ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 11 – Adaptive Recommendations

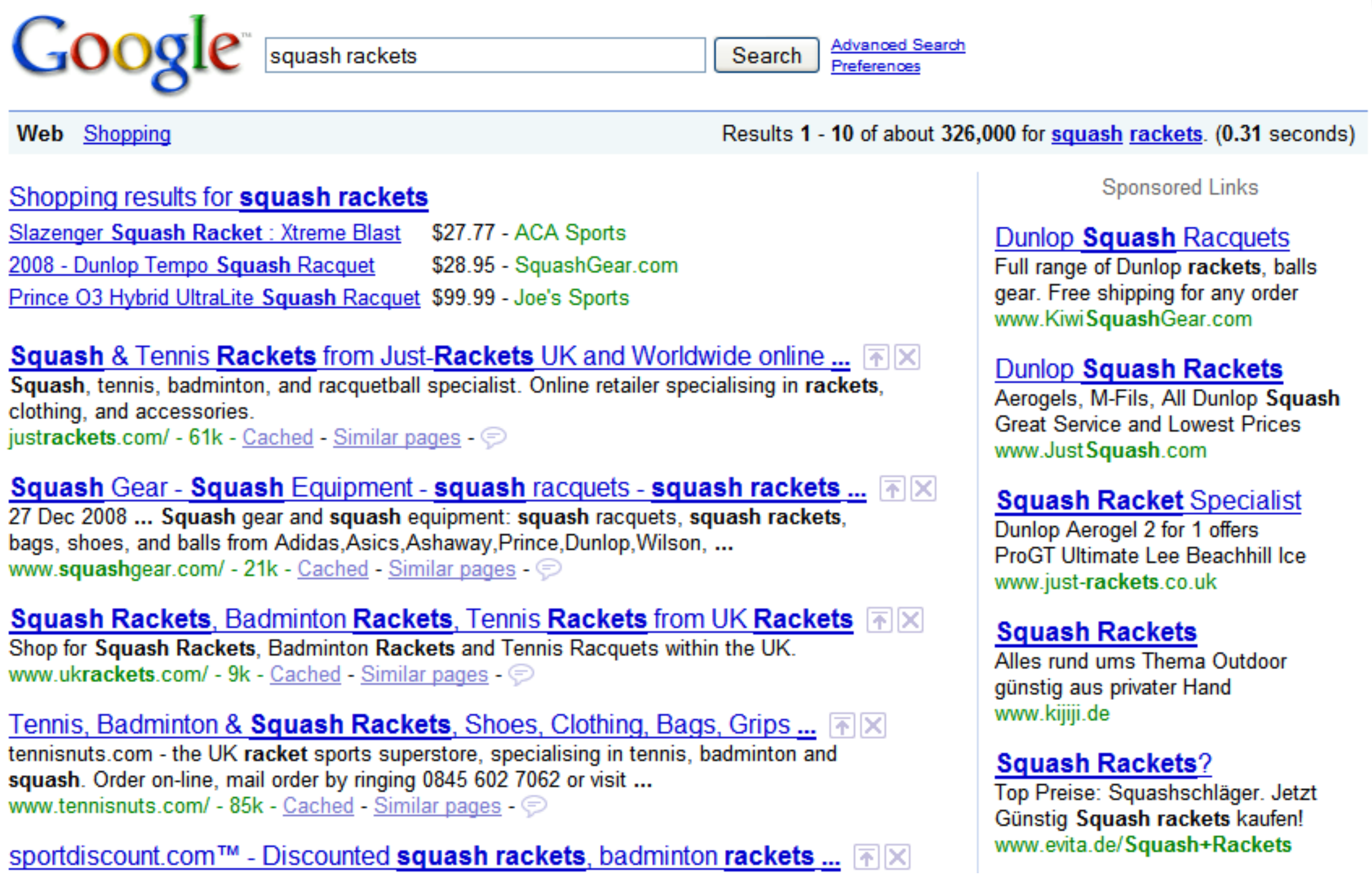
263-5200-00L

Andreas Krause

Course organization

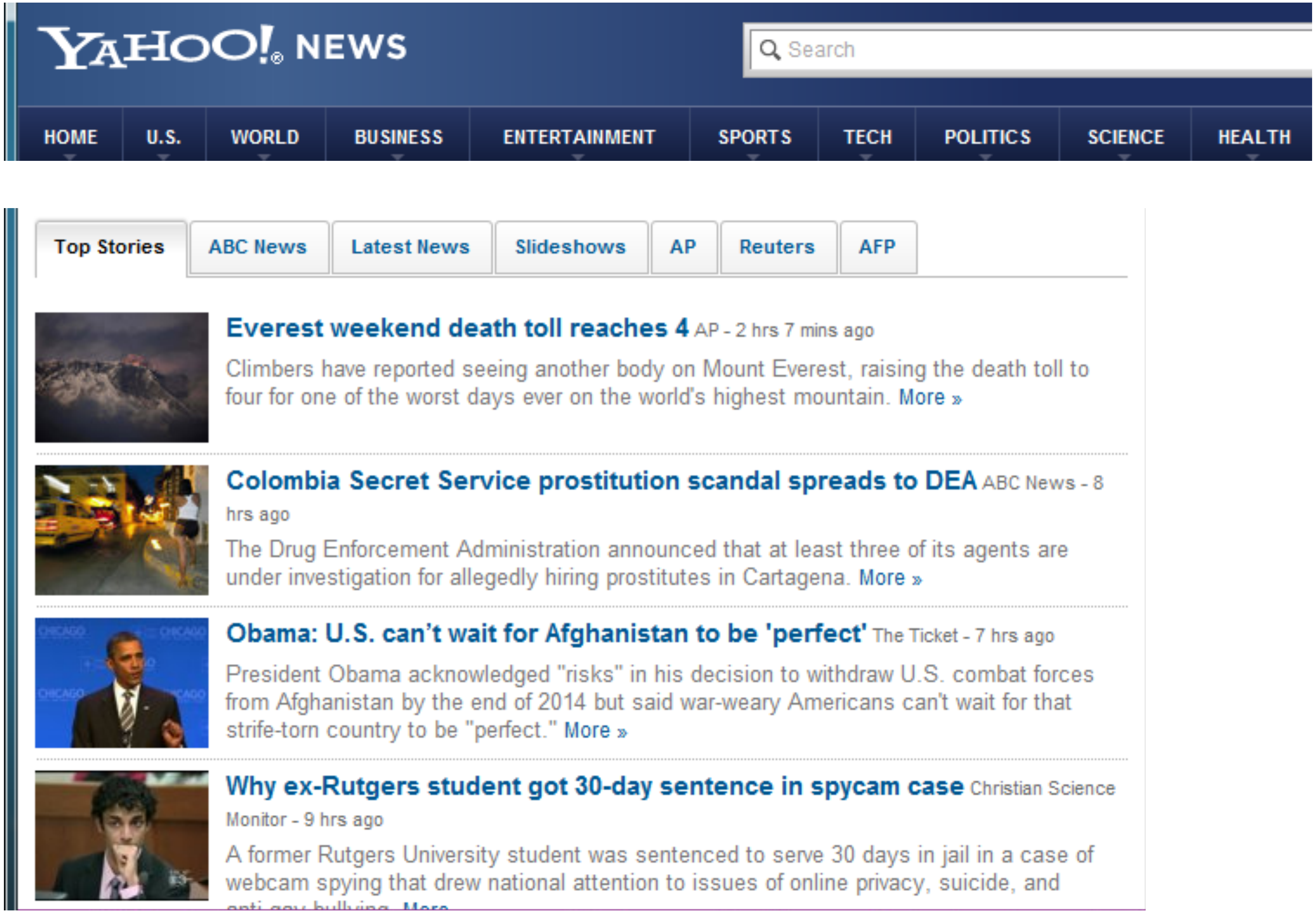
- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - Determine relevance of search results
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Interactive data mining**
  - Learning through experimentation / from limited feedback
  - *Applications*: Online advertising, opt. UI, learning rankings, …

Sponsored search



Which ads should be displayed to maximize revenue?

Which news should we display?



The UCB1 algorithm [Auer et al '02]

- Set $\hat{\mu}_1 = \cdots = \hat{\mu}_k = 0$ $\quad n_1 = \cdots = n_k = 0$
- For t = 1:T
  - For each arm i calculate $UCB(i) = \hat{\mu}_i + \underbrace{\sqrt{\frac{2 \ln t}{n_i}}}$
  - Pick arm $j = \arg\max_i UCB(i)$ and observe y_t
  - Set $n_j \leftarrow n_j + 1$ and $\hat{\mu}_j \leftarrow \hat{\mu}_j + \frac{1}{n_j}(y_t - \hat{\mu}_j)$
- "Optimism in the face of uncertainty"

Performance of UCB

- Theorem [Auer et al 2002]
  - Suppose the optimal mean payoff is $\mu^* = \max_i \mu_i$

    and for each arm let $\Delta_i = \mu^* - \mu_i$
  - Then it holds that

$$\mathbb{E}[R_T] = \underbrace{\left[8 \sum_{i:\mu_i<\mu^*} \left(\frac{\ln T}{\Delta_i} \right) \right]}_{O(k \ln T)} + \underbrace{\left(1 + \frac{\pi^2}{3} \right) \left(\sum_{i=1}^{k} \Delta_i \right)}_{O(k)}$$

$$\Rightarrow O\left(\frac{R_T}{T}\right) = \left(\frac{k \ln T}{T}\right)$$

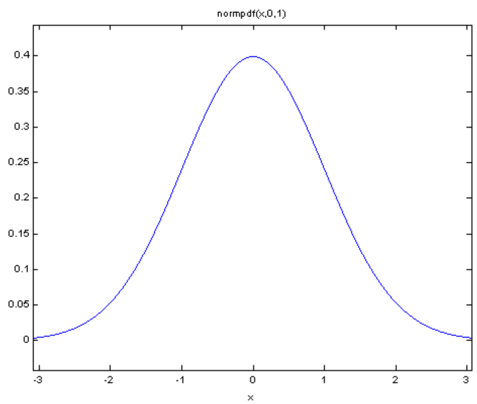
Challenges in recommendation

- Number of recommendations *k* to choose from large
  - Similar ads ➔ similar click-through rates!
- Performance depends on query / context
  - Similar queries ➔ similar click-through rates!
- Need to compile sets of *k* recs. (instead of only one)
  - Similar sets ➔ similar click-through rates!
- **Need to model and exploit "similarity"**

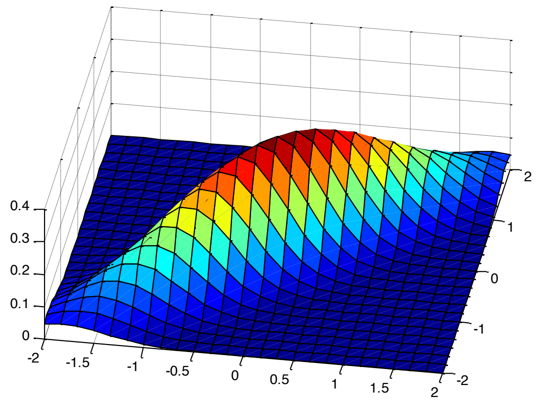
Stochastic ∞-armed bandits

- (Possibly infinite) Set X of choices
- Class *F* of functions on X
- Each choice x in X associated with (unknown) probability distribution P_x supported in [0,1] with means $\mu_x = f(x)$ for some $f \in F$
- Play game for T rounds
- In each round t, we pick an arm x, and obtain an random sample Y_t from P_x independent of previous samples
- Our goal is to maximize $\sum_{t=1}^{T} Y_t$

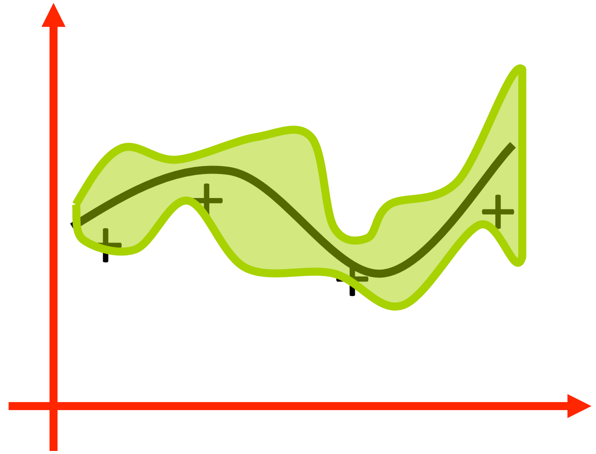
Gaussian Processes to model payoff *f*



Normal dist.
(1-D Gaussian)



Multivariate normal
(n-D Gaussian)



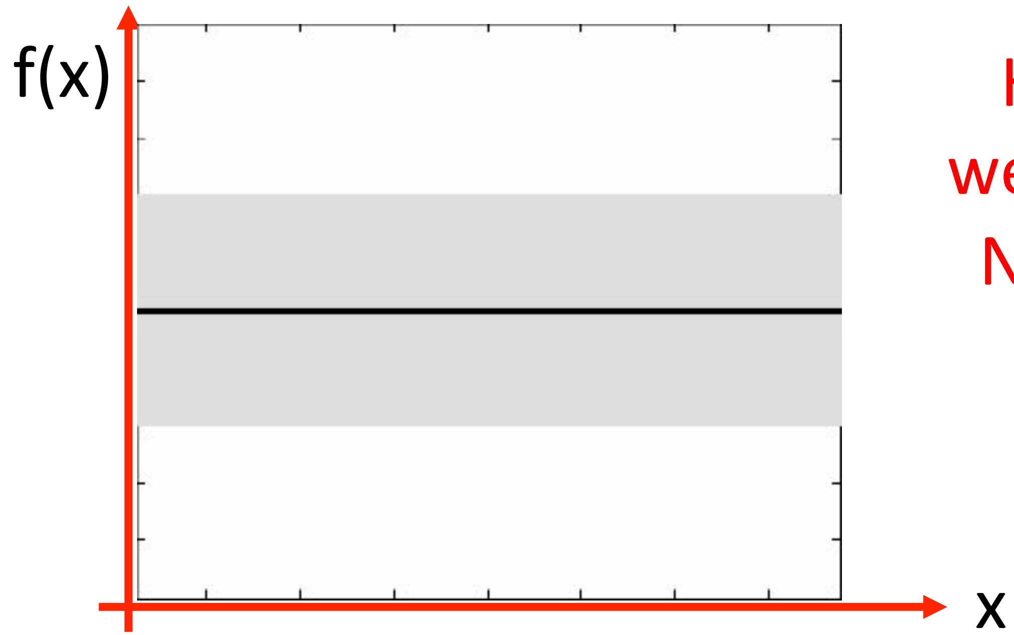
Gaussian process
(∞-D Gaussian)

- Gaussian process (GP) = normal distribution over *functions*
- Finite marginals are multivariate Gaussians
- Closed form formulae for Bayesian posterior update exist
- Parameterized by *covariance function K(x,x') = Cov(f(x),f(x'))*

Upper confidence sampling

Pick input that maximizes upper confidence bound:

$$x_t = \arg\max_{x \in D} \mu_{t-1}(x) + \beta_t \sigma_{t-1}(x)$$



How should we choose β_t?? Need theory!

Naturally trades off exploration and exploitation

Does not waste samples (with high prob.)

Guarantees for GP-UCB

Theorem: [Srinivas et al, ICML '10]

Choose $\beta_t = O(\log t)$. Then, for the following kernels:

- Linear: $$\frac{R_T}{T} = \mathcal{O}^*\left(\frac{d}{\sqrt{T}}\right)$$
- Squared-exponential: $$\frac{R_T}{T} = \mathcal{O}^*\left(\frac{(\log T)^{d+1}}{\sqrt{T}}\right)$$
- Matérn with $\nu > 2$, $$\frac{R_T}{T} = \mathcal{O}^*\left(T^{\frac{\nu+d(d+1)}{2\nu+d(d+1)}-1}\right)$$

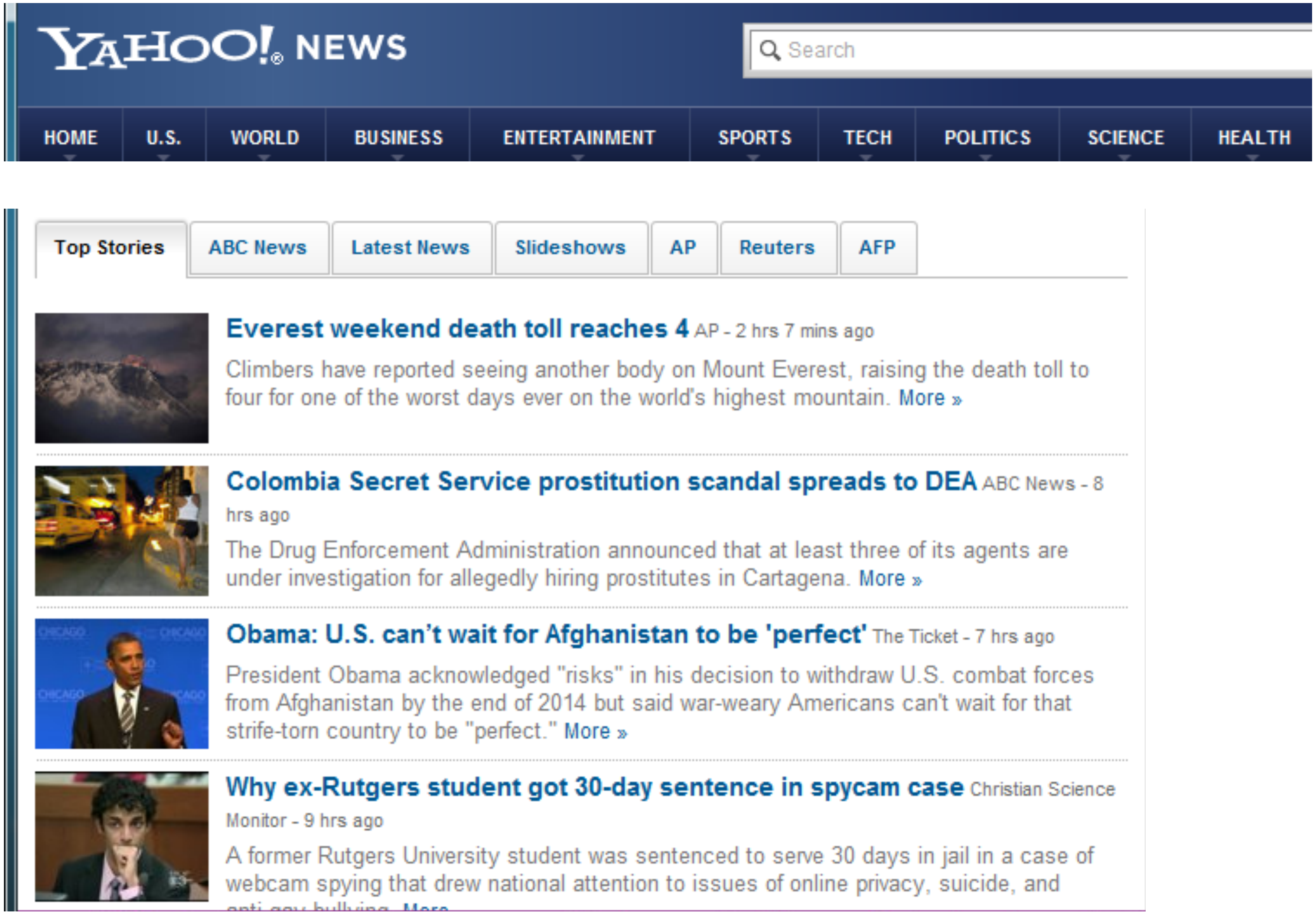
Bandits for recommendation

- Number of recommendations *k* to choose from large
  - Similar ads ➔ similar click-through rates!

- Performance depends on query / context
  - Similar queries ➔ similar click-through rates!

- Need to compile sets of *k* recs. (instead of only one)
  - Similar sets ➔ similar click-through rates!

Which news should we display?



LinUCB for personalized recommendation

[Li et al WWW '10]

- Every round receive context
  - User features (e.g., articles viewed before, …)
- Linear model for each article's click through rate

More generally: Contextual bandits

In each round t do:

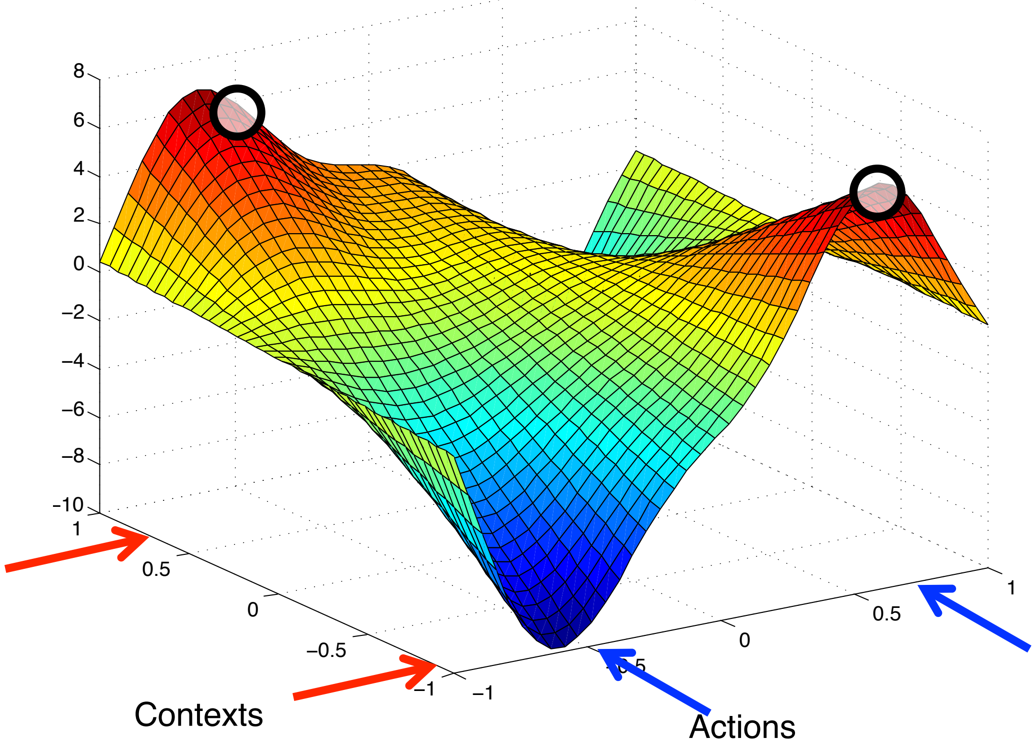
- Observe context $z_t \in Z$
- Pick $x_t \in D$
- Observe $y_t = f(x_t, z_t) + \epsilon_t$
- Incur regret

- Cumulative contextual regret $R_t = \sum_{t=1}^{T} r_t$
- Obtaining sublinear regret requires learning optimal mapping from contexts to actions!

CGP-UCB

Pick input that maximizes upper confidence bound
at current context

$$x_t = \arg\max_{x \in D} \mu_{t-1}(x, z_t) + \beta_t \sigma_{t-1}(x, z_t)$$

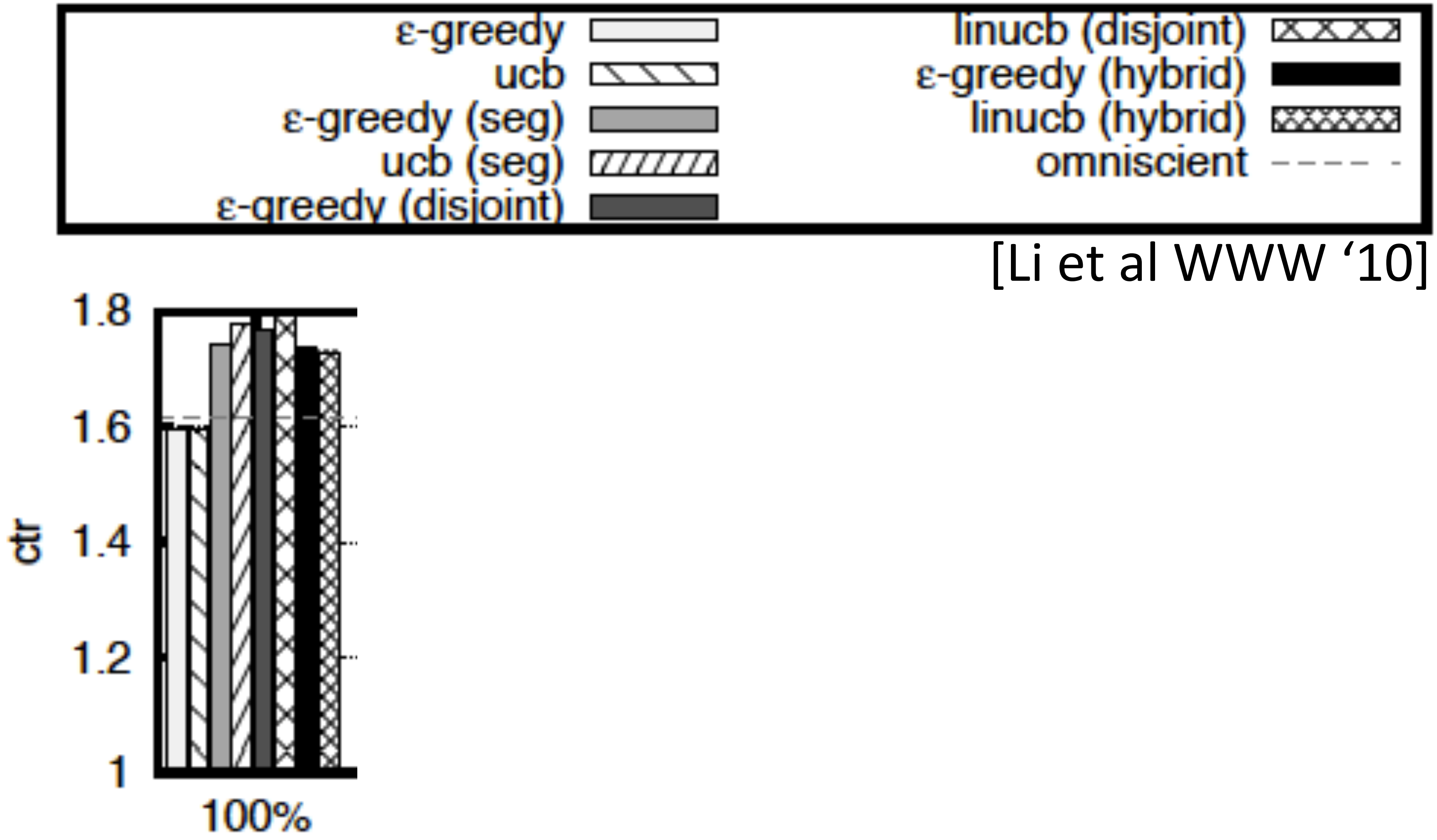


Similar regret bounds as in «context-free» setting.

Algorithm 1 LinUCB with disjoint linear models.

[Li et al '10]

Inputs: $\alpha \in \mathbb{R}_+$
for $t = 1, 2, 3, \ldots, T$ **do**
  Observe features of all arms $a \in \mathcal{A}_t$: $\mathbf{x}_{t,a} \in \mathbb{R}^d$
  for all $a \in \mathcal{A}_t$ **do**
    if a is new **then**
      $\mathbf{A}_a \leftarrow \mathbf{I}_d$ (d-dimensional identity matrix)
      $\mathbf{b}_a \leftarrow \mathbf{0}_{d\times 1}$ (d-dimensional zero vector)
    end if
    $\hat{\boldsymbol{\theta}}_a \leftarrow \mathbf{A}_a^{-1}\mathbf{b}_a$
    $p_{t,a} \leftarrow \hat{\boldsymbol{\theta}}_a^\top \mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^\top \mathbf{A}_a^{-1}\mathbf{x}_{t,a}}$
  end for
  Choose arm $a_t = \arg\max_{a\in\mathcal{A}_t} p_{t,a}$ with ties broken arbitrarily, and observe a real-valued payoff r_t
  $\mathbf{A}_{a_t} \leftarrow \mathbf{A}_{a_t} + \mathbf{x}_{t,a_t}\mathbf{x}_{t,a_t}^\top$
  $\mathbf{b}_{a_t} \leftarrow \mathbf{b}_{a_t} + r_t\mathbf{x}_{t,a_t}$
end for



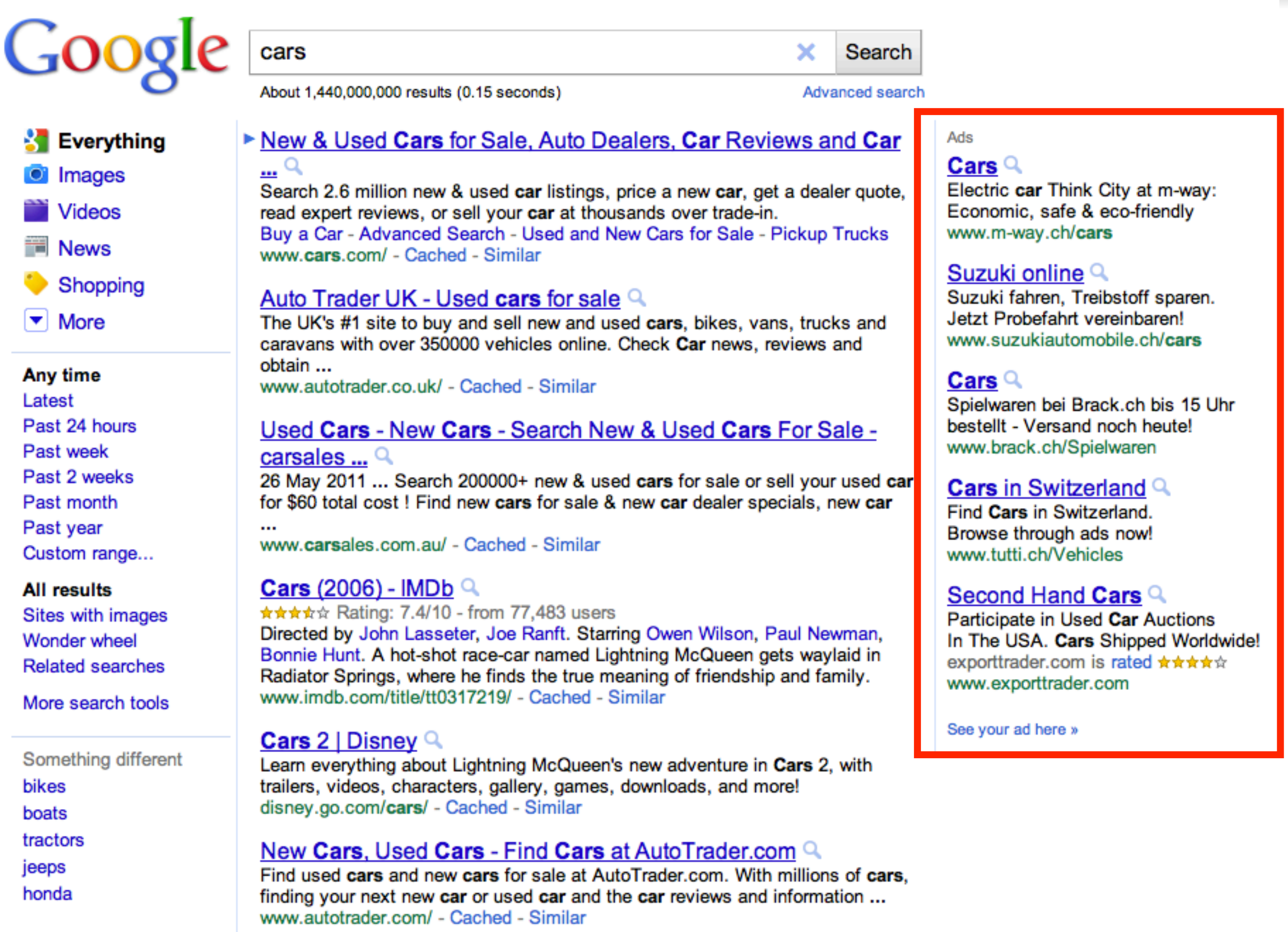
[Li et al WWW '10]

(a) CTRs in the deployment bucket.

Bandits for recommendation

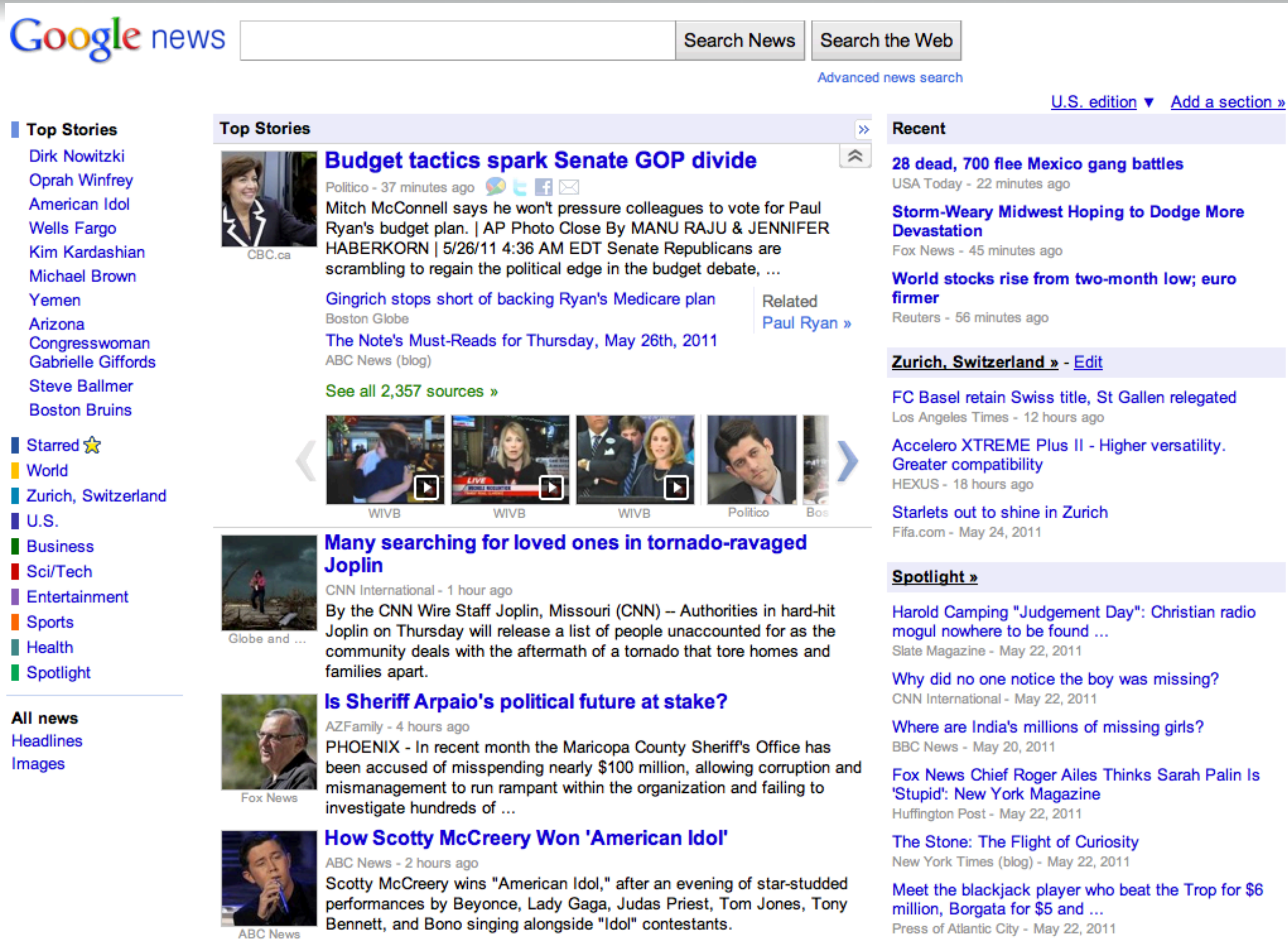
- Number of recommendations *k* to choose from large
  - Similar ads → similar click-through rates!
- Performance depends on query / context
  - Similar queries → similar click-through rates!
- Need to compile sets of *k* recs. (instead of only one)
  - Similar sets → similar click-through rates!

Sponsored search



Which **set of ads** should be displayed to maximize revenue?

News recommendation

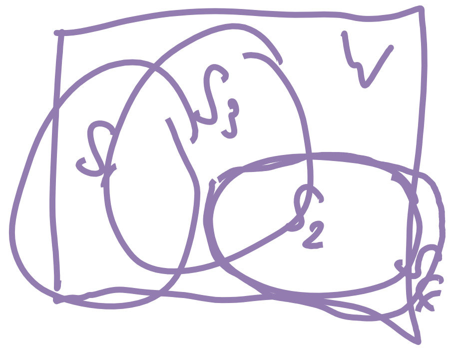


Relevance vs. Diversity

- Want to choose a set that caters to as many users as possible
- Users may have different interests / queries may be ambigious
- Want to optimize both relevance and diversity

Simple abstract model

- Suppose we're given a set W of users and a collection V of ads / documents
- Each ad i is relevant to a set of users $S_i \subseteq W$
- For each set of ads define



$$F(A) = \left| \bigcup_{i \in A} S_i \right|$$

- Want to select k ads to maximize "users covered"

$$\max_{|A| \leq k} F(A)$$

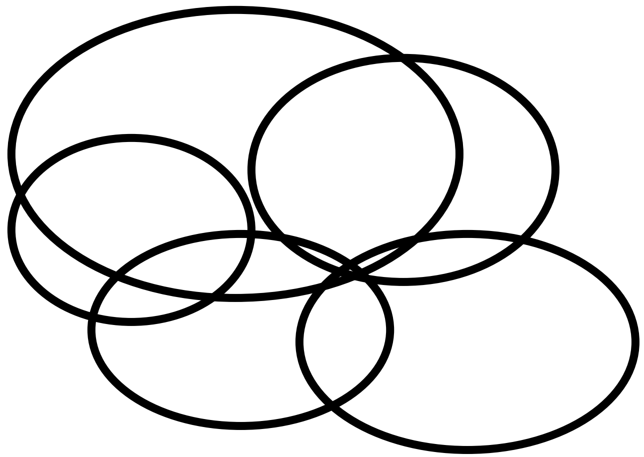
- Number of sets A grows exponential in k!
- Finding optimal A is NP-hard ☹

Maximum coverage

- Given: Collection V of sets, utility function F(A)
  Want: $A^* \subseteq V$ such that

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})$$

NP-hard!



Greedy algorithm:

Start with $A_0 = \{\}$

For i = 1 to k

$s^* = \text{argmax}_s\ F(A \cup \{s\})$

$A_i = A_{i-1} \cup \{s^*\}$

How well does this simple heuristic do?

Approximation guarantee

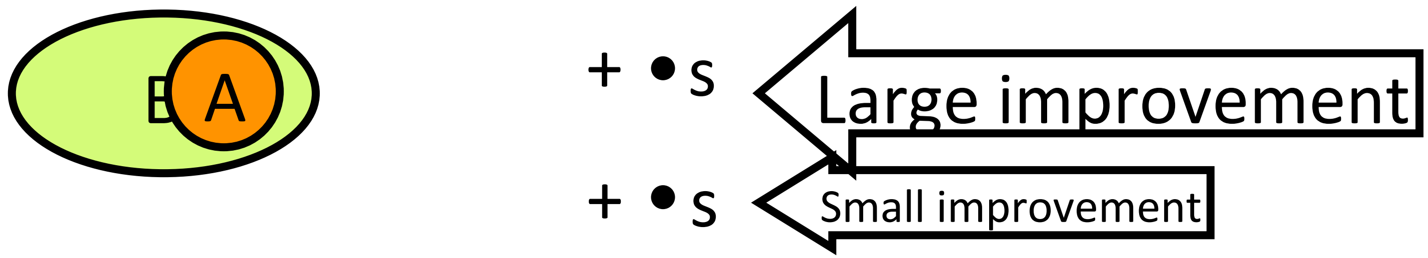
- **Theorem**: Greedy algorithm produces a solution A where F(A) ≥(1-1/e) of optimal value (~63%)
  [Nemhauser, Fisher, Wolsey '78]

- Claim holds for functions *F* with 2 properties:
  - *F* is monotone:
    if A ⊆ B then F(A) ≤ F(B) and *F*({})=0
  - *F* is submodular:
    adding element to a set gives less improvement than adding to one of subsets

Submodularity

- Diminishing returns
- Set function F on V is called submodular if:
  For all $A \subseteq B$, $s \notin B$:

$$\underbrace{F(A \cup \{s\}) - F(A)}_{\text{Gain of adding } s \text{ to a small set}} \geq \underbrace{F(B \cup \{s\}) - F(B)}_{\text{Gain of adding } s \text{ to a large set}}$$

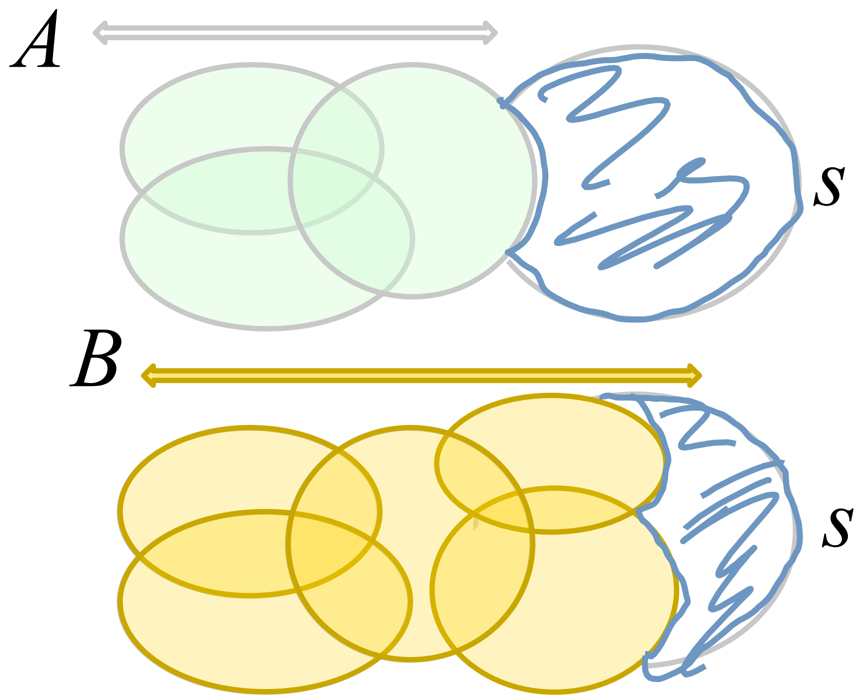


Example: Set cover

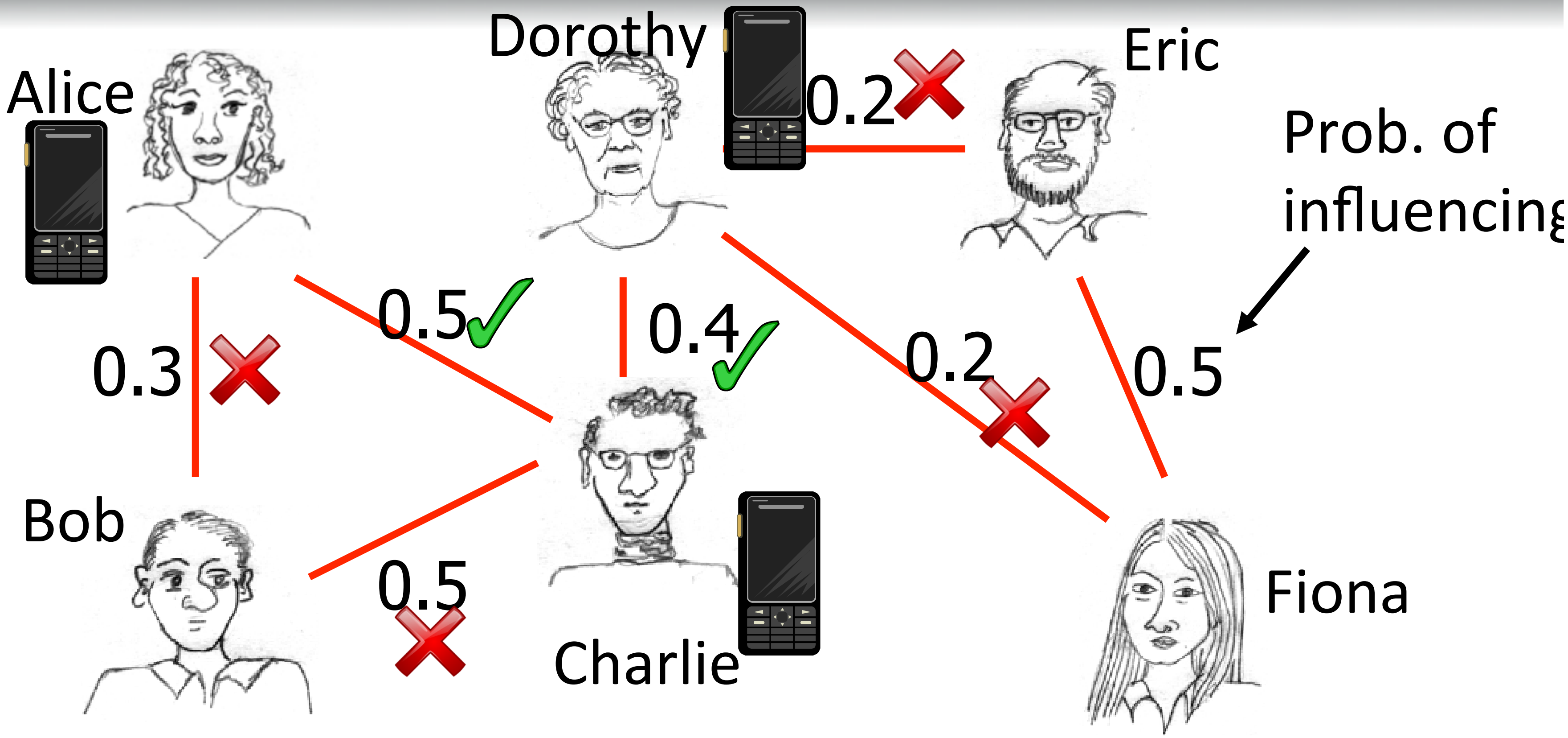
- F is submodular: $\boldsymbol{A} \subseteq \boldsymbol{B}$

$$\underbrace{F(A \cup \{s\}) - F(A)}_{\text{Gain of adding a set s to a small solution}} \geq \underbrace{F(B \cup \{s\}) - F(B)}_{\text{Gain of adding a set s to a large solution}}$$

- Natural example:
  - Sets $s_1, s_2, \ldots, s_n$
  - F(A) = size of union of s_i
    (size of covered area)



Example: Influence in social networks
[Kempe, Kleinberg, Tardos KDD ' 03]

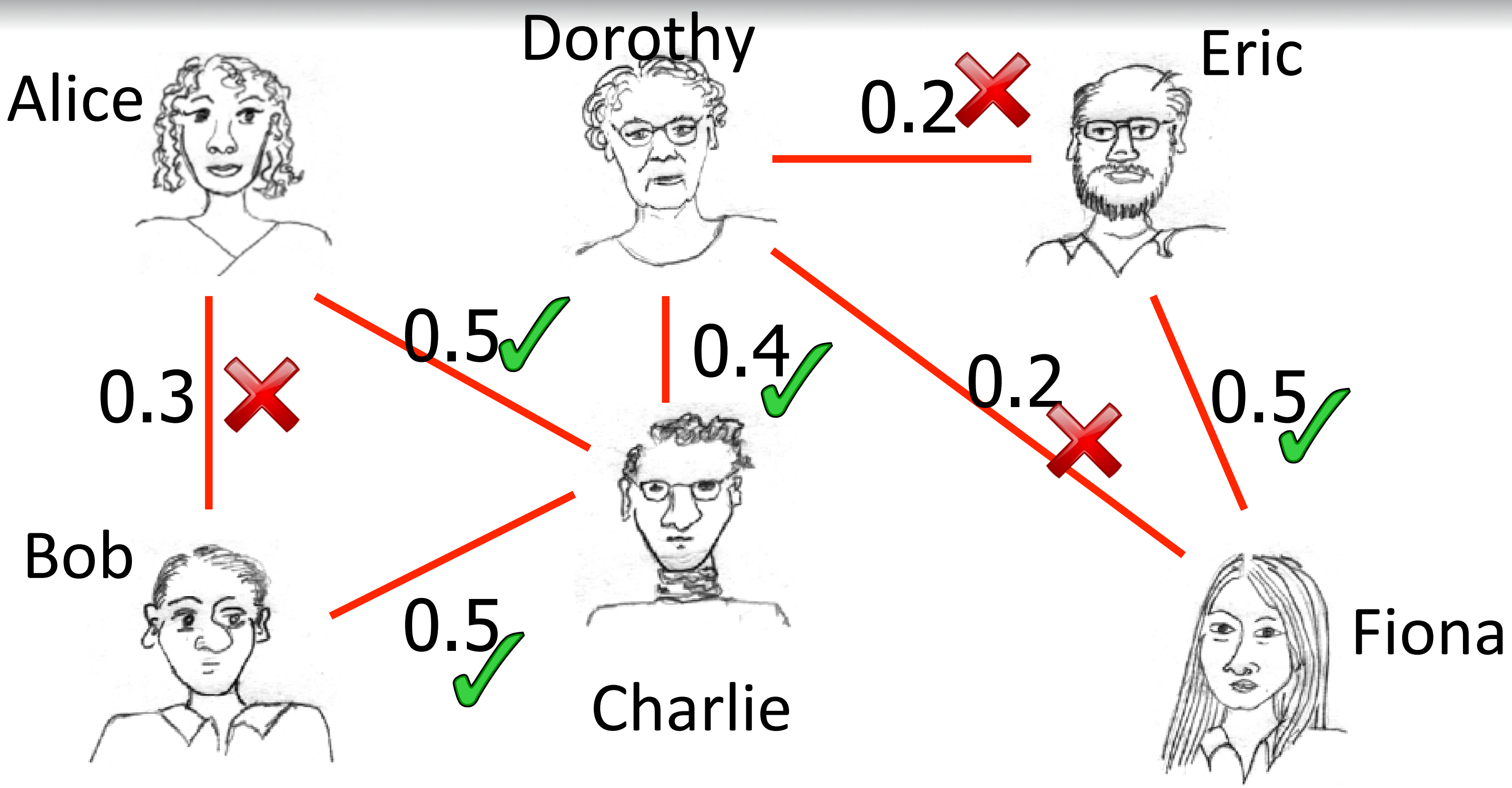


Who should get free cell phones?

V = {Alice,Bob,Charlie,Dorothy,Eric,Fiona}

F(A) = Expected number of people influenced when targeting A

Influence in social networks is submodular [Kempe, Kleinberg, Tardos KDD '03]



Key idea: Flip coins **c** in advance ➔ "live" edges

$F_c(A)$ = People influenced under outcome **c** (set cover!)

$F(A) = \sum_c P(c)\, F_c(A)$ is submodular as well!

Closedness properties

- $F_1,\ldots,F_m$ submodular functions on V and $\lambda_1,\ldots,\lambda_m \geq 0$
- Then: $F(A) = \sum_i \lambda_i F_i(A)$ is submodular!
- Submodularity closed under nonnegative linear combinations
- Extremely useful fact:
  - $F_\theta(A)$ submodular $\Rightarrow \sum_\theta P(\theta) F_\theta(A)$ submodular!
  - Multicriterion optimization: $F_1,\ldots,F_m$ submodular, $\lambda_i>0 \Rightarrow \sum_i \lambda_i F_i(A)$ submodular

Question

I have 10 minutes. Which blogs should I read to be most up to date?

[Leskovec-Krause et al. '07]

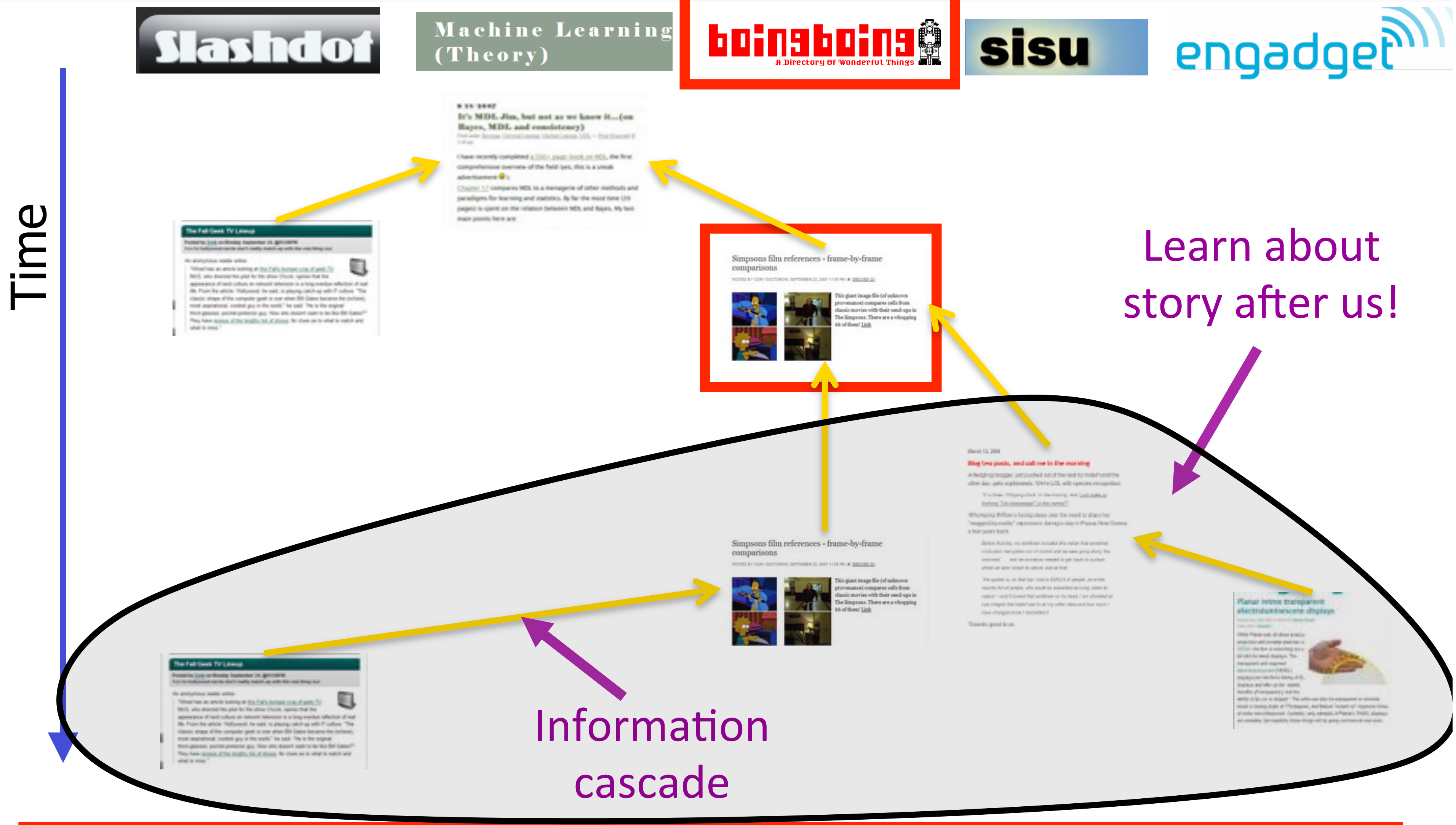
TIME
IN PARTNERSHIP WITH CNN

Thursday, Nov. 20, 2008

How Many Blogs Does the World Need?

By Michael Kinsley

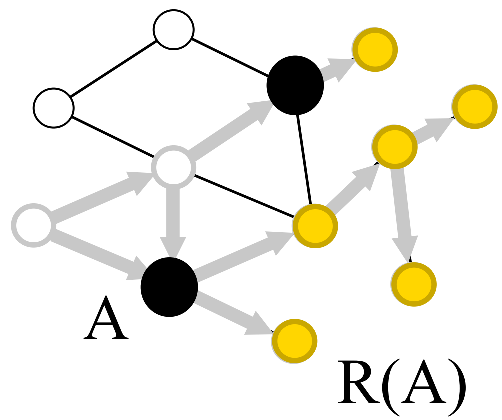
Detecting Cascades in the Blogosphere



Which blogs should we read to learn about big cascades early?

Modeling reward in information cascades

- Maximize the number of nodes that read the story after us:
  - If A are the monitored nodes, let $\mathrm{R(A)}$ denote the number of nodes we "beat"



Optimization problem

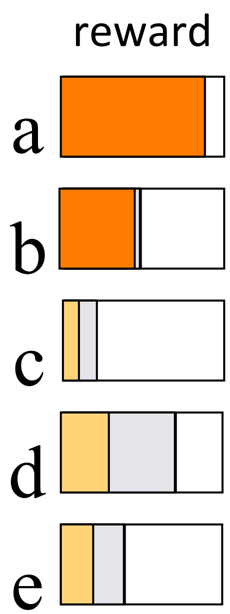
- Given:
  - Graph G(V,E), budget k
  - Data on how cascades 1,…,N spread over time
- Select a set of nodes A maximizing the reward

$$\max_{A\subseteq V} \overbrace{\sum_i \mathrm{Prob}(i)\, \underbrace{R_i(A)}_{\text{Reward for detecting cascade } i}}^{F(A)}$$

subject to $|A| \leq k$

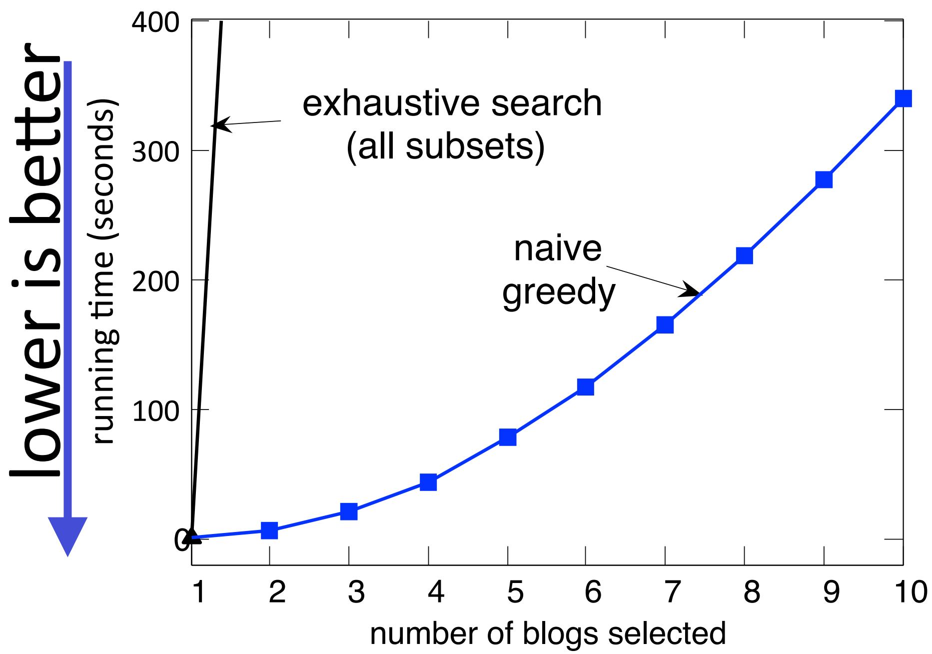
Greedy algorithm

Greedy algo



Add element with highest marginal gain

- Greedy algorithm is slow:
  - At each iteration we need to re-evaluate gains of all blogs
  - It scales as $O(n \cdot k)$



Scaling up greedy algorithm

- In round i+1:
  - have so far picked $A_i = \{s_1,\ldots,s_i\}$
  - pick $s_{i+1} = \text{argmax}_s\, F(A_i \cup \{s\})\text{-}F(A_i)$
    i.e., maximize "marginal benefit" $\delta_s(A_i)$
    $\delta_s(A_i) = F(A_i \cup \{s\})\text{-}F(A_i)$

$A_0 = \{\}$

$A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots \subseteq A_i$

$\delta_s(A_i) = F(A_i \cup \{s\}) - F(A_i)$

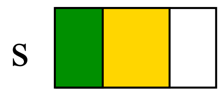
$\geq$

$\delta_s(A_j) = F(A_j \cup \{s\}) - F(A_j)$

- **Observation:** Submodularity implies
  $i \leq j \Rightarrow \delta_s(A_i) \geq \delta_s(A_j)$

Marginal benefits δ_s never increase!

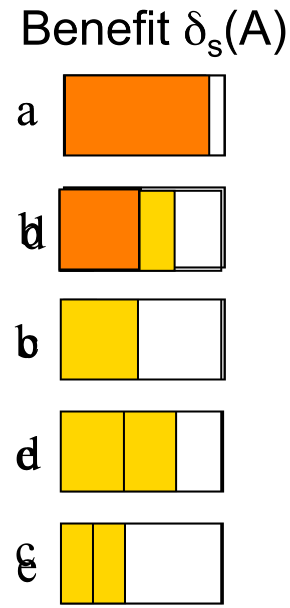
$\delta_s(A_i) \geq \delta_s(A_{i+1})$



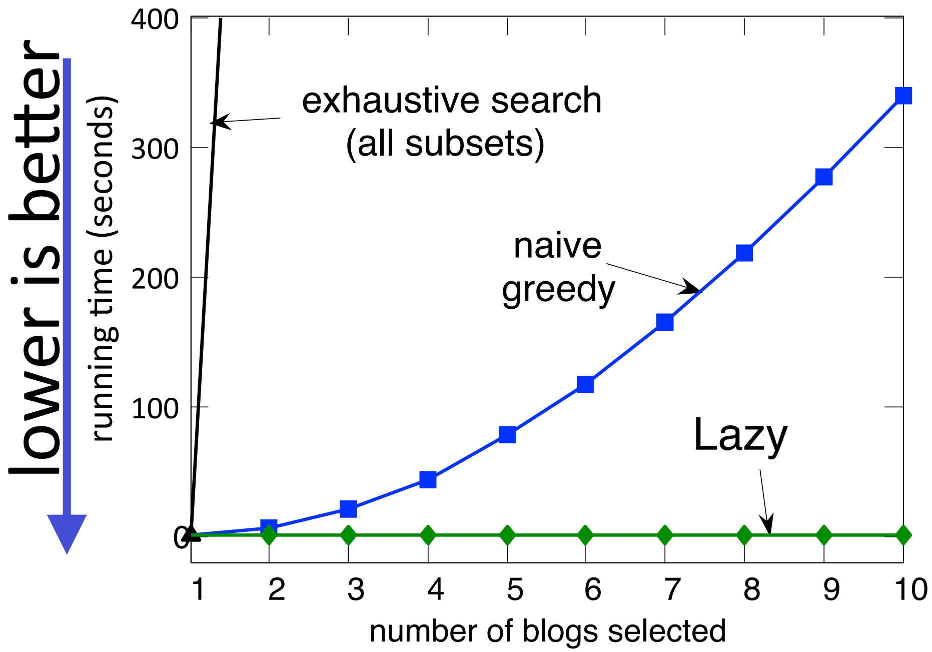
"Lazy" greedy algorithm

Lazy greedy algorithm:

- First iteration as usual
- Keep an ordered list of marginal benefits δ_i from previous iteration
- Re-evaluate δ_i only for top element
- If δ_i stays on top, use it, otherwise re-sort

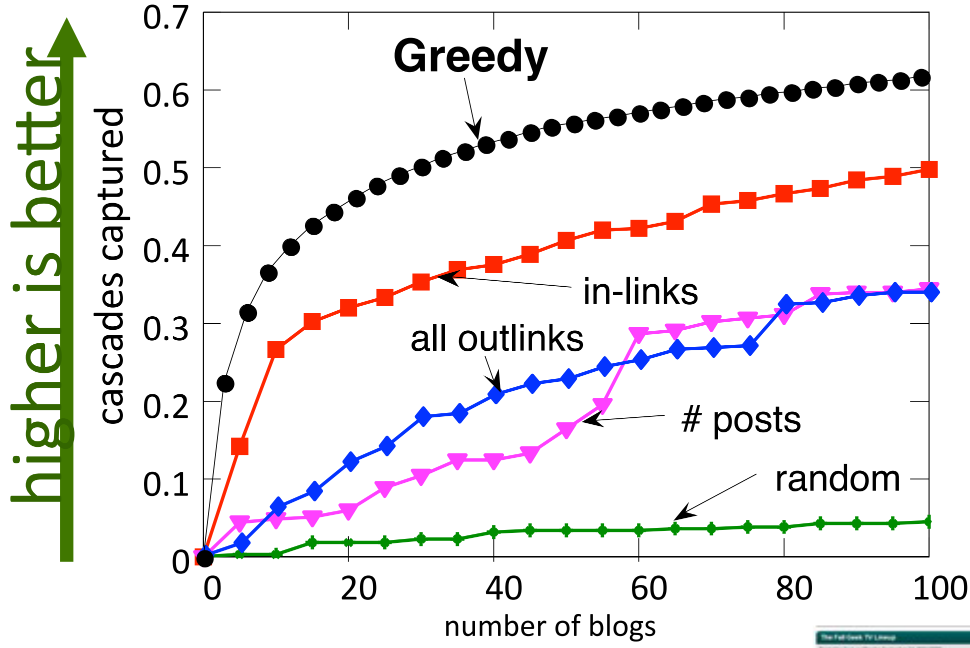


Result of lazy evaluation



- Using “lazy evaluations”
  - 700 times performance improvement!

Performance on Blog selection

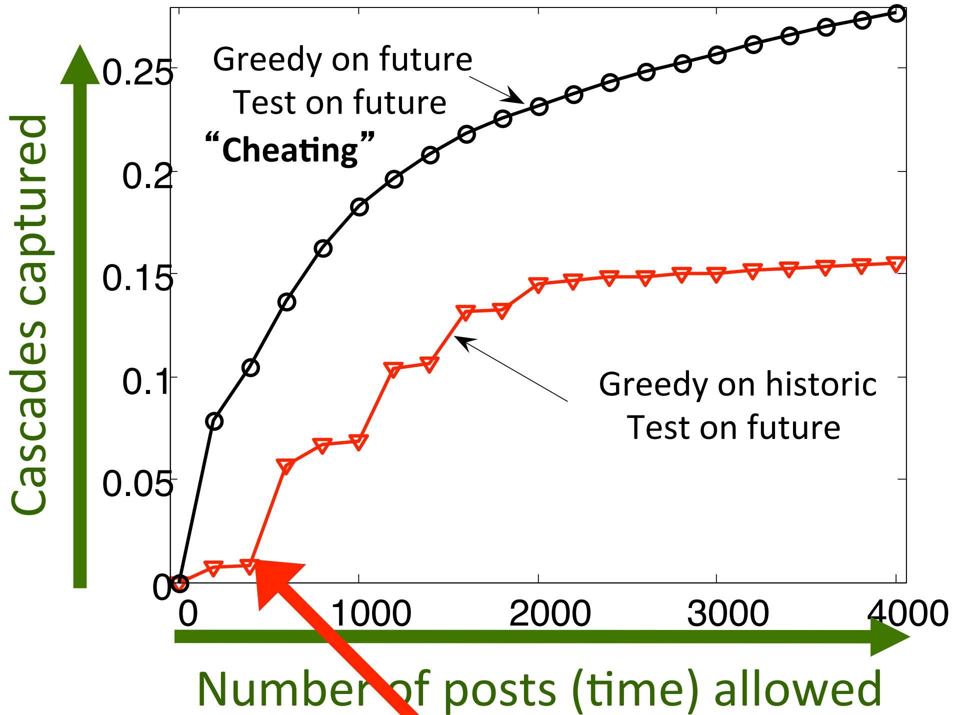


blog selection ~45k blogs

- Submodular formulation outperforms heuristics

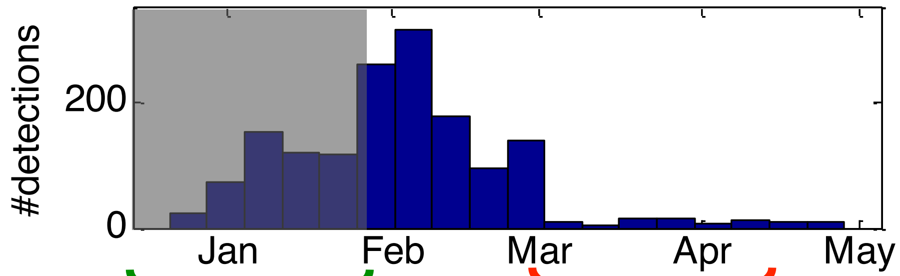
Predicting informativeness

- Want blogs that *will be* informative in the future
- Split data set; train on historic, test on future



Let's see what goes wrong here.

Detects on training set



Detect well here!

Detect poorly here!

Blog selection "overfits" to training data!

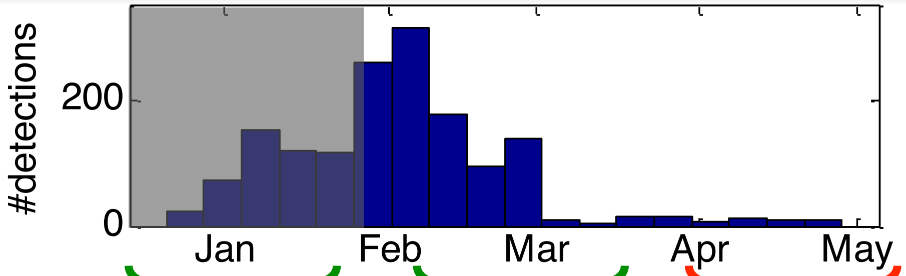
Poor generalization!

Why's that?

Want blogs that continue to do well!

Online optimization

"Overfit" blog selection **A**



$F_i(A)$ = detections in interval i

$F_1(A)$=.5 $F_3(A)$=.6 $F_5(A)$=.02

$F_2(A)$=.8 $F_4(A)$=.01

Online optimization: $$\max \sum_{t=1}^{T} F_t(A_t)$$

Online submodular maximization
[Streeter, Golovin NIPS '08]

Pick sets	A_1	A_2	A_3	…	A_T
SFs	F_1	F_2	F_3	…	F_T
Reward	$r_1=F_1(A_1)$	r_2	r_3	…	r_T

Total: $\sum_t r_t \rightarrow$ max

Time

This is an online learning / multi-armed bandit problem, with one arm for each set of blogs to recommend!

Number of arms is exponential!!

Online maximization of submodular functions

- Suppose we wish to pick k out of n documents
- Initialize k multi-armed bandit (MAB) algorithms

- OnlineGreedy algorithm: In each round do

  For i = 1:k
  - MAB i picks item x_i
  - Feed back reward $F_t(\{x_1, \ldots, x_i\}) - F_t(\{x_1, \ldots, x_{i-1}\})$

- Example: $F_t(A)$
  - Counts the number of clicks (0 or 1) on ads in A
  - Counts the cascades detected by blogs A
  - …

Online submodular maximization
[Streeter, Golovin NIPS '08]

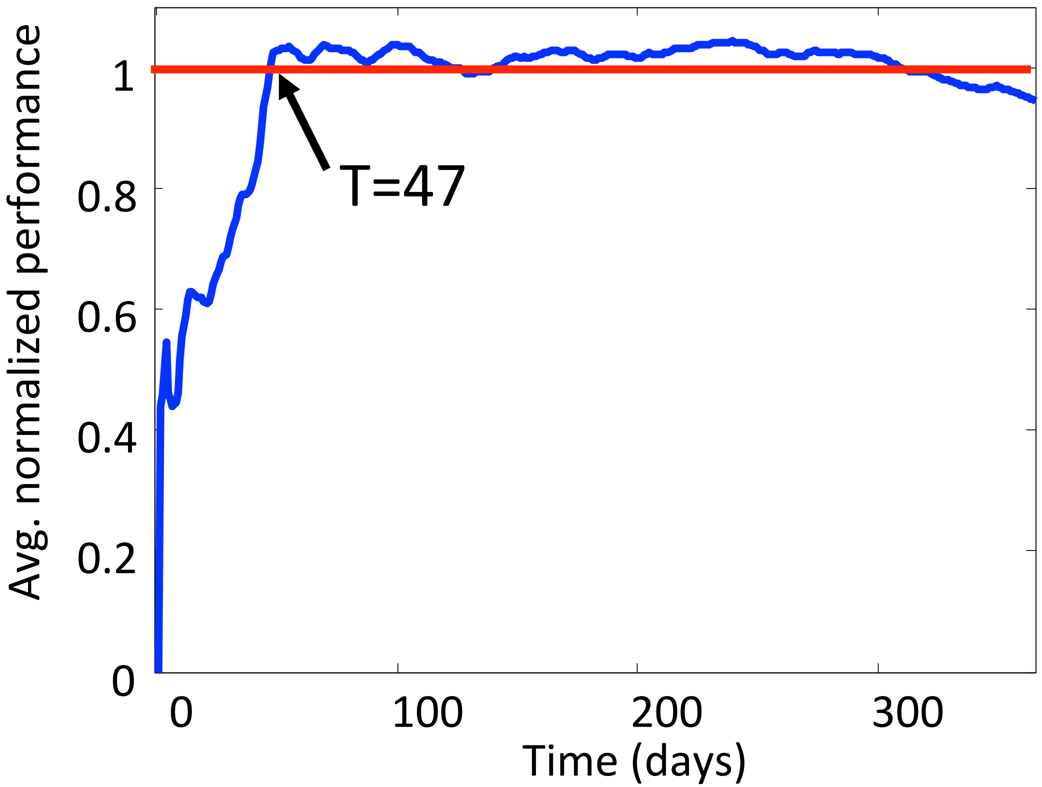
Theorem
OnlineGreedy chooses $A_1, \ldots A_t$ s.t. in expectation

$$\frac{1}{T}\sum_{t=1}^{T} F_t(A_t) \geq \frac{1-1/e}{T} \underbrace{\max_{|A|\leq k} \sum_{t=1}^{T} F_t(A)}$$

for any sequence F_i, as $T \rightarrow \infty$

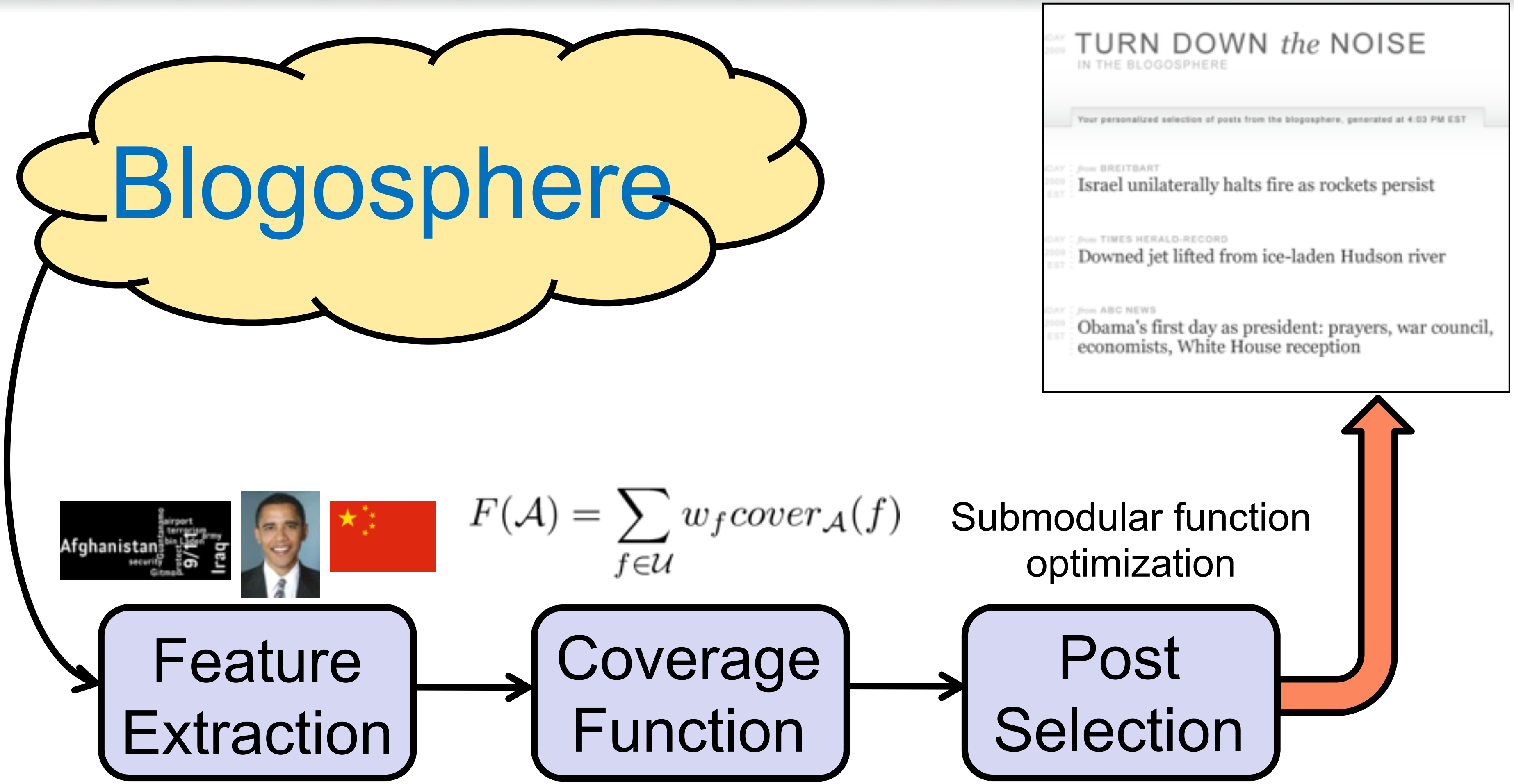
Can get 'no-regret' over "cheating" greedy algorithm

Results on blogs



Performance of online algorithm converges quickly to "cheating" offline greedy algorithm!

Turning down the noise in the blogosphere [El-Arini et al ‘09]



Evaluating Coverage [El-Arini et al '09]

- Evaluate on **real blog data** from Spinn3r
  - 2 week period in January 2009
  - ~200K posts per day (after pre-processing)

- Two variants of the algorithm

TDN+LDA: High level features
Latent Dirichlet Allocation topics

TDN+NE: Low level features

- User study involving 27 subjects to evaluate:

Topicality &
Redundancy

Topicality = Relevance to current events

The Washington Post

CNN npr

REUTERS

ESPN People

…

Reference Stories
(ground truth)

Downed jet lifted from ice-laden Hudson River

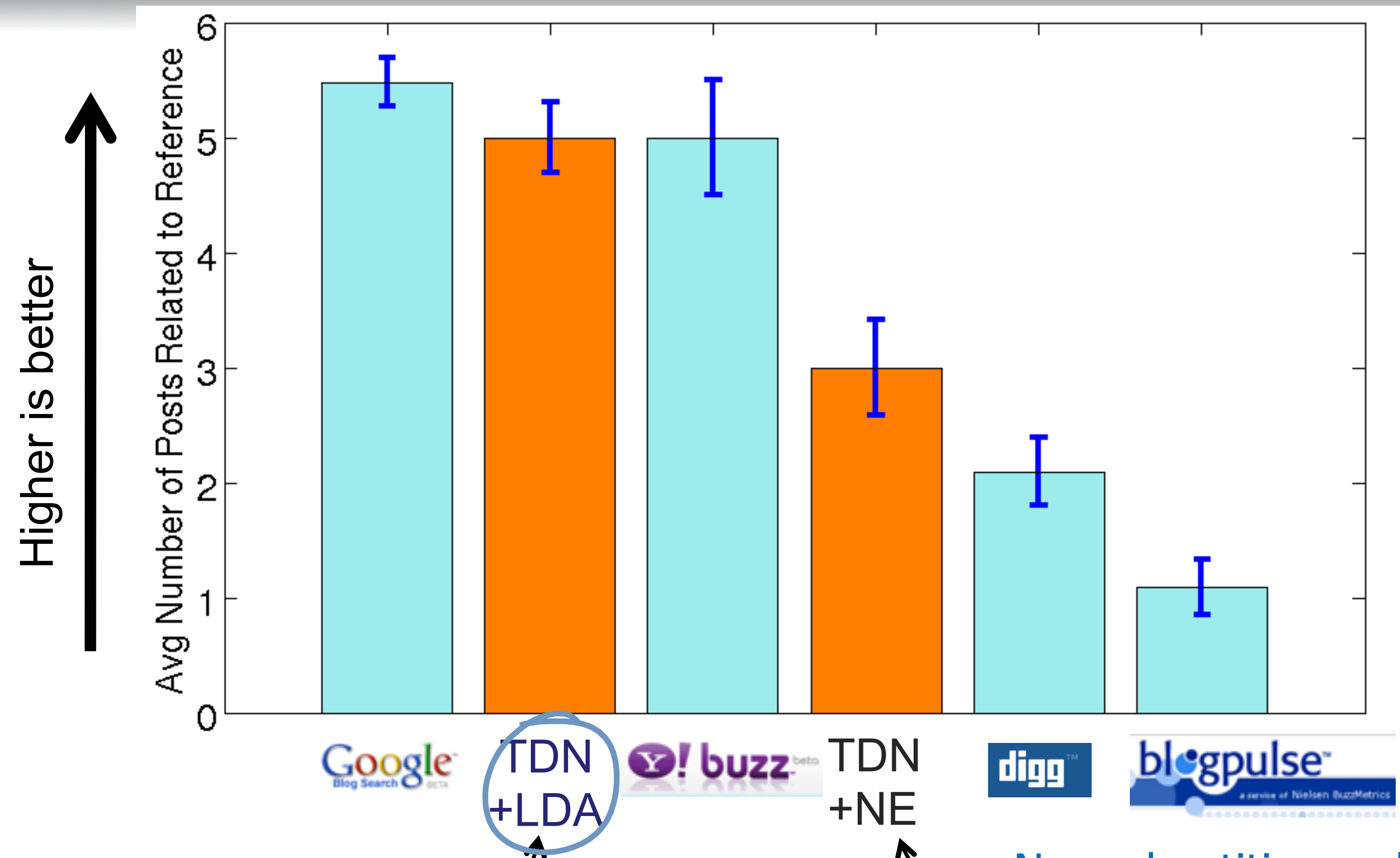
NEW YORK (AP) -
The airli at was
ng

Is this post topical? i.e., is it related to any of the major stories of the day?

Post for evaluation

[El-Arini et al '09]

Results: Topicality



LDA topics as features

Named entities and common noun phrases as features

[El-Arini et al '09]

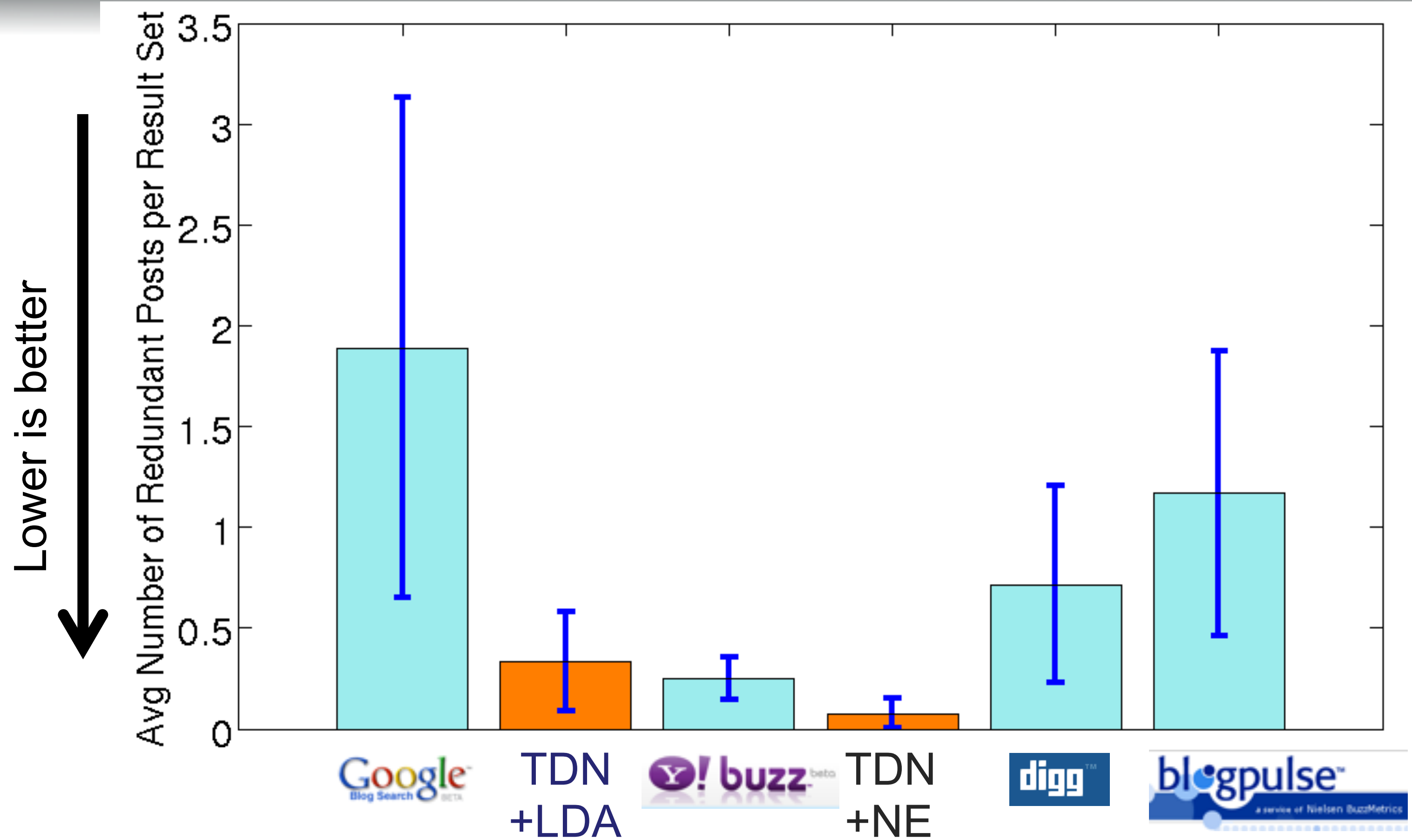
Evaluation: Redundancy

1. Israel unilaterally halts fire as rockets persist
2. Downed jet lifted from ice-laden Hudson River
3. Israeli-trained Gaza doctor loses three daughters and niece to IDF tank shell
4. ...

Is this post redundant with respect to any of the previous posts?

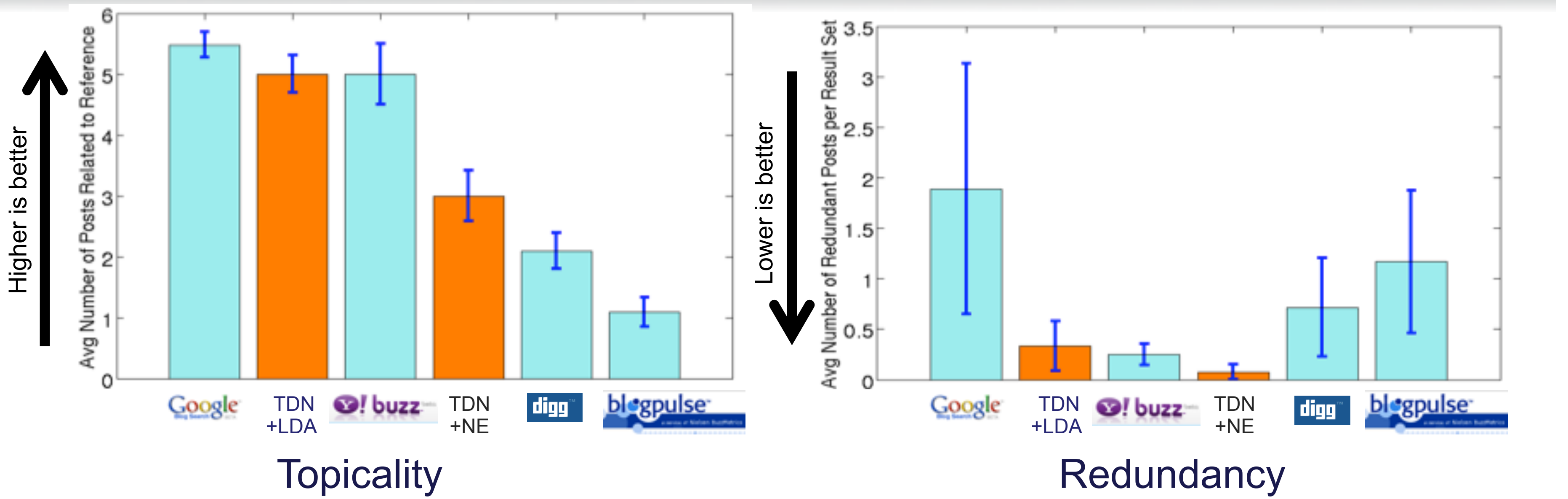
[El-Arini et al '09]

Results: Redundancy



[El-Arini et al '09]

Results: Coverage



- Google: good topicality, high redundancy
- Yahoo!: performs well on both, but **uses rich features**
  - CTR, search trends, user voting, etc.

[El-Arini et al '09]

Summary

- Submodular functions arise in many problems
- Simple greedy algorithm performs well
- Can combine with multi-armed bandit algorithms

You learned a lot

- **Retrieval**
  - Nearest-neighbor, min-hashing, locality sensitive hashing, …
- **Supervised learning**
  - Online SVM, online logistic regression, online convex programming, parallel online learning, …
- **Unsupervised learning**
  - Online k-means and EM, coresets, active learning, uncertainty sampling, informative sampling, …
- **Interactive data mining**
  - Multi-armed bandits, epsilon-greedy, UCB, GP optimization, submodular functions, …

Common insights

- Large data → New computing paradigm
  - Distributed / map-reduce style computations
  - Dealing with data streams
- Large data → Simple algorithms
  - Random sampling (hashing, dim. reduction, coresets, …)
  - Stochastic gradient descent (online SVM, online EM, …)
  - Greedy algorithms (UCB, submodularity)
  - In many cases, can prove that these simple algorithms work well (often the better the larger the data)
- Large data → Attention as a scarce resource
  - Need methods to cope with the information overload (active learning, exploration-exploitation tradeoffs, …)

Where to learn more

- Probabilistic Artificial Intelligence (Fall term)
  How can we build systems that
  - do well in unknown environments and unforeseen situations?
  - exhibit "intelligent" behavior, without explicit rules?
  - learn from experience in order to improve their performance?
  - Will cover modeling techniques and algorithms from statistics, optimization, planning, and control and study applications in areas such as sensor networks, robotics, and the Internet.
- Conference proceedings
  - Data Mining: KDD, SDM, ICDM, …
  - Machine learning: ICML, NIPS, AISTATS, …
  - Web mining: WWW, WSDM, …
- Master's thesis ☺

Acknowledgments

- These slides are partially based on material from Carlos Guestrin and Jure Leskovec