Learning Socially Optimal Information Systems from Egoistic Users

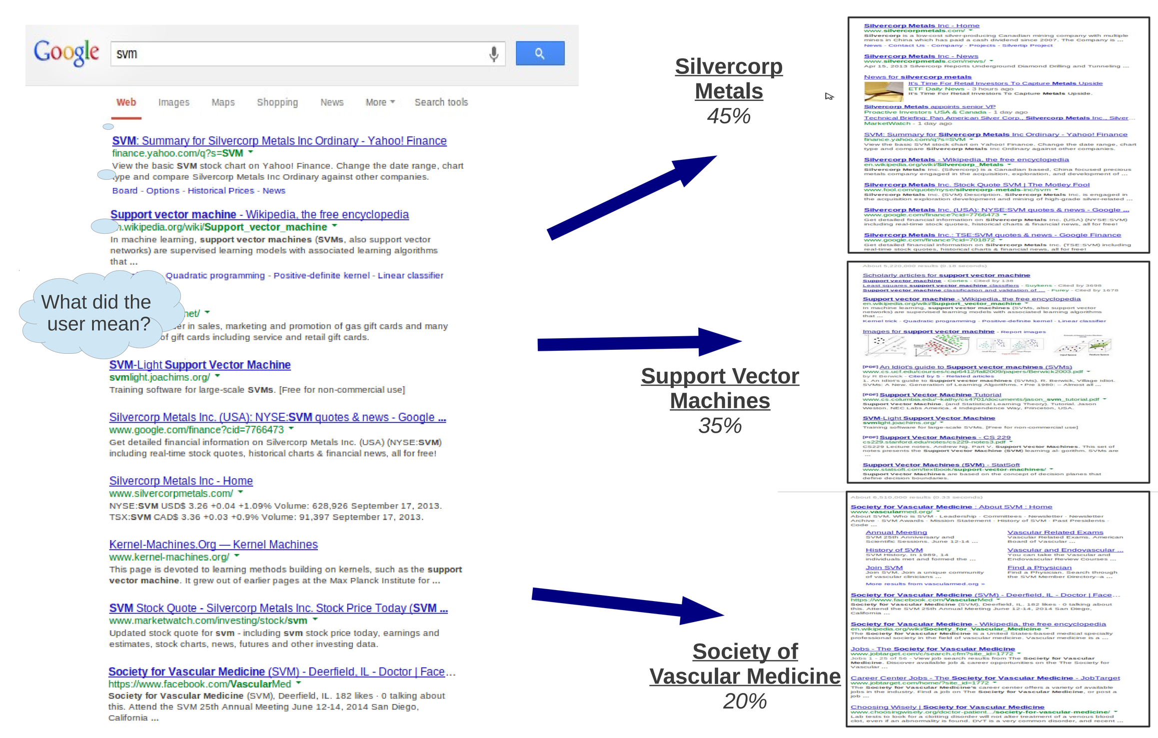
Karthik Raman, Thorsten Joachims ({*karthik,tj*}@cs.cornell.edu)

Cornell University

Problem Overview

- Learn robust systems that collectively satisfy a population of diverse users from user feedback.

Example: Diversity in Search



Goal: Find best overall (*socially optimal*) ranking.

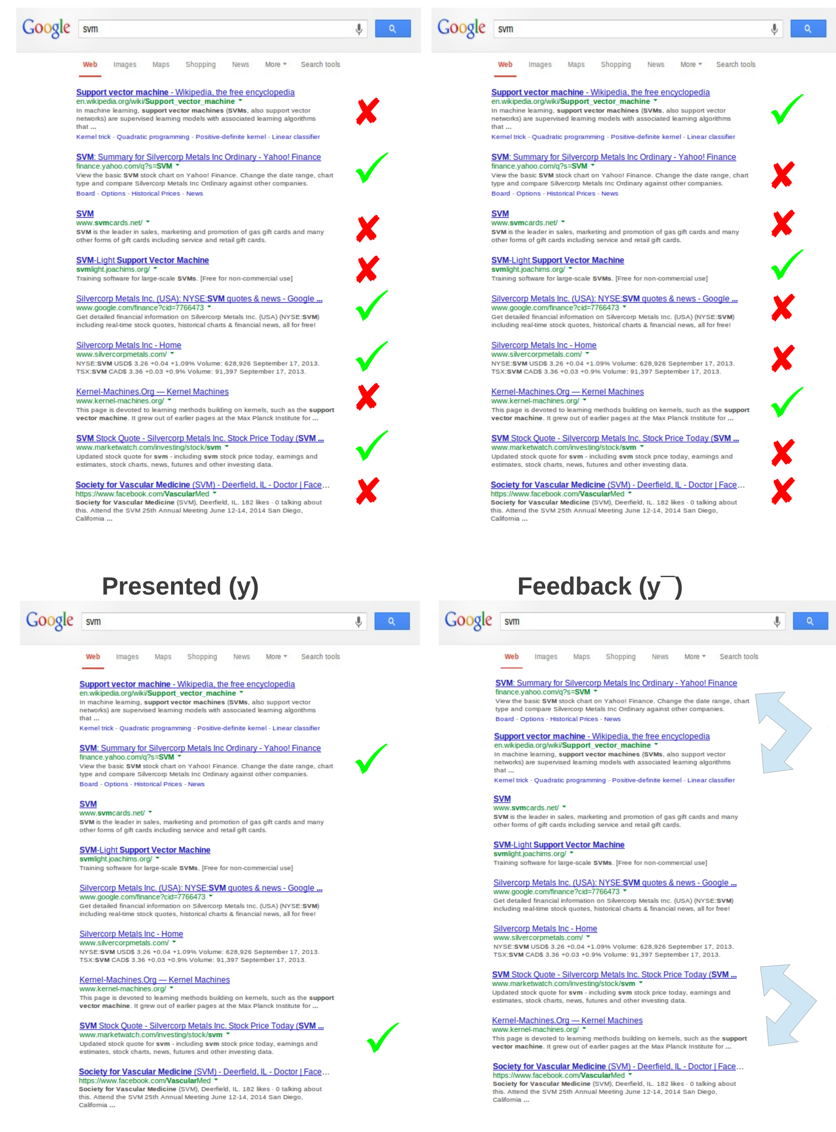
User Feedback

Egoistic user feedback

- User's choice not social.
- Conflicting choices.

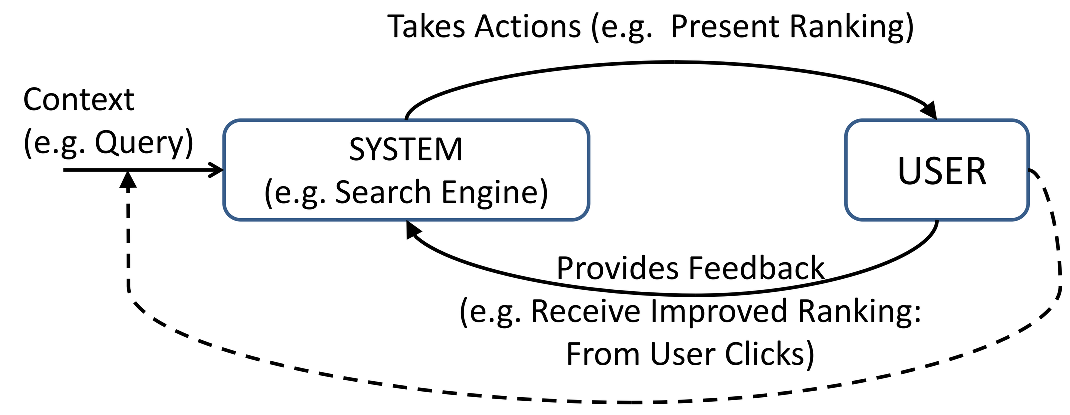
Weak, noisy & biased feedback.

- Cannot regard as cardinal labels.
- Treat as preferences.



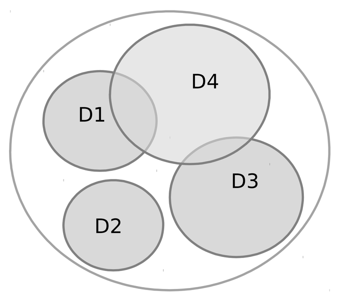
Learning from User Preferences

- Builds on **Coactive Learning** [SJ12,RJSS13].
- Given context $\mathbf{x}$, predict object $\mathbf{y}$.
- **Goal:** Optimize **social** utility $\mathbf{U(x,y) = E[U_i(x,y)]}$.
  - $\mathbf{U_i}$ is **personal** utility of user type $\mathbf{i}$ (w.p. $\mathbf{p_i}$).
- User preferences: Feedback tends to improves personal utility: $\mathbf{U_i(x,\bar{y}) \geq_{\alpha,\delta} U_i(x,y)}$.
  - Not social utility.



Modeling Utility: Submodularity

- Model personal utility of users as submodular in individual elements.
- **Diminishing returns:** Marginal benefit diminishes.
- Example: Coverage Function



- Given ranking/set $\mathbf{y = (d_{i_1}, \ldots, d_{i_n})}$ and position-discount factors $\gamma_1 \geq \gamma_2 \geq \ldots \geq \gamma_n \geq \mathbf{0}$, aggregate features using submodular function $\mathbf{F}$:

$$\phi^j_F(\mathbf{x,y}) = \mathbf{F}(\gamma_1\phi^j(\mathbf{x}, \mathbf{d}_{i_1}), \gamma_2\phi^j(\mathbf{x}, \mathbf{d}_{i_2}) \ldots, \gamma_n\phi^j(\mathbf{x}, \mathbf{d}_{i_n}))$$

  - $\phi^j(\mathbf{x}, \mathbf{d_i})$ is $\mathbf{j^{th}}$ feature of $\mathbf{d_i}$.
- Model personal utility as linear in submodular aggregate: $\mathbf{U_i(x,y) = w^T_{*,i}\phi_F(x,y)}$
  - Submodular aggregation leads to diversity.
- Computing ranking ≈ Submodular maximization
- Use simple, efficient greedy algorithm.
- Approximation guarantee of $\frac{1}{2}$ (under partition matroid constraint).
- Example of Diversity:

Doc	Words
$\mathbf{d_1}$	ma:3 le:3
$\mathbf{d_2}$	ma:5 le:2
$\mathbf{d_3}$	ma:2 le:5
$\mathbf{d_4}$	me:3 si:5
$\mathbf{d_5}$	me:6 si:2
$\mathbf{d_6}$	me:3 si:1

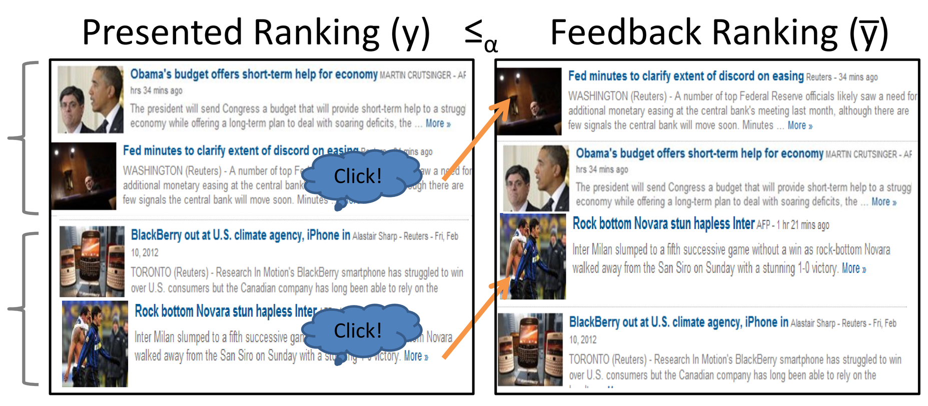
Word	Weight
machine	5
learning	7
metal	4
silver	6

Posn	Doc	ma	le	me	si
1	$\mathbf{d_3}$	2	5	0	0
2	$\mathbf{d_4}$	0	0	3	5
3	$\mathbf{d_2}$	5	2	0	0
MAX of Col		5	5	3	5

Doc	Marginal Benefit		
	Iter1	Iter2	Iter3
$\mathbf{d_1}$	3*5 + 3*7	(3-2)*5	(3-2)*5
$\mathbf{d_2}$	5*5 + 2*7	(5-2)*5	(5-2)*5
$\mathbf{d_3}$	2*5 + 5*7	-	-
$\mathbf{d_4}$	3*4 + 5*6	3*4 + 5*6	-
$\mathbf{d_5}$	6*4 + 2*6	6*4 + 2*6	(6-3)*4
$\mathbf{d_6}$	3*4 + 1*6	3*4 + 1*6	0

Social Perceptron for Ranking

1. Initialize weight vector $\mathbf{w_1} \leftarrow \mathbf{0}$.
2. Given context $\mathbf{x_t}$ present user with $\mathbf{y_t} \leftarrow \text{argmax}_\mathbf{y} \mathbf{w_t^T}\phi(\mathbf{x_t}, \mathbf{y})$.
3. Observe user clicks $\mathcal{D}$.
4. Construct preference feedback: $\bar{\mathbf{y}}_\mathbf{t} \leftarrow \mathbf{PairedFeedback}(\mathbf{y_t}, \mathcal{D})$.
5. Update weight vector: $\bar{\mathbf{w}}_{\mathbf{t+1}} \leftarrow \mathbf{w_t} + \phi(\mathbf{x_t}, \bar{\mathbf{y}}_\mathbf{t}) - \phi(\mathbf{x_t}, \mathbf{y_t})$
6. Clip to be non-negative: $\mathbf{w^j_{t+1}} \leftarrow \mathbf{max}(\bar{\mathbf{w}}^j_{\mathbf{t+1}}, \mathbf{0})$
7. Repeat from step 2.



PairedFeedback: Form pairs and swap if only lower element is clicked.

Referred to as the **SoPer-R** algorithm.

Also provide an algorithm for learning diverse sets called the **SoPer-S** algorithm.

- See paper for more details

Theoretical Analysis

α_i, δ_i-Informative Feedback:

Characterize feedback $\bar{\mathbf{y}}$ in terms of α_i, δ_i, ξ as:

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

- where $\mathbf{y^{*,i}}$ is optimal for user $\mathbf{i}$
- and $\mathbf{y}$ is the presented object.
- **Note** that this is a characterization (not an assumption).
- Does not assume anything about social utility.
- Used to prove regret bounds.

Regret: Define the regret after $\mathbf{T}$ iterations as:

$$\frac{\mathbf{1}}{\mathbf{T}}\sum_{\mathbf{t=1}}^{\mathbf{T}}\left(\mathbf{U(x_t, y^*_t)} - \mathbf{E[U(x_t, y_t)]}\right).$$

- **Note:** In terms of **social** utility and **social optimal**.

Regret Bound

If $\delta_i \geq \left(\Gamma_F \cdot \frac{1-p_i}{p_i}\right)$, average regret of the SoPer-R is:

$$\leq \frac{1}{\eta T}\sum_{t=0}^{T-1} \mathbf{E_i}[p_i\bar{\xi}_t] + \frac{R\|w_*\|}{2\eta} + \frac{\sqrt{15R}\|w_*\|}{\eta\sqrt{2T}}$$

with $\eta = \mathbf{min_i}\ \mathbf{p_i}\alpha_i$.

Understanding the bound:

- Does not depend on number of dimensions only radius of ball $\mathbf{R}$.
- Decays gracefully with weak feedback: α_is.
- Need not converge to optimal (due to NP-hardness of submodular maximization).
- Bound is loose as solution improves.

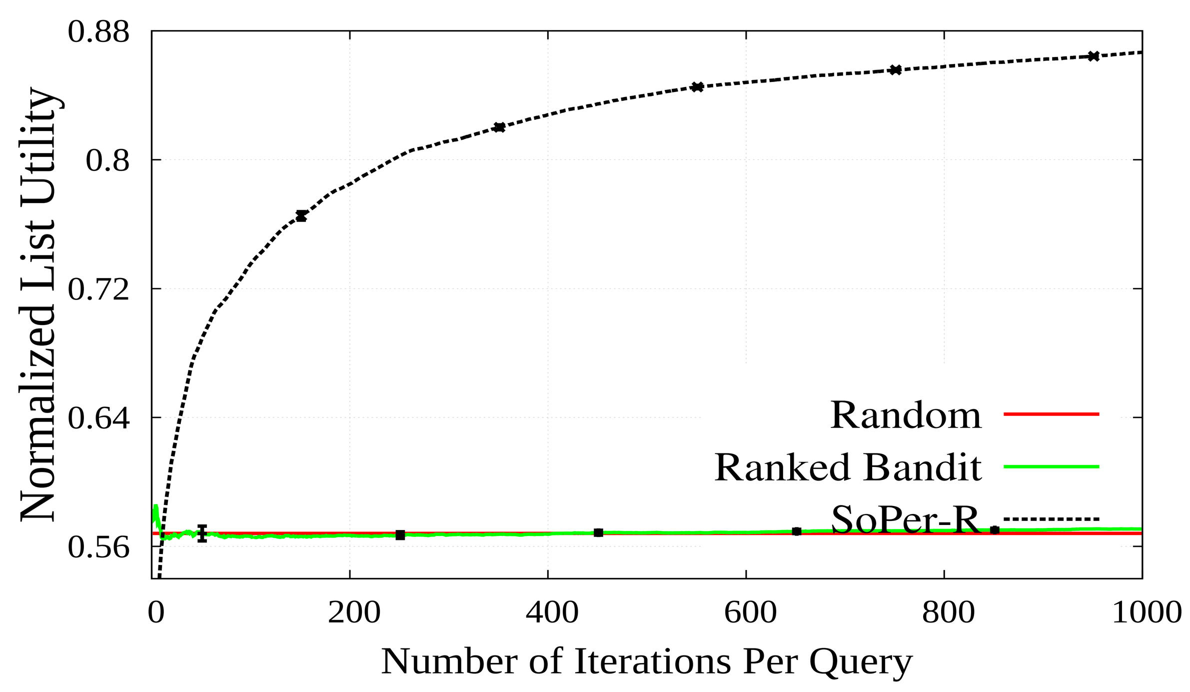
Similar bound for **SoPer-S** algorithm as well.

Experimental Results

- Offline experiments on standard **TREC 6-8 Interactive** search diversification dataset.
  - Queries have 7-56 user types with binary relevance labels.
- Simulated user behavior: Scan rankings top to bottom. Click on first document relevant to them (with some error).
- Utility: Normalized DCG-Coverage function upto rank 5.

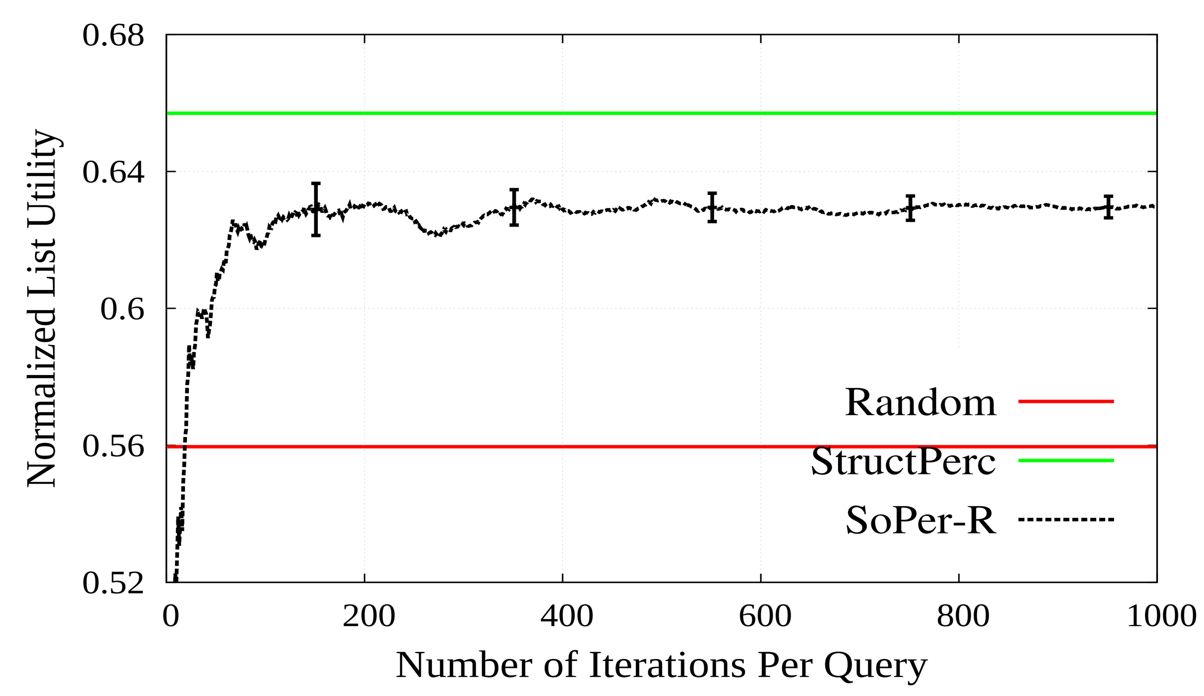
Single Query Diversification:

- Learning to diversify for single query.
- Compare with RankedBandit (Array of coupled MABs).



Cross-Query Diversification:

- Learning to diversify given **any** query.
- Structured Perceptron receives social-optimal as feedback.



- **First** method to diversify across queries from preferences.
- **Robust** to model mis-specification.

TrueSocialF	SoPer-R (Varying Submodular Function)			Rand
	MAX	SQRT	LIN	
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**

- **Robust** to feedback noise (.631 vs .630).