mathcal{V}$. Formally, we define the considered utility functions as

$$f(\sigma) = h(\text{edges}(\sigma)), \tag{1}$$

where $h(\cdot)$ is a set valued function described shortly and where $\text{edges}(\sigma) = \cup_{j \in [|\sigma|]} \{(\sigma_i, \sigma_j) \mid (\sigma_i, \sigma_j) \in \mathcal{E}, i \leq j\}$. The function $\text{edges}(\sigma)$ maps a sequence of items to a set of *active* edges according to $\mathcal{G}$ and σ, i.e. $\text{edges}(\sigma)$ consists of all edges in $\mathcal{G}$ from items appearing early in σ to items appearing later in σ. Informally, the self-cycle (v_i, v_i) in the graph $\mathcal{G}$ represents the utility of selecting an individual item v_i; and the edge (v_i, v_j) in the graph represents the *utility* in selecting item v_i before item v_j.

Submodularity assumption. We assume that $h(E)$ is a non-negative monotone submodular set function over the edges $\mathcal{E}$. Submodular set functions satisfy the *diminishing returns* property which guarantees that the marginal gain of an edge $e \in \mathcal{E}$ is larger in the context of some set of edges A compared to a larger set of edges $B \supseteq A$, i.e. formally $h(A \cup \{e\}) - h(A) \geq h(B \cup \{e\}) - h(B)$ for $A \subseteq B \subseteq \mathcal{E} \setminus \{e\}$. A set function $h(E)$ is *monotone* if $h(A) \leq h(B)$ for all $A \subseteq B$.

Sequences versus sets. Note that although $h(E)$ is a submodular set function over edges, the utility function $f(\sigma)$ we are interested in is *neither* a set function nor does it satisfy a diminishing returns property in general. To see this, consider the example from Figure 1 and assume that $h(E) = |E|$. Then the utilities assigned to the sequences $(B_1), (B_2), (B_1, B_2)$ and (B_2, B_1) are given as follows:

$$\begin{aligned} f((B_1)) &= h(\{(B_1, B_1)\}) = 1 \\ f((B_2)) &= h(\{(B_2, B_2)\}) = 1 \\ f((B_1, B_2)) &= h(\{(B_1, B_1), (B_2, B_2), (B_1, B_2)\}) = 3 \\ f((B_2, B_1)) &= h(\{(B_1, B_1), (B_2, B_2)\}) = 2 \end{aligned}$$

Note that the order of the items matters, i.e. $f((B_1, B_2)) \neq f((B_2, B_1))$, and that $f(\cdot)$ does not have diminishing returns, i.e. $f((B_2)) - f(()) \not\geq f((B_1, B_2)) - f(B_1))$.

Expressive power. The utility functions defined in (1) are strictly more expressive than submodular set functions over items. To see this, consider a graph $\mathcal{G}$ that only contains self-cycles and no other edges. Then $f(\sigma) = h(\text{edges}(\sigma)) = h(\{(\sigma_i, \sigma_i) \mid i \in [|\sigma|]\})$, i.e. $\text{edges}(\sigma)$ is a set of $|\sigma|$ items that does not depend on the ordering of the items in σ. Hence, any submodular set function over items $\mathcal{V}$ can be expressed in the form of (1).

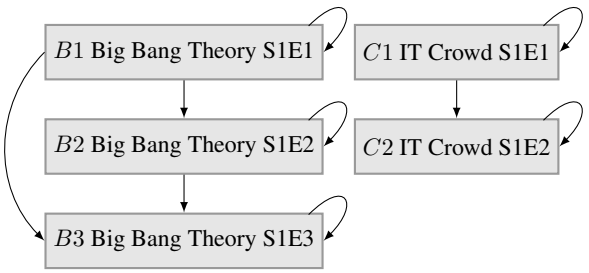


Figure 1: Ordered preferences for TV show recommendation. When watching TV shows, the order of consuming the shows affects the utility, e.g. watching *The Big Bang Theory (TBBT) Season 1 Episode 1* before watching *TBBT Season 1 Episode 2* has higher utility than watching these shows in the opposite order. Furthermore, if a user has already watched episode 1 and episode 2 of *TBBT Season 1*, the utility of watching episode 3 is higher than if that user has watched only episode 1 or 2. Similar preferences hold for the episodes of different series. Self-cycles indicate that there is a utility in selecting individual shows.

Optimization problem. Our objective is to select sequences of items of limited length which maximize the previously defined utility. This can be formalized as the following optimization problem,

$$\max_{\sigma \in \Sigma, |\sigma| \leq k} f(\sigma). \tag{P}$$

The solution to this problem depends on the graph $\mathcal{G}$ through the utility function $f(\sigma)$, and, as we will observe later, the key properties characterizing the performance of the proposed algorithms for solving (P) in the next section will be the indegree $\Delta_{\text{in}} := \max_{v_i \in \mathcal{V}} |\{v_j \in \mathcal{V} \mid (v_j, v_i) \in \mathcal{E}\}|$ and the outdegree $\Delta_{\text{out}} := \max_{v_i \in \mathcal{V}} |\{v_j \in \mathcal{V} \mid (v_i, v_j) \in \mathcal{E}\}|$.

Remarks. The search space for optimal solutions of problem (P) is much larger than in the case of unordered sets. In particular there are *only* $\binom{|\mathcal{V}|}{k}$ subsets of the items of size k but $k!\binom{|\mathcal{V}|}{k}$ sequences of items of length k. That is, the search space is exponentially larger in terms of k.

Algorithms

In this section we consider algorithms for solving problem (P). Firstly, we consider natural extensions of the greedy algorithm used for subset selection and show that these do not have any constant factor approximation guarantees, even for cases in which the graphs underlying the utility function satisfy $\Delta = \min\{\Delta_{\text{in}}, \Delta_{\text{out}}\} \leq c$, for some constant $c \in \mathbb{N}$. Secondly, we present our proposed algorithm for optimizing (P) which enjoys strong approximation guarantees and achieves good performance in synthetic and real world experiments, cf. Experiments section.

For notational convenience, we introduce the following: (i) for a given set of edges $E \subseteq \mathcal{E}$, the function nodes(E) returns the set of items covered by the edges in E; and (ii) for a given sequence of items $\sigma \in \Sigma$, the function nodes(σ) returns the set of items present in σ.

The proofs of all theorems are presented in the extended version of this paper (Tschiatschek, Singla, and Krause 2017).

Greedily Picking Items

For the case of non-negative monotone submodular utility functions over (unordered) sets, the classical greedy algorithm enjoys a $(1 - e^{-1})$ constant factor approximation ratio for maximization under cardinality constraints (Nemhauser, Wolsey, and Fisher 1978). Given this result, it is natural to consider the following naive extension of this greedy algorithm to our setting: The algorithm iteratively computes a sequence σ by greedily appending items which maximize the marginal gain in terms of the utility function (1) to the interim solution at the respective iteration. More generally, the algorithm can include a lookahead parameter l which allows the algorithm to expand the sequence σ by appending up to l items in every iteration, cf. Algorithm 1. We refer to this algorithm simply as GREEDY. For the case $l = 1$, a single item is appended to the sequence σ in every iteration, mimicking the greedy algorithm used for submodular maximization. For $l = 2$, sequences of up to length 2 can be appended in line 5 of the algorithm. While the complexity of the algorithm grows exponentially with l, the algorithm can exploit some of the ordered dependencies for $l \geq 2$.

In spite of the resemblance of the greedy algorithm for maximizing submodular functions, GREEDY does *not* enjoy similarly nice theoretical properties. This is substantiated in the following theorem.

Theorem 1. *There exist instances of problem* (P) *where the graph underlying the utility function is a DAG (excluding self-cycles) with bounded* $\Delta = \min\{\Delta_{\text{in}}, \Delta_{\text{out}}\}$ *for which* GREEDY *does not enjoy any constant factor approximation guarantee for any constant* l*.*

Algorithm 1 GREEDY: Node-based Greedy Algorithm

Require: Set of items $\mathcal{V}$, lookahead parameter l, maximum number of items k, utility function f

1: $\sigma \leftarrow ()$
2: **while** $|\sigma| < k$ **do**
3: $\quad \mathcal{C} \leftarrow \{\pi \in \Sigma \mid |\pi| \leq \min\{l, k - |\sigma|\}\}$
4: $\quad \pi^* \leftarrow \arg\max_{\pi \in \mathcal{C}} f(\sigma \oplus \pi)$
5: $\quad \sigma \leftarrow \sigma \oplus \pi^*$
6: **end while**
7: **return** σ

Greedily Picking Edges with Reordering

In this section, we propose our algorithm OMEGA (Ordered MAximization), a greedy algorithm applied to selecting edges in the graph $\mathcal{G}$ for solving problem (P), cf. Algorithm 2. OMEGA iteratively and greedily extends a set of edges E (and a corresponding sequence of items σ). The key observation for obtaining a provably efficient algorithm for our sequence selection problem is the following: If $\mathcal{G}$ is a DAG, then for every set of edges, we can (efficiently) compute an ordering of the items which are covered by the edges which maximizes the utility function (1) over all possible orderings of these items. In other words, by exploiting the DAG property of the graph underlying the utility function, we can compute for every set of items its optimal ordering. This reordering is implemented in the algorithm REORDER presented in Algorithm 3. The algorithm first computes a topological ordering of the graph $\mathcal{G}$ and then sorts the provided sets of items according to that order.

Algorithm 2 OMEGA: Edge-based Greedy Algorithm with Reordering

Require: Set of items $\mathcal{V}$, maximum number of items k, utility function f

1: $E \leftarrow \emptyset$
2: **while** $\exists e \in \mathcal{E} \setminus E$ s.t. $|\text{nodes}(E \cup \{e\})| \leq k$ **do**
3: $\quad \mathcal{C} \leftarrow \{e \in \mathcal{E} \setminus E \mid |\text{nodes}(E \cup \{e\})| \leq k\}$
4: $\quad e^* \leftarrow \arg\max_{e \in \mathcal{C}} f(\text{REORDER}(E \cup \{e\}))$
5: $\quad E \leftarrow E \cup \{e^*\}$
6: **end while**
7: $\sigma \leftarrow \text{REORDER}(E)$
8: **return** σ

Algorithm 3 REORDER: Compute Sequence of Items from Set of Edges

Require: set $A = \text{nodes}(E)$ to order

1: $\rho \leftarrow$ topological order of $\mathcal{G}$ {if $\mathcal{G}$ is not a DAG after excluding self-cycles, let π be a user specified or random order of the items}
2: $\sigma \leftarrow ()$
3: **for** i=1,...,$|\sigma|$ **do**
4: $\quad a^* \leftarrow \arg\min_{a \in A} \rho(a)$
5: $\quad \sigma \leftarrow \sigma \oplus (a^*)$
6: $\quad A \leftarrow A \setminus \{a^*\}$
7: **end for**
8: **return** σ

These ideas when applied to the greedy algorithm are sufficient to ensure a constant factor approximation guarantee for sequence selection:

Theorem 2. OMEGA *enjoys a* $(1 - e^{-\frac{1}{2\Delta}})$ *approximation guarantee for problem* (P)*, where the graph underlying the utility function is a DAG (excluding self-cycles) that satisfies* $\Delta = \min\{\Delta_{\text{in}}, \Delta_{\text{out}}\}$.

We can also easily obtain the following result for the special case that $\mathcal{G}$ only contains self-cycles:

Theorem 3. *If* $\mathcal{G}$ *contains only self-cycles,* OMEGA *enjoys a* $(1 - e^{-1})$ *approximation ratio, recovering the guarantee of the greedy algorithm for submodular maximization.*

For the special case that the utility function $h(\cdot)$ is modular, i.e. both $h(\cdot)$ and $-h(\cdot)$ are submodular, we obtain the following (improved) approximation guarantee.

Theorem 4. *For modular $h(\cdot)$,* OMEGA *enjoys a $\frac{1}{2\Delta}$ approximation ratio, where the graph is a DAG (excluding self-cycles) and $\Delta = \min\{\Delta_{\text{in}}, \Delta_{\text{out}}\}$.*

Note that a non-negative modular utility function $h(\cdot)$ does *not* imply a modular utility over the items, but rather has attractive properties in terms of the (ordered set of) items.

Remarks. If the graph $\mathcal{G}$ is not a DAG, our algorithm OMEGA can still be applied, although our theoretical guarantees do not hold. In this case, the algorithm REORDER uses either a random order of the items or must be supplied with an *approximate* order. Such an approximate order can for example be determined by computing a *feedback vertex set* of $\mathcal{G}$ (Karp 1972), i.e. a set of nodes whose removal results in a graph without cycles. Then, a topological ordering of $\mathcal{G}$ after removing the feedback vertex set can be computed. This ordering can finally be augmented by a random or user specified order over the items in the feedback vertex set.

Runtime Complexity. The computational complexity of GREEDY for picking items is $\mathcal{O}(kn^l)$, i.e. exponential in the lookahead. In contrast, our proposed algorithm OMEGA for picking edges with reordering has a runtime complexity of $\mathcal{O}((m+n) + k\Delta m(k \log k))$, where $m = |\mathcal{E}|$ is the number of edges in the graph. Here, the first term is the runtime complexity for computing a topological sort of the graph.[1] In the second term, the factor $k\Delta$ results from the fact that for a length constraint of k items, the algorithm can pick at most $k\Delta$ edges; and the factor $k \log k$ is the complexity for sorting, i.e. executing REORDER.

Experiments

Synthetic Experiments

In this section we present results on synthetic experiments demonstrating the superior performance of our algorithm over several natural baselines.

Modular utility over edges. As a first synthetic experiment, we evaluate the performance of OMEGA for instances where $h(E)$ is a modular function and where we vary the maximum outdegree Δ_{out} of the underlying graph $\mathcal{G}$. We created the graph $\mathcal{G}$ as follows: Let A be the adjacency matrix of $\mathcal{G}$, i.e. $A_{i,j} = 1$ if there is an edge from the ith item to the jth item. For every $i \in [n]$ we selected a subset of size $\min\{\Delta_{\text{out}}, n-i\}$ uniformly at random from $[i+1, n]$ and set the corresponding entries of A to 1. This construction ensures the desired maximum outdegree. We assigned every edge of $\mathcal{G}$ a utility independently drawn from the uniform distribution $\mathcal{U}([0,1])$. Furthermore, we assigned a weight $\mathcal{U}([0,1])$ to every edge $(i,i), i \in [n]$, i.e. we add self-cycles for every item. We then used these weights to define the modular function $h(\cdot)$. The objective value of solutions computed by OMEGA and certain baseline algorithms relative to the optimal solution (computed by exhaustive enumeration) are shown in Figure 2a for $n = 20$, $k = 6$ and $\Delta \in \{1, \ldots, 10\}$. As baselines we use random selection of k items (*random*) and GREEDY with lookahead parameters $l = 1$ (GREEDY $l = 1$) and $l = 2$ (GREEDY $l = 2$). For every setting of $n, k, \Delta_{\text{out}}$ we averaged the results over 50 problem instances. We can observe that our algorithm OMEGA performs best, achieving a performance close to optimal—almost independent of Δ_{out}. The performance of GREEDY degrades with increasing Δ_{out} for $l = 1$ and $l = 2$, in particular for small values of Δ_{out}.

Submodular utilities over edges. For this synthetic experiment we constructed the graph and the weights similar to before with the only difference that the weights of the self-cycles are drawn from $\mathcal{U}([0, 0.1])$. Furthermore, we use a probabilistic coverage function for $h(E)$, i.e.

$$h(E) = \sum_{j \in \text{nodes}(E)} \Big[1 - \prod_{(i,j) \in E} (1 - w_{i,j})\Big],$$

where $w_{i,j}$ is the weight associated with edge (i,j). Results are shown in Figure 2b. We observe, that OMEGA performs best again. With increasing Δ_{out}, all algorithms perform better, even random selection, indicating that the problems to solve become easier.

Real World Experiments

We performed real world movie recommendation experiments on the *Movielens 1M* dataset[2].

Dataset. This dataset contains 1,000,209 ratings of 6,040 users for 3,706 movies. Every rating takes a value in $\{1, \ldots, 5\}$ and has a timestamp. On average, every user rated 165.6 movies and every movie received 269.9 ratings. In our experiments, we do not seek to predict the rating values, but rather the sequence of movies watched by the user, i.e. $\mathcal{V}$ is the set of movies. Hence we view the data as a collection of (movie) sequences, one sequence σ^i for each user i, i.e. $\mathcal{D} = \{\sigma^1, \sigma^2, \ldots\}$. In particular, $\sigma^i = (\sigma^i_1, \sigma^i_2, \ldots)$, where σ^i_j is the jth movie rated by user i according to the timestamps of the ratings (ties are broken arbitrarily). We randomly partitioned the data $\mathcal{D}$ into training data $\mathcal{D}_{\text{train}}$ and testing data $\mathcal{D}_{\text{test}}$ such that $|\mathcal{D}_{\text{test}}| = 500$.

Recommendation task and evaluation. The task is to recommend movies to a test user i not present in the training data, given a few past ratings of that user. Formally, let $\sigma = (\sigma_1, \ldots, \sigma_m)$ be the sequence of movies rated by the test user. Then, given the first half of the movies rated by the user, i.e. $\sigma^{\text{pr}} = (\sigma_1, \ldots, \sigma_l)$ where $l = \lfloor m/2 \rfloor$, we want to make predictions about which other movies the user rated later, i.e. $\sigma^{\text{ft}} = (\sigma_{l+1}, \ldots, \sigma_m)$. Given some $k \in \mathbb{N}$, the precision when predicting k elements Prec@k is the number of correct predictions out of the k predictions averaged over the test data, i.e.

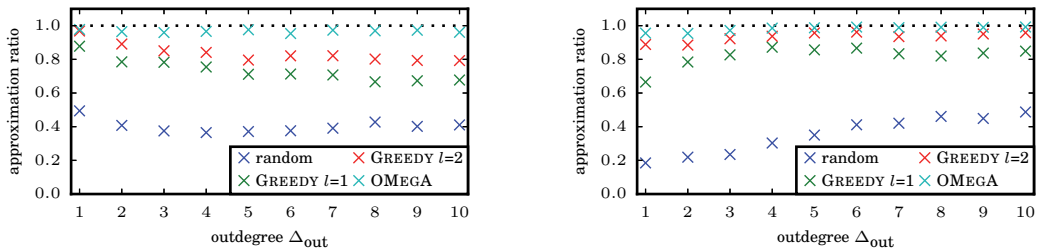
$$\text{Prec@k} := \frac{1}{k|\mathcal{D}_{\text{test}}|} \sum_{\sigma \in \mathcal{D}_{\text{test}}} |\text{nodes}(\sigma^{\text{ft}}) \cap P_k(\sigma^{\text{pr}})|, \quad (2)$$

where $P_k(\sigma^{\text{pr}})$ is a set of k predictions given the partial sequence of ratings σ^{pr} and all sequences σ^{ft} are assumed to be at least of size k.

Baselines. We compare our model with two commonly used baselines (Devooght and Bersini 2016). The models explained in the following are *test instance dependent* models,

[1] When executing OMEGA this has to be done only once at the beginning and not, as indicated for convenience of presentation, in every call to REORDER.

[2] http://grouplens.org/datasets/movielens/1m/



(a) Modular $h(\cdot)$ (b) Submodular $h(\cdot)$

Figure 2: Average performance of our algorithm on random instances for *(a)* modular and *(b)* submodular utility functions over the edges and varying maximum outdegree Δ_{out}. Our proposed Algorithm OMEGA achieves the best approximation ratio, almost independent of the oudegree Δ_{out}, outperforming various baseline algorithms. Error bars are omitted for legibility.

i.e. their construction depends on the given test instance σ. It turns out, that these baselines are special instances of (1). Hence, predictions $P_k(\sigma^{\text{pr}})$ from these models can be computed by executing a *conditional variant* of OMEGA, which is explained shortly.

The first baseline, FREQ, selects items based on their empirical frequencies estimated from the training data without accounting for item dependencies (Devooght and Bersini 2016; Tschiatschek, Djolonga, and Krause 2016). This baseline can be seen a special instance of the utility function (1), as illustrated in Figure 3a. In particular, $h(\cdot)$ is a modular function and the graph $\mathcal{G}$ contains only self-cycles for all items. The value associated with every self-cycle corresponds to the empirical frequency of the respective item.

The idea behind the second baseline, BG is to model the *transition probabilities* from the last item σ_l in σ^{pr} to unselected items (Devooght and Bersini 2016). Again, this is a specific instance of the utility function (1), as illustrated in Figure 3b. The utility function $h(\cdot)$ is again modular. There is an edge between item σ_l and another item $v \in \mathcal{V}$ if v is not in σ^{pr}. The value associated with that edge is the empirical transition probability from σ_l to v.

When estimating the empirical frequencies used in the baseline models by counting and subsequent normalization, we used a threshold of 10 to reduce noise, i.e. we set counts below 10 to 0.

Our model. In our model we use a probabilistic coverage utility function $h(\cdot)$ such that it combines and extends ideas from the two baseline models. Similar to the baseline models, our model, illustrated in Figure 3c, is test instance dependent. We model dependencies between the last z items in σ^{pr} and the items that can be selected, where we used $z \in \{1, 2, 5, \infty\}$ in our experiments ($z = \infty$ means that all items in σ^{pr} are considered). Our utility function takes the form

$$h(E) = \sum_{j \in \text{nodes}(E)} \Big[1 - \prod_{(i,j) \in E} (1 - p_{j|i})\Big],$$

where $p_{j|i}$ is the conditional probability that a user rates movie j given that she has rated movie i before. Note that our model includes self-cycles and associates the item frequency p_i (also denoted as $p_{i|i}$ for notational convenience) with the edge (i, i). Informally, the utility of a set of edges E is large if the nodes nodes(E) are well covered. We estimated the probabilities $p_{j|i}$ from the training data. We found that it is beneficial for this estimation, to account only instances in ranking histories for which the *distance* between items i and j in the sequence is below a certain limit (we limited this distance to 5 in our estimations). Predictions $P_k(\sigma^{\text{pr}})$ from our model are computed by executing a *conditional variant* of OMEGA. This variant iteratively grows the set of edges given that the set of nodes determined by reorder always starts with σ^{pr}, i.e. no edges linking back to the items in σ^{pr} must be selected.

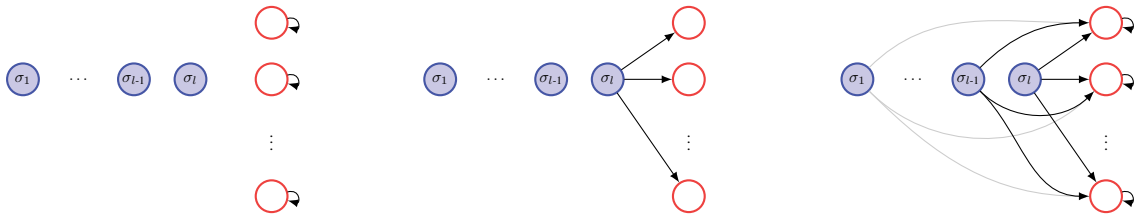
Table 1: Recommendation results for Movielens (Prec@k).

			OUR			
k	FREQ	BG	$z = 1$	$z = 2$	$z = 5$	$z = \infty$
1	0.28	0.36	0.36	0.37	0.39	0.40
2	0.27	0.34	0.34	0.36	0.37	0.38
3	0.25	0.32	0.32	0.35	0.37	0.37
4	0.25	0.32	0.31	0.34	0.35	0.36
5	0.25	0.31	0.31	0.33	0.35	0.35

Results. Recommendation results for our model and the baseline models are shown in Table 1 for varying length of predicted sequences k. For our model, we also varied the length of the users' history considered for constructing our model. We observe that our model outperforms the two baselines FREQ and BG in terms of precision for $z \geq 2$ for all considered k. For our model and fixed k, the prediction performance increases slightly with increasing z, indicating that considering a longer history of a user is beneficial.

Related Work

Submodular utility functions. Submodular set functions arise in numerous applications and their optimization has been studied extensively. A popular subclass is that of non-negative monotone submodular functions. For this subclass, the seminal work of Nemhauser, Wolsey, and Fisher (1978) shows that a simple greedy algorithm provides a constant factor approximation of $(1 - e^{-1})$ for maximization under cardinality constraints. Different generalizations have been considered recently, for example, maximizing non-monotone submodular functions (Buchbinder et al. 2014) and maxi-



(a) FREQ (b) BG (c) OUR

Figure 3: Illustration of the utility functions $h(\cdot)$ instantiated for the Movielens experiments. Solid nodes represent the items in $\sigma^{\text{pr}} = (\sigma_1, \ldots, \sigma_l)$, empty nodes represent items that are not in σ^{pr} and can be selected. *(a)* In FREQ, there is no interaction between the items and the utility of an item is encoded by the represented self-cycles. *(b)* In BG, the last item σ_l in σ^{pr} determines which items are selected next. Items that have been selected commonly after σ_l have higher utilities than items that were selected less frequently. This is encoded by the edge weights. *(c)* Our proposed model OUR combines properties of FREQ and BG. Furthermore, dependencies between all items (not only the last one) in σ^{pr} and items that can still be selected are encoded.

mization under noisy value oracles (Singla, Tschiatschek, and Krause 2016). The results have been extended to consider more general constraints such as graph connectivity constraints (Singh, Krause, and Kaiser 2009) to tackle the submodular orienteering / path planning problem (Chekuri and Pal 2005), and privacy-aware optimization (Singla et al. 2014; 2015). Recently, distributed algorithms for submodular optimization have been studied for large-scale problems (Mirzasoleiman et al. 2013). However, all these optimization problems and algorithms consider selecting sets instead of sequences and are unable to exploit the sequential information.

String and adaptive submodularity. A few extensions have been recently introduced to extend the expressive power of submodular utility functions. Zhang et al. (2016) and Alaei and Malekian (2010) study string or sequence submodular optimization. A function over sequences of items (*strings*) is string submodular (Zhang et al. 2016) if it satisfies the following *diminishing returns* property: for two sequences σ and π such that σ is a prefix for π, the gain of adding an item to σ is larger than that of adding it to the longer sequence π. For non-negative string submodular functions, a greedy algorithm similar to Algorithm 1 achieves a constant factor approximation ratio of $(1-e^{-1})$. Golovin and Krause (2011) introduced the notion of adaptive submodularity for solving stochastic optimization problems where the items are random variables associated with outcomes. The submodular set function is defined over the set of outcomes of the selected items. Their approach generalizes the problem of set selection to adaptive policies. A variant of the greedy algorithm achieves a constant factor approximation ratio of $(1-e^{-1})$ w.r.t. the optimal policy. The key limitation of these extensions is that they rely on some notion of diminishing returns, which is in general not satisfied by our class of utility functions.

Recommending sequences of items. Many real-world applications involve the recommendation of sequences of items. Shahaf and Guestrin (2010) study the problem of finding structured sequences of information in the domain of news articles. Specifically, given two news articles, their approach finds a coherent sequence of articles connecting the two given articles. Shahaf, Guestrin, and Horvitz (2012) further extend this work to find a sequence of articles which are coherent, diverse with high topic coverage and high connectivity in terms of conveying the underlying structure. However, the techniques developed in Shahaf and Guestrin (2010) and Shahaf, Guestrin, and Horvitz (2012) are very different from our paper — their work does not formalize the optimization problem of sequence selection by encoding the preferences via a graph as done in our model. Motivated by the application of sequential recommendations in online shoping portals such as Amazon, McAuley, Pandey, and Leskovec (2015) study the problem of inferring the underlying network of substitutable and complementary products. This work is complementary to ours: we assume access to the preference graph as well as the underlying utility function, and rather focus on the optimization problem of sequence selection.

Conclusion

We introduced a novel class of utility functions over sequences of items, thereby extending the expressive power of commonly used submodular set functions. We showed that the naive extensions of classical algorithms fail for our problem of selecting sequences of items of bounded length that maximize the utility. In fact, the search space for the optimal solutions for this new class of functions is exponentially larger than that of the classical subset selection problems. We developed a novel algorithm for sequence selection which takes into account the structural properties of the graph underlying the sequential preferences. Our theoretical analysis provides approximation guarantees for our algorithm w.r.t. the intractable optimal solution. We performed experiments on a movie recommendation dataset for the task of recommending sequences of movies to a user. We demonstrate that several existing baselines can be cast as a special instances of our model and show the effectiveness of our approach in terms of improved accuracy over these baselines.

Acknowledgements. This work was supported in part by the Swiss National Science Foundation, and Nano-Tera.ch program as part of the Opensense II project, ERC StG 307036, and a Microsoft Research Faculty Fellowship. Adish Singla acknowledges support by a Facebook Graduate Fellowship.

References

Alaei, S., and Malekian, A. 2010. Maximizing sequence-submodular functions and its application to online advertising. *arXiv preprint arXiv:1009.4153*.

Bilmes, J. 2015. Submodularity in machine learning applications. Twenty-Ninth Conference on Artificial Intelligence, AAAI-15 Tutorial Forum.

Buchbinder, N.; Feldman, M.; Naor, J.; and Schwartz, R. 2014. Submodular maximization with cardinality constraints. In *SODA*, 1433–1452.

Chekuri, C., and Pal, M. 2005. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, 245–253. IEEE.

Devooght, R., and Bersini, H. 2016. Collaborative filtering with recurrent neural networks. *arXiv preprint arXiv:1608.07400*.

Golovin, D., and Krause, A. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *JAIR* 42:427–486.

Karp, R. M. 1972. *Reducibility among Combinatorial Problems*. Springer US. 85–103.

Krause, A., and Golovin, D. 2014. Submodular function maximization. In *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press.

McAuley, J.; Pandey, R.; and Leskovec, J. 2015. Inferring networks of substitutable and complementary products. In *KDD*, 785–794.

Mirzasoleiman, B.; Karbasi, A.; Sarkar, R.; and Krause, A. 2013. Distributed submodular maximization: Identifying representative elements in massive data. In *NIPS*.

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

Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*, 285–295.

Shahaf, D., and Guestrin, C. 2010. Connecting the dots between news articles. In *KDD*, 623–632.

Shahaf, D.; Guestrin, C.; and Horvitz, E. 2012. Metro maps of science. In *KDD*, 1122–1130.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An mdp-based recommender system. *Journal of Machine Learning Research* 6(Sep):1265–1295.

Singh, A.; Krause, A.; and Kaiser, W. J. 2009. Nonmyopic adaptive informative path planning for multiple robots. In *IJCAI*, 1843–1850.

Singla, A.; Horvitz, E.; Kamar, E.; and White, R. W. 2014. Stochastic privacy. In *AAAI*.

Singla, A.; Horvitz, E.; Kohli, P.; White, R.; and Krause, A. 2015. Information gathering in networks via active exploration. In *IJCAI*.

Singla, A.; Tschiatschek, S.; and Krause, A. 2016. Noisy submodular maximization via adaptive sampling with applications to crowdsourced image collection summarization. In *AAAI*.

Tschiatschek, S.; Djolonga, J.; and Krause, A. 2016. Learning probabilistic submodular diversity models via noise contrastive estimation. In *AISTATS*.

Tschiatschek, S.; Singla, A.; and Krause, A. 2017. Selecting sequences of items via submodular maximization (extended version).

Zhang, Z.; Chong, E. K.; Pezeshki, A.; and Moran, W. 2016. String submodular functions with curvature constraints. *IEEE Transactions on Automatic Control* 61(3):601–616.