Learning Socially Optimal Information Systems from Egoistic Users

Karthik Raman and Thorsten Joachims

Department of Computer Science,
Cornell University, Ithaca, NY, USA
{karthik,tj}@cs.cornell.edu
http://www.cs.cornell.edu

Abstract. Many information systems aim to present results that maximize the collective satisfaction of the user population. The product search of an online store, for example, needs to present an appropriately diverse set of products to best satisfy the different tastes and needs of its user population. To address this problem, we propose two algorithms that can exploit observable user actions (e.g. clicks) to learn how to compose diverse sets (and rankings) that optimize expected utility over a distribution of utility functions. A key challenge is that individual users evaluate and act according to their own utility function, but that the system aims to optimize collective satisfaction. We characterize the behavior of our algorithms by providing upper bounds on the social regret for a class of submodular utility functions in the coactive learning model. Furthermore, we empirically demonstrate the efficacy and robustness of the proposed algorithms for the problem of search result diversification.

Keywords: Online Learning, Coactive Learning, Implicit Feedback, Diversified Retrieval.

1 Introduction

Many information systems serve a diverse population of users who have conflicting preferences. This poses the challenge of maximizing collective user satisfaction over a distribution of conflicting needs. A typical example is the problem of search result diversification [1]. For an ambiguous query such as "jaguar", a diversified set of results should ideally provide some relevant results for each of the query intents. Similar challenges also arise in an online store that wants to appeal to a range of customers with different tastes, or in a movie recommendation system where even a single user may have different preferences (e.g. moods, viewing companions) on different days. More generally, "diversification" describes the problem of hedging against uncertainty about a user's preferences.

Much prior work on this problem has focused on manually-tuned methods for generating diverse results [2–6]. Some learning approaches exist as well and have been shown to outperform manually tuned methods [7–10]. Unfortunately, the practical use of those learning methods is limited, since most require expert

H. Blockeel et al. (Eds.): ECML PKDD 2013, Part II, LNAI 8189, pp. 128–144, 2013.

annotated training data that explicitly lists all facets of an information need (e.g. the different moods a user can be in).

The use of implicit feedback from user interaction (e.g. clicks) has the potential to overcome this data bottleneck. Not only is it available in abundance, but it also directly reflects the users' – not the experts' – preferences. The challenge, however, is that the learning algorithm no longer gets (expert constructed) examples of socially optimal results, but needs to construct a socially optimal compromise from the egoistic actions of the users. Some learning methods for this problem already exist, but these methods either cannot generalize across queries [11] or are specific to a particular notion of user utility that cannot be adjusted through learning [12].

In this paper we consider the problem of learning socially optimal rankings from egoistic user feedback in the following online learning model. In each iteration a user, drawn from an unknown but fixed distribution, presents a context (e.g., query) to the system and receives a ranking in response. The user is represented by a utility function that determines the actions (e.g. clicks) and therefore the feedback to the learning algorithm. The same utility function also determines the value of the presented ranking. We focus on utility functions that are submodular in the elements of the ranking, since those naturally lead to diverse result sets. The goal is to learn a ranking function that has high social utility, which is the expected utility over the user distribution.

For this model, we present two coactive learning algorithms that learn to compose rankings that optimize social utility. Note that this setup is fundamentally different from previous coactive learning problems [13–15], which assume that user actions always come from the same utility function. After characterizing the informativeness and noisiness of the implicit feedback, we give theoretical results bounding the regret of our algorithms in terms of the social utility. Furthermore, we empirically show that the algorithms perform well for both single query and cross-query diversification tasks. In particular, we show that the algorithms can robustly learn, using only implicit feedback, to compose rankings with an appropriate amount of diversity.

2 Related Work

Coactive learning [13] is a framework for modeling the interaction between users and a learning algorithm, where the user feedback is interpreted as a revealed preference from a boundedly rational user. Recently, coactive learning [14] has been applied to the problem of *intrinsic diversity*. As opposed to our problem (*i.e.,* extrinsic diversity [1]) intrinsic diversity is diversity required by a single individual among their various different interests. More specifically, in their problem there is only a single user utility, based on which feedback is received. However in our problem, users have different utilities, which may conflict, and the goal of the system is finding a socially optimal solution.

Yue and Guestrin [16] also proposed online learning algorithms for the problem of intrinsic diversity. While they too maximize submodular utilities, their model

relies on observing cardinal utilities which can be substantially less reliable than preference feedback, as shown in user studies [17]. El-Arini and Guestrin [18] also propose submodularity-based techniques to optimize for both diversity and relevance in the context of scientific literature discovery. However, their model is motivated by *exploration* diversity that hedges against uncertainty about a single utility, while we optimize social utility over a distribution of utility functions.

Our work also relates to a large body of work in the game theory literature on finding social optimally solutions, such as work on maximizing welfare in congestion games [19, 20], auctions [21, 22] and social choices [23]. However, to the best of our knowledge, there has been no work on learning socially optimal rankings from noisy user feedback. While coactive learning is related to partial monitoring games [24], here the loss and feedback matrices are not revealed to the learning algorithm. Furthermore partial monitoring games have no explicit notion of context that is available at the beginning of each round.

3 Learning Problem and Model

Let's start with an example to motivate the formalization of the learning problem considered in this paper. Suppose we have a search engine that receives an ambiguous query (e.g. "jaguar"). In particular, there are three user populations that consider different documents relevant to the query as detailed in Fig. 1. The user populations have different sizes, and Fig. 1 lists the probability of each type. Note that the search engine has no way of identifying which type of user issued the query (*i.e.,* the search engine does not know whether "jaguar" refers to the cat or the car for this user). Suppose the utility of a ranking R to users of type i is $U_i(R) = \sqrt{\#\text{of rel docs in top 4 of R}}$. This means it is beneficial to show at least one relevant document, and that the marginal utility of showing additional relevant documents is sub-linear.

User Type	Prob.	Relevant docs
1	0.5	$a_1, a_2, a_3, \ldots$
2	0.25	$b_1, b_2, b_3, \ldots$
3	0.25	$c_1, c_2, c_3, \ldots$

Fig. 1. Illustrative example showing different user preferences

Now consider the following two rankings that the search engine could show.

- $R_1 = (a_1, a_2, a_3, a_4)$: While ideal for the predominant users (*i.e.,* type 1 users get utility $U_1 = 2$), it provides no value for the other users (utility $U_2 = U_3 = 0$). Thus in expectation, this ranking has expected utility of $\mathbf{E}[U] = 1$.
- $R_2 = (a_1, b_1, c_1, a_2)$: This ranking provides some relevant documents for all user types ($U_1 \sim 1.4; U_2 = 1; U_3 = 1$), maximizing the collective user satisfaction with $\mathbf{E}[U] \sim 1.2$.

Our goal is to find rankings of the latter type, which we call *socially optimal* since they maximize expected utility (*i.e., social utility*).

In this paper we use *implicit feedback* from the users for learning these rankings. Consider, for example, a user of type 1 that chooses to click/read relevant documents a_1, a_2 from the *presented ranking* $\mathbf{y}_t = (b_1, c_1, b_2, a_1, c_2, a_2)$. These

actions reveal information about the user's utility functions which we can exploit to construct a *feedback ranking* $\bar{\mathbf{y}}_t$, such as $(b_1, c_1, a_1, b_2, a_2, c_2)$, that has higher utility for that user (or at least not worse utility) *i.e.,* $U_1(\bar{\mathbf{y}}_t) \geq U_1(\mathbf{y}_t)$.

The key challenge in learning socially optimal rankings from the feedback of individual users lies in resolving the contradicting feedback from different user types. Each user's feedback reflects only their own utility, not social utility. For example, even if presented with the socially optimal ranking R_2, users may provide feedback indicating a preference for a different ranking (e.g. type 1 users may indicate their preference for R_1). Thus, a successful learning algorithm for this problem should be able to reconcile such differences in preference and display stability despite the *egoistic* feedback, especially when the presented solution approaches the optimal.

3.1 Learning Problem

We now define the learning problem and user-interaction model more formally. We assume there are N types of users, each associated with a probability p_i according to which individual users accessing the system are sampled. Given a context $\mathbf{x}_t$ (e.g., query), the **personal utility** of an object (*e.g.,* ranking) $\mathbf{y}_t$ for users of type i is $U_i(\mathbf{x}_t, \mathbf{y}_t)$. The **social utility** $U(\mathbf{x}_t, \mathbf{y}_t)$ is defined as the expected utility over the user distribution.

$$U(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{E}[U_i(\mathbf{x}_t, \mathbf{y}_t)] = \sum_{i=1}^{N} p_i U_i(\mathbf{x}_t, \mathbf{y}_t) \tag{1}$$

The optimal object for context $\mathbf{x}_t$ and user type i is denoted as

$$\mathbf{y}_t^{*,i} := \arg\max_{\mathbf{y}_t \in \mathcal{Y}} U_i(\mathbf{x}_t, \mathbf{y}_t). \tag{2}$$

The socially optimal object for context $\mathbf{x}_t$ is denoted as

$$\mathbf{y}_t^{*} := \arg\max_{\mathbf{y}_t \in \mathcal{Y}} U(\mathbf{x}_t, \mathbf{y}_t). \tag{3}$$

Users interact with the system like in the standard coactive learning model [13], but it is no longer assumed that all users act according to a single utility function. Specifically, at each timestep t the system receives a context $\mathbf{x}_t$ and a user type i is sampled from the user distribution. In response, the system presents the user with an object $\mathbf{y}_t$ and the user draws utility $U_i(\mathbf{x}_t, \mathbf{y}_t)$. The algorithm then observes (implicit) feedback from the user (who acts according to U_i), updates its model, and repeats. The goal of the algorithm is to present objects as close to the social optimal $\mathbf{y}_t^*$, as measured by the following notion of regret over time steps t of the learning process:

$$REG_T := \frac{1}{T} \sum_{t=0}^{T-1} \left(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t) \right). \tag{4}$$

Thus the lower the regret, the better the performance of the algorithm. Note that the social optimal $\mathbf{y}_t^*$ is never given to the learning algorithms, but nevertheless used to measure predictive performance.

To be able to prove anything about the regret of any learning algorithm, we need to make an assumption about the quality of the user feedback. Coactive learning assumes that the user feedback reveals an object $\bar{\mathbf{y}}_t$ that has improved utility compared to the presented object $\mathbf{y}_t$. In a ranked retrieval system, for example, $\bar{\mathbf{y}}_t$ can be constructed from $\mathbf{y}_t$ by moving the clicked documents to the top. Unlike in the traditional coactive learning model, this paper studies the case where users do not provide feedback from a single global utility function that directly reflects social utility. Instead, users valuate and provide feedback according to their own personal utility. Thus, we characterize feedback quality through the following definition.

Definition 1. *User feedback is* **expected** α_i, δ_i**-informative** *for a presented object* $\mathbf{y}_t$ *under context* $\mathbf{x}_t$ *for a user with personal utility function* U_i*, if* $\bar{\xi}_t \in \Re$ *is chosen such that for some given* $\alpha_i \in]0, 1]$ *and* $\delta_i > 0$ *it holds that*

$$\mathbf{E}_{\bar{\mathbf{y}}_t}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t)] \geq (1 + \delta_i)U_i(\mathbf{x}_t, \mathbf{y}_t) + \alpha_i \left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t)\right) - \bar{\xi}_t. \quad (5)$$

Note that the expectation is over the user feedback.

The expected α_i, δ_i-informative criterion states that the user's feedback object $\bar{\mathbf{y}}_t$ has better personal utility than the presented object $\mathbf{y}_t$ on average. More precisely, the first term on the right-hand side implies that the improvement should be at least by a factor of $(1 + \delta_i)$. Note, though, that this condition is based only on the personal utility of the specific user, not the social utility. The second term on the right-hand side further prescribes that personal utility increases proportional to how far $\mathbf{y}_t$ is away from the optimal object $\mathbf{y}_t^{*,i}$, and the factor $\alpha_i \in [0, 1]$ describes the informativeness of the feedback. This second term captures that it is easier to make large improvements in utility when the presented $\mathbf{y}_t$ is far from optimal for this user. Finally, since it would be unreasonable to assume that user feedback is always strictly α_i, δ_i-informative, the $\bar{\xi}_t$ captures the amount of violation.

3.2 Submodular Utility Model

The following defines the class of utility function we consider for modeling users. As done in previous work [14], we assume that the utility functions $U_i(\mathbf{x}_t, \mathbf{y}_t)$ is linear in its parameters $\mathbf{v}_i \in \mathbf{R}^m$.

$$U_i(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{v}_i^\top \phi_F(\mathbf{x}_t, \mathbf{y}_t) \quad (6)$$

$\phi_F(\mathbf{x}_t, \mathbf{y}_t)$ is a feature vector representation of the context-object pair and F is a submodular function as further elaborated on below. We require that all $\mathbf{v}_i$'s and $\phi_F(\mathbf{x}_t, \mathbf{y}_t)$'s are component-wise non-negative. The linear model implies that one can write the *social utility* as

$$U(\mathbf{x}_t, \mathbf{y}_t) = \mathbf{w}_*^\top \phi_F(\mathbf{x}_t, \mathbf{y}_t), \text{ where } \mathbf{w}_* = \sum_{i=1}^{N} p_i \mathbf{v}_i. \quad (7)$$

Algorithm 1. GreedyRanking($\mathbf{w}, \mathbf{x}$)

```
y ← 0
for i = 1 to k do
    bestU ← −∞
    for all d ∈ x/ y do
        if w^⊤(x, y ⊕ d) > bestU then
            bestU ← w^⊤ φ(x, y ⊕ d)
            best ← d
    y ← y ⊕ best                    {Append document to ranking y}
return y
```

We model $\phi_F(\mathbf{x}_t, \mathbf{y}_t)$ using a submodular aggregation of its components, which is a well accepted method for modeling diversity [9, 14]. To simplify the exposition, we focus on rankings as objects $\mathbf{y}$, but analogous constructions also work for other types of objects. Given context $\mathbf{x}$, each document in ranking $\mathbf{y} = (d_{i_1}, d_{i_2}, \ldots, d_{i_n})$ has a feature representation given by $\phi(\mathbf{x}, d_{i_j}) \in \mathbf{R}^m$. We then obtain the overall feature vector $\phi_F(\mathbf{x}, \mathbf{y})$ as

$$\phi_F^j(\mathbf{x}, \mathbf{y}) = F(\gamma_1 \phi^j(\mathbf{x}, d_{i_1}), \gamma_2 \phi^j(\mathbf{x}, d_{i_2}), \ldots \ldots, \gamma_n \phi^j(\mathbf{x}, d_{i_n})) \tag{8}$$

where $\phi^j(\mathbf{x}, d)$ and $\phi_F^j(\mathbf{x}, \mathbf{y})$ represent the j^{th} feature in the vectors $\phi(\mathbf{x}, d)$ and $\phi_F(\mathbf{x}, \mathbf{y})$ respectively. We also introduce position-discounting factors $\gamma_1 \geq \ldots \geq \gamma_j \geq \ldots \geq \gamma_n \geq 0$, which determine how important each position in the ranking is. The choice of aggregation function F determines the diminishing returns profile of the users utility. For example, using a coverage-like aggregation function $F(A) = \max_{a \in A} a$, strongly promotes diversity, since a single document can already maximize utility. On the other extreme lies the additive aggregation function $F(A) = \sum_{a \in A} a$, which leads to a diversity-agnostic (i.e., modular) feature vector. More generally, any monotone increasing and concave function of $\sum_{a \in A} a$ can be used. It was shown [9, 14] that this allows for a broad class of performance measures to be modeled, including many common IR performance metrics (e.g. NDCG, Precision, Coverage).

For a component-wise non-negative vector $\mathbf{w}$, we can compute a ranking that approximately maximizes the utility function, *i.e.,* $\mathbf{y} := \arg\max_{\mathbf{y} \in \mathcal{Y}} \mathbf{w}^\top \phi_F(\mathbf{x}, \mathbf{y})$, using the Greedy Algorithm 1. The algorithm iteratively picks the document with the highest marginal utility to be added to the ranking. Despite its simplicity, Algorithm 1 has good approximation properties for this NP-hard problem.

Lemma 1. *For $\mathbf{w} \geq 0$ and monotone, concave $F : \mathbf{R}^n_{\geq 0} \to \mathbf{R}_{\geq 0}$ that commutes in all arguments, Algorithm 1 produces a ranking that is a β_{gr}-approximate solution, with $\beta_{gr} = \left(1 - \frac{1}{e}\right)$ if $\gamma_1 = \cdots = \gamma_k$ or $\beta_{gr} = 1/2$ otherwise.*

Proof. For $\gamma_1 = \ldots = \gamma_k$ this is a straightforward reduction to monotone submodular maximization with a cardinality constraint for which the greedy algorithm is $(1 - \frac{1}{e})$-approximate [25]. For the more general case we reduce it to submodular

maximization over a partition matroid. Suppose we have documents $\{d_1,\ldots,d_N\}$ and want to find a ranking of length k. Let the new ground set A contain k copies $d_{i,j}$:$j \in \{1,k\}$ of each document d_i, one for each position. The matroid only permits sets containing at most one document per position. Define set function H over A: For set $B(\subseteq A)$, let $C=\{\ldots d_{i_j,j}\ldots\}$ be the set obtained by removing all duplicates from B (*i.e.,* keep only the highest ranked occurrence of a document). Define $H(B)=F(\ldots,\gamma_j\phi(\mathbf{x},d_{i_j}),\ldots)$. The lemma follows from observing that Algorithm 1 is equivalent to the greedy algorithm for maximizing H over A under a matroid constraint, which is known to provide a $\frac{1}{2}$-approximate solution [25].

4 Social Learning Algorithms

In this section, we present two coactive learning algorithms for predicting rankings that optimize social utility. The first considers rankings with discount factors for each rank while the second considers the special case of evaluating the top k results as a set. For both algorithms, we characterize their regret by providing upper bounds.

4.1 Social Perceptron for Rankings (SoPer-R)

Following the utility model introduced in Section 3.2, we now present an algorithm for learning rankings $\mathbf{y} = (d_{i_1}, d_{i_2}, \ldots, d_{i_n})$ that aim to optimize social utility where personal user utility can be represented as

$$U_i(\mathbf{x}_t,\mathbf{y}_t) = \mathbf{v}_i^\top \phi_F(\mathbf{x}_t,\mathbf{y}_t), \tag{9}$$

$$\phi_F^j(\mathbf{x},\mathbf{y}) = F(\gamma_1\phi^j(\mathbf{x},d_{i_1}),\gamma_2\phi^j(\mathbf{x},d_{i_2}),\ldots\ldots,\gamma_n\phi^j(\mathbf{x},d_{i_n})), \tag{10}$$

with $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_n \geq 0$. The submodular DCG metric proposed in [9], where the discount factors are $\gamma_i = \frac{1}{\log_2(1+i)}$, is an example of such a utility function.

The Social Perceptron for Rankings (SoPer-R) is detailed in Algorithm 2. It applies to any F that satisfies the conditions of Lemma 1. The algorithm maintains a weight vector $\mathbf{w}_t$, which is its estimate of $\mathbf{w}_*$. For the given context $\mathbf{x}_t$, the algorithm first computes ranking $\mathbf{y}_t$ using the greedy Algorithm 1, which is then presented to the user. The user actions (e.g., clicks) are observed and used to construct the feedback as follows. The ranking is first partitioned into adjacent pairs by randomly selecting an odd or even grouping. The feedback ranking $\bar{\mathbf{y}}_t$ is constructed by swapping the documents whenever the user clicks on the lower element of the pair. This relates to the idea of FairPairs [26], which is used to help de-bias click data. Note that feedback is only generated whenever the lower elements was clicked but not the upper, otherwise $\bar{\mathbf{y}}_t := \mathbf{y}_t$. After the feedback $\bar{\mathbf{y}}_t$ is received, the algorithm performs a perceptron-style update to the weight vector. To to ensure that the weight vector contains only non-negative weights, any negative weights are *clipped* to zero.

Given function g and constant $0 \leq \lambda \leq 1$ define $\tau_g(\lambda)$ as:

$$\tau_g(\lambda) = \lim_{x\to 0} \frac{g(\lambda\cdot x,0,\ldots,0)}{g(x,0,\ldots,0)} \tag{11}$$

Algorithm 2. Social Perceptron for Ranking (SoPer-R)

Initialize $\mathbf{w}_0 \leftarrow \mathbf{0}$
for $t = 0$ **to** $T - 1$ **do**
  Observe $\mathbf{x}_t$
  Present $\mathbf{y}_t \leftarrow GreedyRanking(\mathbf{w}_t, \mathbf{x}_t)$ {Present argmax ranking}
  Observe user clicks $\mathcal{D}$ {Get User Feedback}
  Construct feedback $\bar{\mathbf{y}}_t \leftarrow ListFeedback(\mathbf{y}_t, \mathcal{D})$ {Create Feedback Object}
  Update: $\bar{\mathbf{w}}_{t+1} \leftarrow \mathbf{w}_t + \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)$ {Perceptron Update}
  Clip: $\mathbf{w}^j_{t+1} \leftarrow \max(\bar{\mathbf{w}}^j_{t+1}, 0) \quad \forall 1 \leq j \leq m.$

Function ListFeedback$(\mathbf{y}, \mathcal{D})$ {$\mathbf{y}$: Presented Ranking; $\mathcal{D}$: User clicks }
$\bar{\mathbf{y}} \leftarrow \mathbf{y}$ {Initialize with presented object}
With probability 0.5: $\mathcal{PR} \leftarrow (\{1,2\},\{3,4\},\{5,6\}\cdots)$
else: $\mathcal{PR} \leftarrow (\{1\},\{2,3\},\{4,5\},\{6,7\}\cdots)$
for $i = 0 \cdots len(\mathcal{PR})$ **do**
  $\{j_{upper}, j_{lower}\} \leftarrow \mathcal{PR}[i]$ {Get Pair}
  if $\mathbf{y}[j_{lower}] \in \mathcal{D}$ AND $\mathbf{y}[j_{upper}] \notin \mathcal{D}$ **then**
    Swap($\bar{\mathbf{y}}[j_{upper}]$, $\bar{\mathbf{y}}[j_{lower}]$) {Place clicked doc above the other doc}
return $\bar{\mathbf{y}}$

The below lemma bounds the change in a concave function on scaling arguments.

Lemma 2. *For any function g (satisfying the conditions of Lemma 1), constant $0 \leq \lambda \leq 1$ and values $v_1, v_2, \ldots, v_n \geq 0$, we can bound the change in value of g on scaling the values v_i by λ as follows:*

$$g(v_1, \ldots, v_i, \ldots, v_n) \geq \tau_g(\lambda) \cdot g(\lambda \cdot v_1, \ldots, \lambda \cdot v_i, \ldots, \lambda \cdot v_n) \tag{12}$$

We use this to characterize the sequence of position discounts and their smoothness, which is a key parameter of the main theorem. Thus for a utility measure with function F and γ_i discount factors, we define:

$$\Gamma_F = 1 - \min_i \tau_F(\frac{\gamma_{i+1}}{\gamma_i}) \tag{13}$$

We can now characterize the regret suffered by the SoPer-R algorithm for list-based utilities, as shown below in Theorem 1.

Theorem 1. *For any $\mathbf{w}_* \in \mathbf{R}^m$ and $\|\phi(\mathbf{x}, \mathbf{y})\|_{\ell_2} \leq R$ the average regret of the SoPer-R algorithm can be upper bounded as:*

$$REG_T \leq \frac{1}{\eta T} \sum_{t=0}^{T-1} \mathbf{E}_i[p_i \bar{\xi}_t] + \frac{\beta R \|\mathbf{w}_*\|}{\eta} + \frac{\sqrt{2}\sqrt{4 - \beta^2} R \|\mathbf{w}_*\|}{\eta \sqrt{T}}. \tag{14}$$

with: $\delta_i \geq \left(\Gamma_F \cdot \frac{1-p_i}{p_i}\right)$, $\eta = \min_i \ p_i \alpha_i$ *and* $\beta = (1 - \beta_{gr}) = \frac{1}{2}$.

Before presenting the proof of the theorem, we first analyze the structure of the regret bound. The first term on the right-hand side characterizes in how far the

user feedback violates the desired α_i, δ_i-informative feedback assumption due to model misspecification and bias/noise in the user feedback. This term implies that the regret does not necessarily converge to zero in such cases.

The second term results from the fact that we can only guarantee a β_{gr}-approximate solution for greedy Algorithm 1. In practice, however, the solutions computed by greedy Algorithm 1 tend to be much better, making the second term much smaller than in the worst case.

The third and final term converges to zero at a rate of $T^{0.5}$. Note that none of the terms in the bound depend explicitly on the number of features, but that that it scales only in terms of margin $R||\mathbf{w}_*||$.

Proof. From Lemma 1, we get that:

$$\begin{aligned} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \mathbf{y}_t) &\geq \beta_{gr} \mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) \\ \mathbf{w}_t^\top (\phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) - \phi(\mathbf{x}_t, \mathbf{y}_t)) &\leq (1 - \beta_{gr}) \mathbf{w}_t^\top \phi(\mathbf{x}_t, \bar{\mathbf{y}}_t) \leq \beta R \|\mathbf{w}_t\| \end{aligned} \tag{15}$$

Next, we bound the ℓ_2 norm of $\mathbf{w}_T$:

$$\begin{aligned} \|\mathbf{w}_T\|^2 &= \|\mathbf{w}_{T-1}\|^2 + 2\mathbf{w}_{T-1}^\top (\phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})) \\ &\quad + \|\phi(\mathbf{x}_{T-1}, \bar{\mathbf{y}}_{T-1}) - \phi(\mathbf{x}_{T-1}, \mathbf{y}_{T-1})\|^2 \\ &\leq \|\mathbf{w}_{T-1}\|^2 + 2\beta \|\mathbf{w}_{T-1}\| R + 4R^2 \\ &\leq (\beta T + \sqrt{4 - \beta^2}\sqrt{2T})^2 R^2 \end{aligned} \tag{16}$$

Eq. (15) is used for the second inequality. The last line is obtained using the inductive argument made in [14]. Similarly we bound $\mathbf{E}[\mathbf{w}_T^\top \mathbf{w}_*]$ using Cauchy-Schwartz and concavity:

$$\|\mathbf{w}_*\| \mathbf{E}[\|\mathbf{w}_{T+1}\|] \geq \mathbf{E}[\mathbf{w}_T^\top \mathbf{w}_*] = \sum_{t=0}^{T-1} \mathbf{E}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] \tag{17}$$

Now we use the α_i, δ_i-informativeness condition:

$$\begin{aligned} \mathbf{E}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U_i(\mathbf{x}_t, \mathbf{y}_t)] &\geq \alpha_i \left(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t) \right) + \delta_i U_i(\mathbf{x}_t, \mathbf{y}_t) - \bar{\xi}_t \\ &\geq \frac{\eta}{p_i} \Big(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t) \Big) + \delta_i U_i(\mathbf{x}_t, \mathbf{y}_t) - \bar{\xi}_t \end{aligned} \tag{18}$$

Next we bound the expected difference in the social utility between $\bar{\mathbf{y}}_t$ and $\mathbf{y}_t$ IF a user of type i provided feedback at iteration t:

$$\begin{aligned} \Delta_i &= \mathbf{E}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] \geq -\Gamma_F \sum_{j \neq i} p_j U_j(\mathbf{x}_t, \mathbf{y}_t) + p_i \mathbf{E}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U_i(\mathbf{x}_t, \mathbf{y}_t)] \\ &= -\Gamma_F (U(\mathbf{x}_t, \mathbf{y}_t) - p_i U_i(\mathbf{x}_t, \mathbf{y}_t)) + p_i \mathbf{E}[U_i(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U_i(\mathbf{x}_t, \mathbf{y}_t)] \\ &\geq -\Gamma_F U(\mathbf{x}_t, \mathbf{y}_t) + p_i \Gamma_F U_i(\mathbf{x}_t, \mathbf{y}_t) + \eta \Big(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t) \Big) + p_i \delta_i U_i(\mathbf{x}_t, \mathbf{y}_t) - p_i \bar{\xi}_t \\ &\geq \eta \Big(U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i}) - U_i(\mathbf{x}_t, \mathbf{y}_t) \Big) + \Gamma_F \Big(U_i(\mathbf{x}_t, \mathbf{y}_t) - U(\mathbf{x}_t, \mathbf{y}_t) \Big) - p_i \bar{\xi}_t \end{aligned} \tag{19}$$

Algorithm 3. Social-Set-Based-Perceptron(C, M, p)

Function SetFeedback$(\mathbf{y}, \mathcal{D})$
$\bar{\mathbf{y}} \leftarrow \mathbf{y}$ {Initialize with presented object}
$\mathcal{D}_{\mathcal{O}} \leftarrow \mathcal{D}/\mathbf{y}[1:M]$ {Clicks on docs outside top M}
for $i = 1 \cdots \min(C, |\mathcal{D}_{\mathcal{O}}|)$ **do**
  $c \leftarrow \mathcal{D}_{\mathcal{O}}[i]$ {Clicked document}
  $u \leftarrow$ Random (unclicked) document from $\mathbf{y}[1:M]$ {Non-clicked document}
  Swap($\bar{\mathbf{y}}[j_u]$, $\bar{\mathbf{y}}[j_c]$)
return $\bar{\mathbf{y}}$

The first line is obtained by using Lemma 2 and definition of Γ_F (Eq. 13). The second line uses the definition of the social utility (Eq. 1). The third line uses Eq. 18. The fourth step uses the condition on δ_i and rearranging of terms. Note that the expectations in the above lines are w.r.t. the user feedback (and the feedback construction process).

We next consider the expected value of Δ_i (over the user distribution):

$$\begin{aligned}\mathbf{E}_i[\Delta_i] = \mathbf{E}[U(\mathbf{x}_t, \bar{\mathbf{y}}_t) - U(\mathbf{x}_t, \mathbf{y}_t)] &\geq \eta\Big(\mathbf{E}_i[U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i})] - U(\mathbf{x}_t, \mathbf{y}_t)\Big) - \mathbf{E}_i[p_i\bar{\xi}_t] \\ &\geq \eta\Big(U(\mathbf{x}_t, \mathbf{y}_t^*) - U(\mathbf{x}_t, \mathbf{y}_t)\Big) - \mathbf{E}_i[p_i\bar{\xi}_t] \quad (20)\end{aligned}$$

where the second line uses the fact that $\mathbf{E}_i[U_i(\mathbf{x}_t, \mathbf{y}_t^{*,i})] \geq U(\mathbf{x}_t, \mathbf{y}_t^*)$. We can put together Eqns. 16, 17 and 20 to give us the required bound.

4.2 Social Perceptron for Sets (SoPer-S)

While DCG-style position discounts γ_i that decay smoothly are often appropriate, other models of utility require more discrete changes in the rank discounts. The *coverage* metric is an example of such a metric, which measures what fraction of the users will find atleast 1 document relevant to them in the set of M documents [11, 8, 12]. We call these metrics *set-based*, since they consider the first M documents in a ranking as a set (*i.e.,* position within the top-M positions does not matter). Clearly, we can model such metrics by setting the γ_i in the aggregation step (defined in Eq. 8) as

$$\gamma_i = \begin{cases} 1 & \text{if } i \leq M \\ 0 & \text{if } i > M. \end{cases}$$

However, the bound in Theorem 1 can be rather loose for this case, and the pairwise feedback construction model "wastes" information. In particular, since utility is invariant to reordering in the top M or below the top M, only pairwise feedback between position M and $M+1$ provides information. To overcome this problem, we now present an alternate algorithm that is more appropriate for set-based utility functions.

The *Social Perceptron for Sets* (SoPer-S), shown in Algorithm 3, uses the same basic algorithm, but replaces the feedback mechanism. Now, clicked documents

outside the top M are swapped with a random non-clicked document in the top M. This leads to a feedback set $\bar{\mathbf{y}}_t$ (of size M), that contains more (or at least as many) of the user's preferred documents than the top M elements of the presented ranking. Note that during the feedback creation, we only consider the first C clicks outside the top M. This parameter C is used to restrict the difference between the feedback set and the presented set. We now state a lemma we will use to bound the regret of the SoPer-S algorithm.

Lemma 3. *For any non-negative, submodular function g and set X with $|X| = n$, we can lower bound the function value of a random subset of size k as:*

$$\mathbf{E}_{Y:Y\subseteq X,|Y|=k}[g(Y)] \geq \frac{k}{n} g(X) \tag{21}$$

Using Lemma 3, we can now characterize the regret suffered by the SoPer-S algorithm for set-based utilities, as shown below in Theorem 2.

Theorem 2. *For any $\mathbf{w}_* \in \mathbf{R}^m$ and $\|\phi(\mathbf{x}, \mathbf{y})\|_{\ell_2} \leq R$ the average regret of the SoPer-S algorithm can be upper bounded as:*

$$REG_T \leq \frac{1}{\eta T} \sum_{t=0}^{T-1} \mathbf{E}_i[p_i \bar{\xi}_t] + \frac{\beta R \|\mathbf{w}_*\|}{\eta} + \frac{\sqrt{2}\sqrt{4-\beta^2} R \|\mathbf{w}_*\|}{\eta\sqrt{T}}. \tag{22}$$

with: $\delta_i \geq \left(\frac{C}{M} \cdot \frac{1-p_i}{p_i}\right)$, $\eta = \min_i \ p_i\alpha_i$ *and* $\beta = (1 - \beta_{gr}) = \frac{1}{e}$.

Note that the proposed algorithms are efficient (due to the online updates) and scalable with the greedy algorithm requiring $O(nk)$ time to find a length k ranking over n documents. This can be further improved using lazy evaluation.

5 Empirical Evaluation

In this section, we empirically analyze the proposed learning algorithms for the task of *extrinsic* [1] search result diversification. In particular, we (a) explore how well the algorithms perform compared to existing algorithms that do single-query learning; we (b) compare how close our algorithms get to the performance of algorithms that require expert annotated examples of socially optimal ranking for cross-query learning; and (c) we explore the robustness of our algorithm to noise and misspecification of the utility model.

5.1 Experiment Setup

We performed experiments using the standard diversification dataset from the *TREC 6-8 Interactive Track*. The dataset contains 17 queries, each with binary relevance judgments for 7 to 56 different user types, which we translate into binary utility values. Similar to previous work [9], we consider the probability of a user type to be proportional to the number of documents relevant to that

Table 1. Summary of key properties of the TREC dataset

Statistic	Value
Average number of documents per query	46.3
Average number of user types	20.8
Fraction of docs. relevant to > 1 user	0.21
Average number of users a document is relevant for	1.33
Fraction of docs. relevant to most popular user	0.38
Average probability of most popular user	0.29

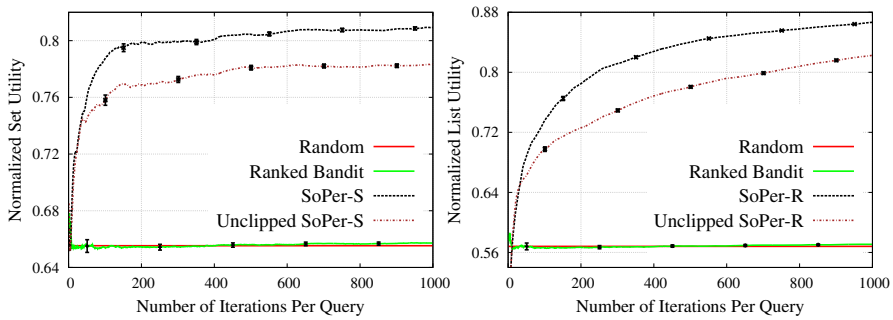

Fig. 2. Performance of different methods for single-query learning to diversify. Performance is averaged over all queries, separately considering Set Utility (Left) and List Utility (Right). Standard error bars are shown in black.

user type. Also following [9], we only consider documents that are relevant to at least 1 user type to focus the experiments on learning to diversify, not learning to determine relevance. Table 1 summarizes some key properties of the data.

To simulate user behavior, we use the following model. Users scan the documents of a ranking in order and click on the first document they consider relevant. Each (binary) decision of relevance is made incorrectly with a small probability of error. This error probability was set to zero for most experiments but later varied when studying the effect of user noise.

Unless mentioned otherwise, we used the coverage function ($F(x_1, \ldots, x_n) = \max_i x_i$) to define the submodular function for utility aggregation. We measured performance of the different methods in terms of the utility being optimized - *i.e.,* Set Utility (of size 5 sets) for the Set-Based methods and List Utility (up to rank 5) with DCG discounting factors, for the List-Based methods. Additionally we normalize the maximum scores per query to 1 (i.e., $\forall x : U(\mathbf{x}, \mathbf{y}^*) = 1$), so as to get comparable scores across queries. We report the performance of each algorithm in terms of its running average of these scores (i.e., $1 - REG_T$).

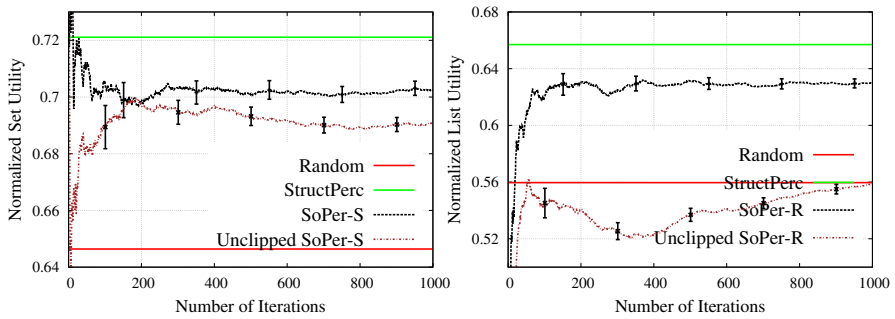

Fig. 3. Set (L) and List (R) Utilities for learning to diversify across queries

5.2 Can We Learn to Diversify for a Single Query?

We first evaluate our algorithms in the setting of the Ranked Bandits algorithm [11], which serves as a baseline. The Ranked Bandit algorithm learns a separate model for each query and cannot generalize across queries. Furthermore, its original version was limited to optimizing the coverage function, corresponding to the *max* aggregation in our framework. We use the *UCB1* variant of the Ranked Bandits algorithm, which was empirically found to be the best variant.

As a second baseline we report randomly ordering the results. Note that this is a competitive baseline, since (a) all documents are relevant to at least 1 user, and (b) the probability of users is proportional to the number of documents relevant to them.

For the SoPer-R and SoPer-S algorithms, documents were represented as unit-norm TFIDF word vectors. All learning algorithms were run twice for each of the 17 queries (with different random seeds) and the results are averaged across all 34 runs. As seen from Figure 2, the proposed algorithms perform much better than either of the two baselines. The Ranked Bandits algorithm converges extremely slowly, and is barely better than the random baseline after 1000 iterations. Both the SoPer-R and SoPer-S algorithm are able to learn substantially faster. Already within 200 iterations, the SoPer-S method is able to provide at least 1 relevant document to 80% of the user population, while random and Ranked Bandits perform at around 65%. Thus both proposed methods are clearly able to learn the diversity required in such rankings from individual user feedback.

We also explore variants of the SoPer-S and SoPer-R algorithms where we omit the final step of clipping negative weights to 0. While the unclipped versions of both algorithms still perform better than random, they fall short of the corresponding clipped versions as seen from Figure 2. Thus we can conclude that ensuring non-negative weights not only guarantees theoretical results, but is important for empirical performance.

5.3 Can We Learn a Cross-Query Model for Diversification?

While the previous experiments indicate that the new algorithms can learn to diversify for a single query, such single-query learning is restricted to frequent queries that are issued hundreds of times. Instead, it is more desirable for diversification models to be trained across a distribution of queries.

To get a suitable representation that allows cross-query learning, we use the same *word-importance* feature vectors that were used in previous work on learning from expert-annotated feedback [8, 9]. These features capture both the overall importance of a word (e.g., "Does the word appears in at least x% of the documents?"), as well as the importance in the documents of the ranking (e.g., "Does the word appear with frequency of atleast y% in the document?"). Using different such values of x and y along with other similar features, we get a total of 1197 features.

To produce the following results, all methods were run for 1000 iterations with 5 random seeds. The values reported are averaged across these 5 runs.

In this cross-query setting, we cannot apply Ranked-Bandits as it only works for a single query. Thus we again use the Random baseline in this experiment. Existing supervised learning algorithms for diversification are also not applicable here, as they require explicit training data of socially optimal rankings (i.e., knowledge of all document-user relevance labels). However, we would like to estimate how well our algorithms can learn from (far weaker) implicit feedback data, in relation to conventional methods trained in such a *full information* setting. Thus we trained a structural perceptron, which internally uses the greedy algorithm for prediction. This uses the same feature vector representation as our algorithm, but is provided the *social optimal* at every iteration.

Fig. 3 shows the average utility for the SoPer-S and SoPer-R algorithms, as well as the random baseline and the Structured Perceptron after 1000 iterations. Both SoPer-S and SoPer-R substantially outperform the random baseline, indicating that the proposed algorithms can learn to diversify for this cross-query setting. Both methods get close to the performance of the supervised method despite learning from far weaker feedback. For example, the SoPer-S method is able to satisfy 70% of the user population, as compared to the 64% of the baseline and 72% of the Structured Perceptron. We also again evaluate the unclipped versions of the algorithms. For the the unclipped SoPer-R, performance never rises above random, indicating the practical importance of maintaining a positive weight vector to ensure good performance of the greedy algorithm.

5.4 How Robust Are the Algorithms to Misspecification of the Model?

While the previous experiments showed that the algorithms can learn efficiently when the submodular function of the user population (as used in computing the personal and social utilities) and the algorithm match, we now study what happens when there is a mismatch. More specifically, for the *cross-query diversification* setting, we ran the algorithms with three different submodular functions

Table 2. Set and List Utilities (with standard error) when the two submodular functions *i.e.,* of the population (fixed for row) and the algorithm (fixed for column) are mismatched

UserF	SET			
	Max	Sqrt	Lin	Rand
Max	.699 ±.005	.695 ±.005	.683 ±.005	.646 ±.006
Sqrt	.675 ±.006	.686 ±.006	.706 ±.006	.634 ±.006
Lin	.509 ±.006	.532 ±.006	.574 ±.007	.492 ±.006
UserF	LIST			
	Max	Sqrt	Lin	Rand
Max	.630 ±.007	.620 ±.006	.618 ±.006	.557 ±.006
Sqrt	.656 ±.007	.654 ±.007	.684 ±.006	.610 ±.007
Lin	.500 ±.006	.504 ±.006	.566 ±.007	.474 ±.007

Table 3. Ranking performance in the presence of feedback noise

Utility	Random	No Noise	Noise
Set	.646 ±.006	.699 ±.005	.694 ±.006
List	.557 ±.006	.630 ±.007	.631 ±.007

as defined by the concave function F: a) **Max**: $F(x_1, \ldots, x_n) = \max_i x_i$; b) **Lin**: $F(x_1, \ldots, x_n) = \sum_i x_i$; c) **Sqrt**: $F(x_1, \ldots, x_n) = \sqrt{\sum_i x_i}$. We also varied the population utility to each of these three functions, and obtained the average utility value (after 200 iterations) for all 9 combinations of functions. Note that we still ensured that SoPer-R was used to optimize the List based utilities, while SoPer-S was used for set-based ones.

The results (averaged over 5 runs) are shown in Table 2. We find that for both methods and all three population utility functions, the utility value is always better than the random baseline, regardless of the algorithm and function used. While the values may be highest when the functions align, we still find significant improvements over the baselines even when there is a mismatch. In fact, for some situations we find that the utility is highest when there is a mismatch: The case of a linear algorithm utility but SQRT population utility is one such example. We conjecture that is due to the relatively small set/list size of 5. On short rankings LIN and SQRT do not differ as much as on longer rankings. Additionally LIN does not suffer any approximation degradation as the greedy algorithm always provides an optimal solution for LIN.

5.5 Is the Method Robust to Noise in the Feedback?

In the real world, users make errors in judging the relevance of documents. To model this, we simulated users who make an error in each binary relevance judgment with 0.1 probability. This means that, as users go down the ranking,

they may flip the true relevance label. Users now return as feedback the first document they *perceive* as relevant, which contains significant noise. We ran both our algorithms and measured the average utility after 200 iterations in the cross-query setting, with matching algorithm and population utilities using the Max function.

Table 3 shows the results (averaged over 5 runs) comparing the performance of the algorithms in both the noise-free and noisy settings. We see that the performance for both SoPer-S and SoPer-R is almost the same, with the gap to the baseline still being significant. The robustness to noise is also supported by the theoretical results. In particular, note that the definition of α_i, δ_i-informative feedback only requires that feedback be informative in expectations, such that the slack terms $\bar{\xi}_t$ may be zero even for noisy feedback. In general, we conclude that the algorithms are robust and applicable in noisy settings.

6 Conclusions

We proposed two online-learning algorithms in the coactive setting for aggregating the conflicting preferences of a diverse user population into a ranking that aims to optimize social utility. Formalizing the learning problem and model as learning an aggregate utility function that is submodular in the elements of the ranking and linear in the parameters, we were able to provide regret bounds that characterize the worst-case behavior of the algorithm. In an empirical evaluation, the algorithms learned substantially faster than existing algorithms for single-query diversification. For learning cross-query diversification models, the algorithms are robust and the first that can be trained using implicit feedback. This work was supported in part by NSF Awards IIS-1217686, IIS-1247696, IIS-0905467, the Cornell-Technion Joint Research Fund, and a Google Fellowship.

References

1. Radlinski, F., Bennett, P.N., Carterette, B., Joachims, T.: Redundancy, diversity and interdependent document relevance. SIGIR Forum 43(2), 46–52 (2009)
2. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: SIGIR, pp. 335–336 (1998)
3. Zhai, C.X., Cohen, W.W., Lafferty, J.: Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In: SIGIR, pp. 10–17 (2003)
4. Chen, H., Karger, D.R.: Less is more: probabilistic models for retrieving fewer relevant documents. In: SIGIR, pp. 429–436 (2006)
5. Swaminathan, A., Mathew, C.V., Kirovski, D.: Essential pages. In: Web Intelligence, pp. 173–182 (2009)
6. Clarke, C.L.A., Kolla, M., Vechtomova, O.: An effectiveness measure for ambiguous and underspecified queries. In: Azzopardi, L., Kazai, G., Robertson, S., Rüger, S., Shokouhi, M., Song, D., Yilmaz, E. (eds.) ICTIR 2009. LNCS, vol. 5766, pp. 188–199. Springer, Heidelberg (2009)
7. Santos, R.L., Macdonald, C., Ounis, I.: Selectively diversifying web search results. In: CIKM, pp. 1179–1188 (2010)

8. Yue, Y., Joachims, T.: Predicting diverse subsets using structural SVMs. In: ICML, pp. 1224–1231 (2008)
9. Raman, K., Joachims, T., Shivaswamy, P.: Structured learning of two-level dynamic rankings. In: CIKM, pp. 291–296 (2011)
10. Kulesza, A., Taskar, B.: Learning determinantal point processes. In: UAI, pp. 419–427 (2011)
11. Radlinski, F., Kleinberg, R., Joachims, T.: Learning diverse rankings with multi-armed bandits. In: ICML, pp. 784–791 (2008)
12. Slivkins, A., Radlinski, F., Gollapudi, S.: Ranked bandits in metric spaces: learning optimally diverse rankings over large document collections. JMLR 14, 399–436 (2013)
13. Shivaswamy, P., Joachims, T.: Online structured prediction via coactive learning. In: ICML (2012)
14. Raman, K., Shivaswamy, P., Joachims, T.: Online learning to diversify from implicit feedback. In: KDD, pp. 705–713 (2012)
15. Raman, K., Joachims, T., Shivaswamy, P., Schnabel, T.: Stable coactive learning via perturbation. In: ICML (2013)
16. Yue, Y., Guestrin, C.: Linear submodular bandits and their application to diversified retrieval. In: NIPS, pp. 2483–2491 (2012)
17. Joachims, T., Granka, L., Pan, B., Hembrooke, H., Radlinski, F., Gay, G.: Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. TOIS 25(2) (April 2007)
18. El-Arini, K., Guestrin, C.: Beyond keyword search: discovering relevant scientific literature. In: KDD, pp. 439–447 (2011)
19. Blumrosen, L., Dobzinski, S.: Welfare maximization in congestion games. In: EC, pp. 52–61 (2006)
20. Meyers, C.A., Schulz, A.S.: The complexity of welfare maximization in congestion games. Netw. 59(2), 252–260 (2012)
21. Dobzinski, S., Schapira, M.: An improved approximation algorithm for combinatorial auctions with submodular bidders. In: SODA, pp. 1064–1073 (2006)
22. Feige, U.: On maximizing welfare when utility functions are subadditive. In: STOC, pp. 41–50 (2006)
23. Boutilier, C., Caragiannis, I., Haber, S., Lu, T., Procaccia, A.D., Sheffet, O.: Optimal social choice functions: a utilitarian view. In: EC, pp. 197–214 (2012)
24. Bartók, G., Pál, D., Szepesvári, C.: Toward a classification of finite partial-monitoring games. In: Hutter, M., Stephan, F., Vovk, V., Zeugmann, T. (eds.) Algorithmic Learning Theory. LNCS, vol. 6331, pp. 224–238. Springer, Heidelberg (2010)
25. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions. Mathematical Programming 14, 265–294 (1978)
26. Radlinski, F., Joachims, T.: Minimally invasive randomization for collecting unbiased preferences from clickthrough logs. In: AAAI, pp. 1406–1412 (2006)