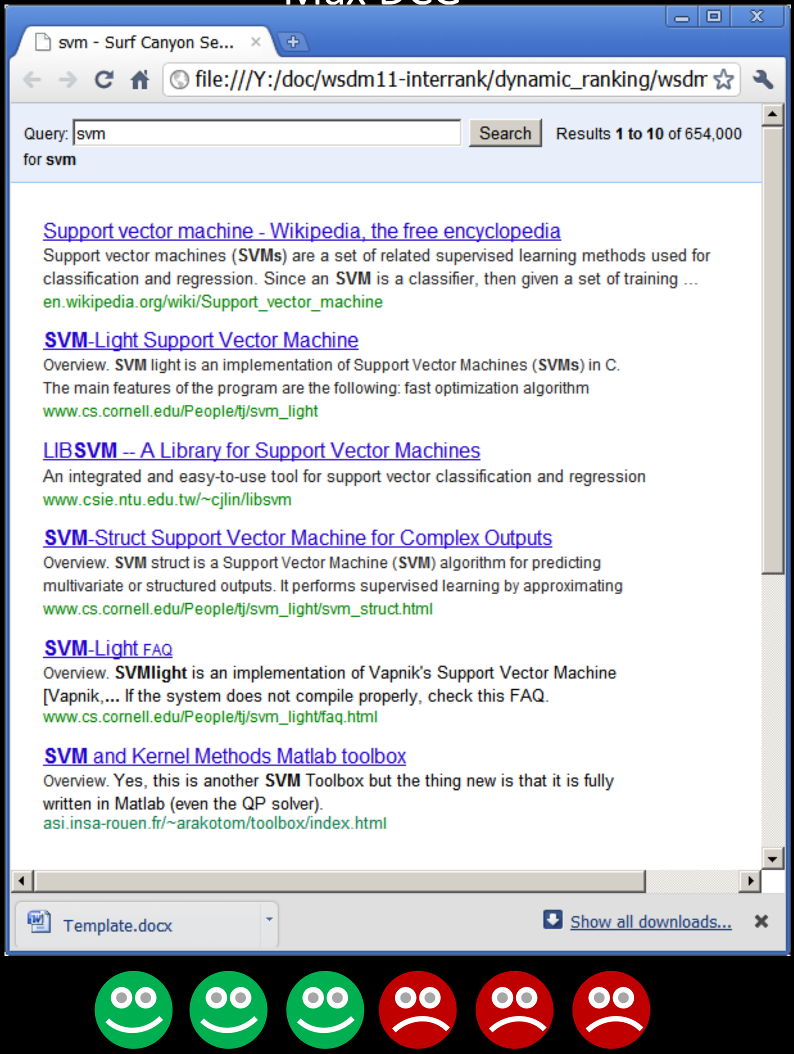
Dynamic Ranked Retrieval

Christina Brandt, Thorsten Joachims,
Yisong Yue, Jacob Bank

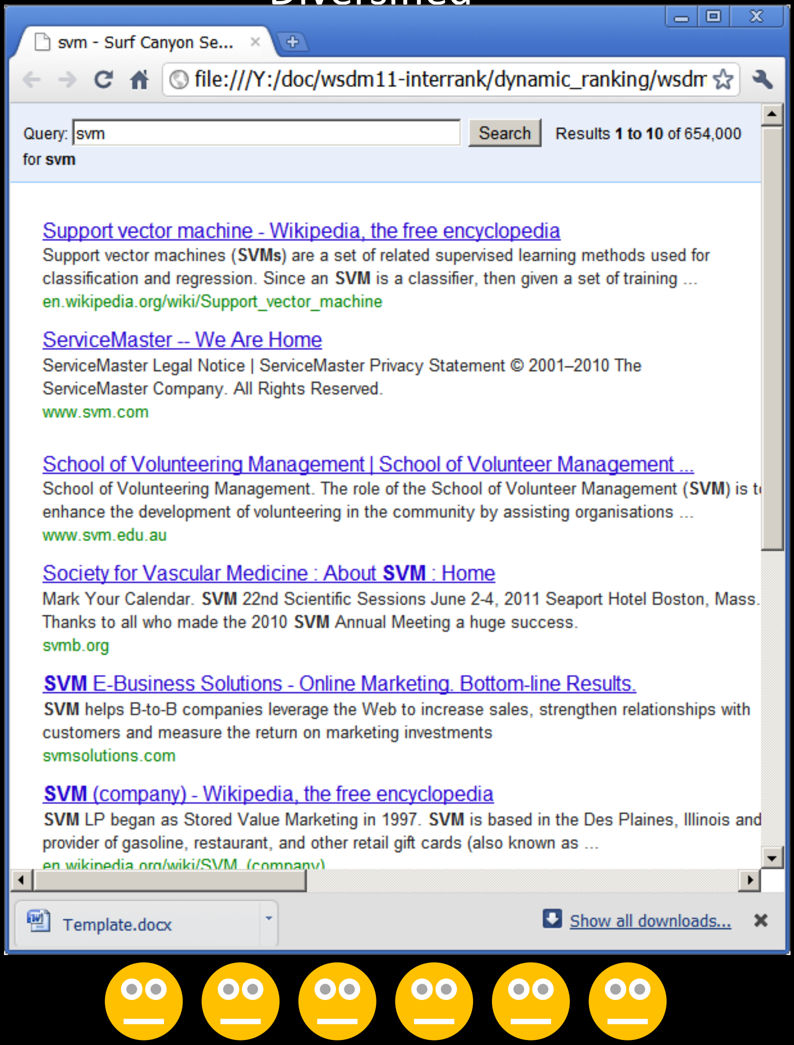
Department of Computer Science
Cornell University

Handling Query Ambiguity

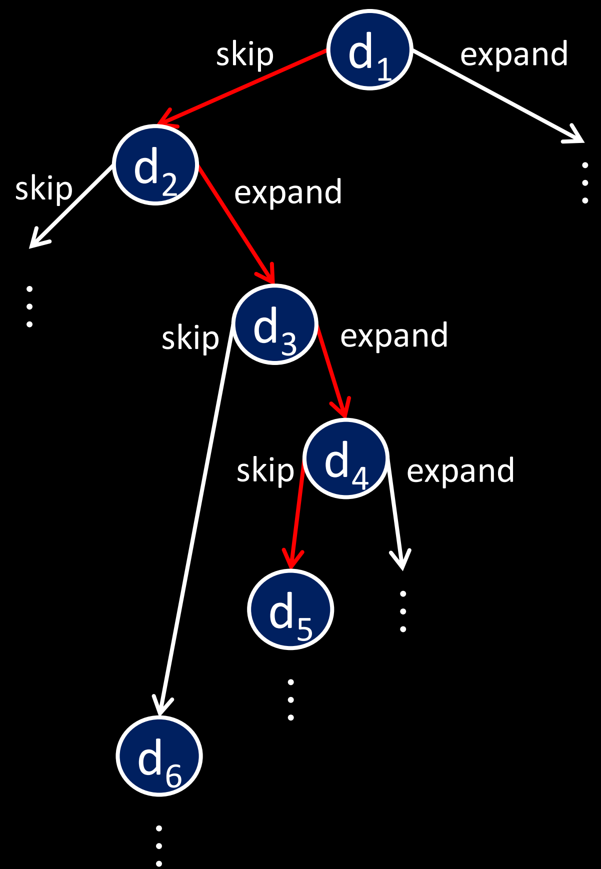
Max DCG

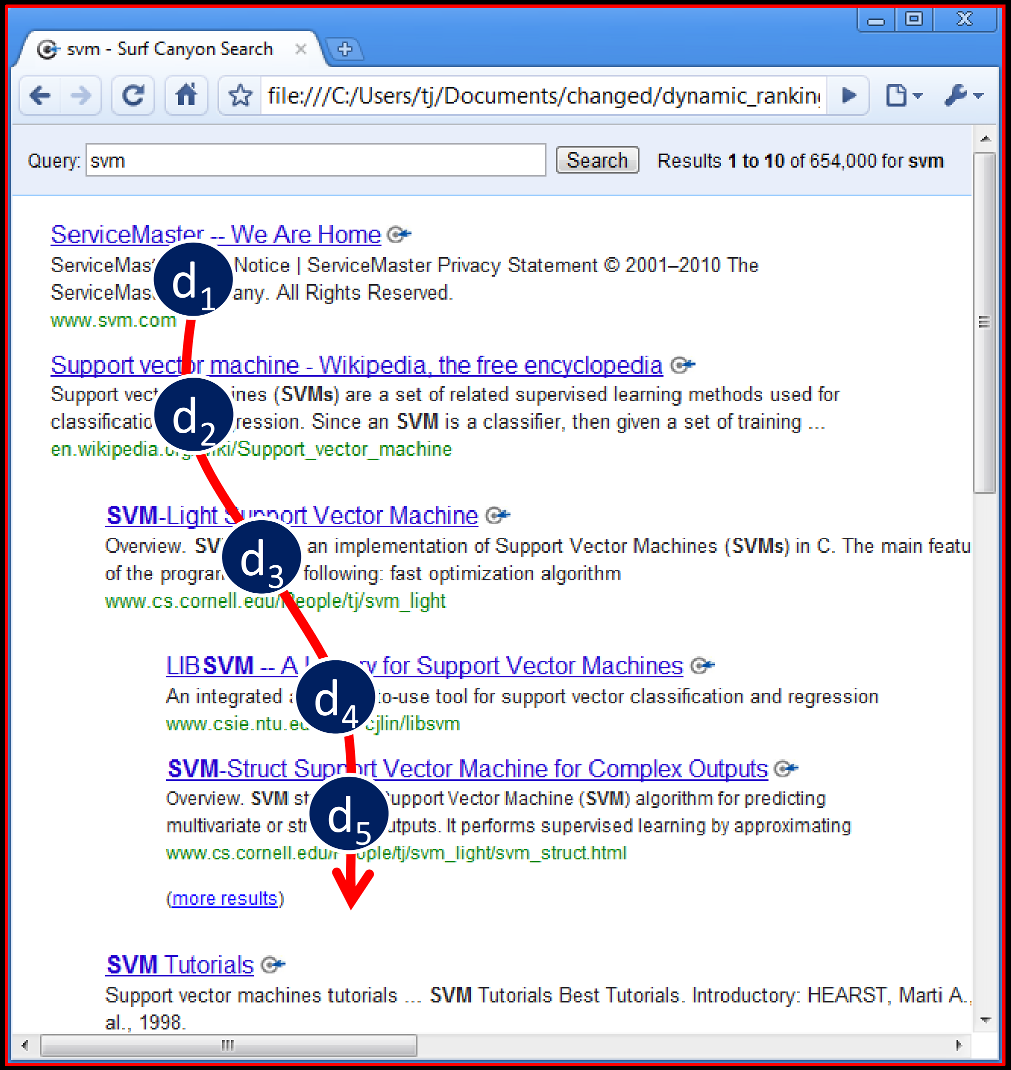


Diversified

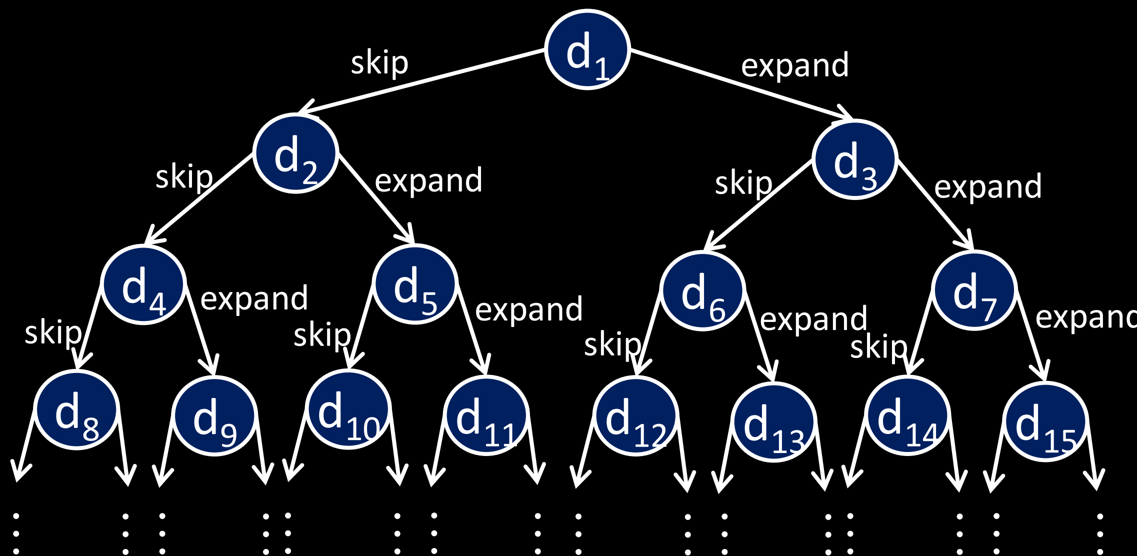


Dynamic Ranking





Dynamic Ranking: Model

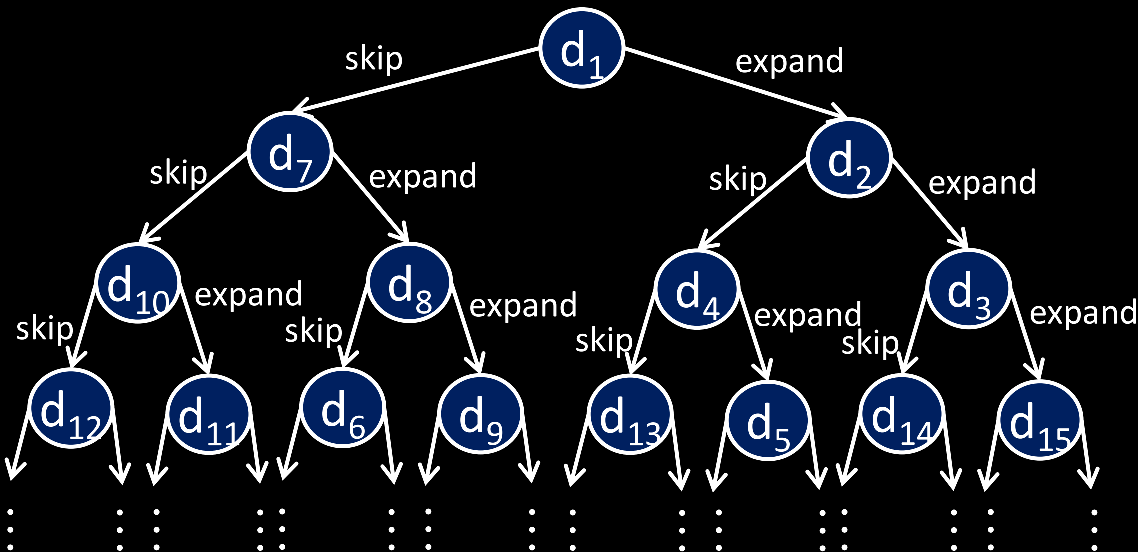


- Users: u_1 … u_k with P(u)
- User behavior: π: u £ $\Psi \rightarrow \sigma$
  $\rightarrow$ paths: σ_1 … σ_k
- Utility of σ for user u: U(u,σ)
- Overall utility: $U(\Psi) = \sum_u P(u)\, U(u,\pi(u,\Psi))$

Your favorite IR measure

"Intent-Aware" Measure [Agrawal et al. 2009]

Example



U(u,d)	u_1	u_2	u_3	u_4	u_5
d_1	1	1			
d_2	1				
d_3	1				
d_4		1			
d_5		1			
d_6			1		
d_7			1	1	
d_8				1	
d_9				1	
d_{10}					1
d_{11}					1
d_{12}					

Measure: $U(u,\sigma)=\sum_i 1/\log(i+1)\, U(u,\sigma[i])$

User behavior π: Expand if doc is relevant, else skip.

- **User 1**: $U(u_1,\sigma)=2.13$
- **User 2**: $U(u_2,\sigma)=1.93$
- **User 3**: $U(u_3,\sigma)=1.06$
- **User 4**: $U(u_4,\sigma)=1.56$
- **User 5**: $U(u_5,\sigma)=0.93$

$\Big\}$ $U(\Psi)=1.52$

Best static ranking σ^*: $U(\sigma^*)=0.84$

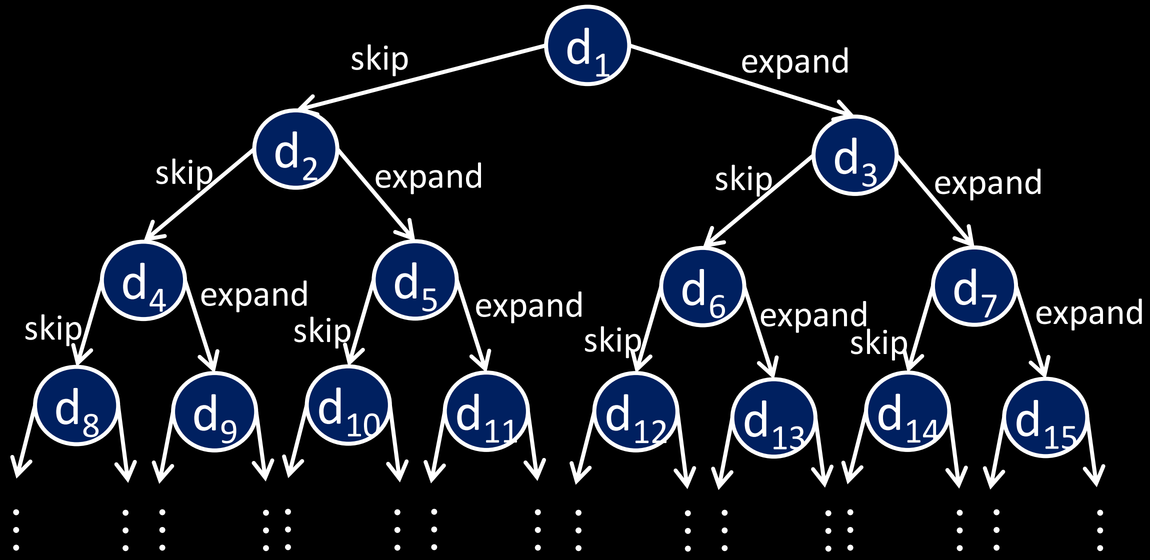
Assumption

- *Definition: Modular Utility Functions*

  A utility function $U(u,\sigma)$ is modular, if

  $$U(u,\sigma)=\sum_i \gamma[i]\ U(u,\sigma[i])$$

  with $\gamma[i] \geq 0$ and decreasing, and $U(u,\sigma[i]) \geq 0$.
  (e.g. DCG, nDCG, Prec@k)

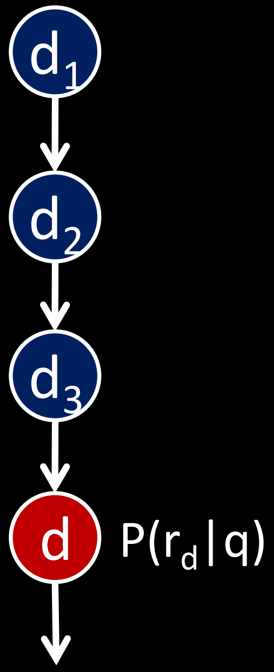


Algorithm for Static Ranking

- *Algorithm 3: StatMyopic(q, D, P(u))*
  - sort documents by expected doc utility
  - $P(r_d|q) = \sum_u P(r_d|q,u)\, P(u)$

- *Theorem: Optimality of StaticMyopic*

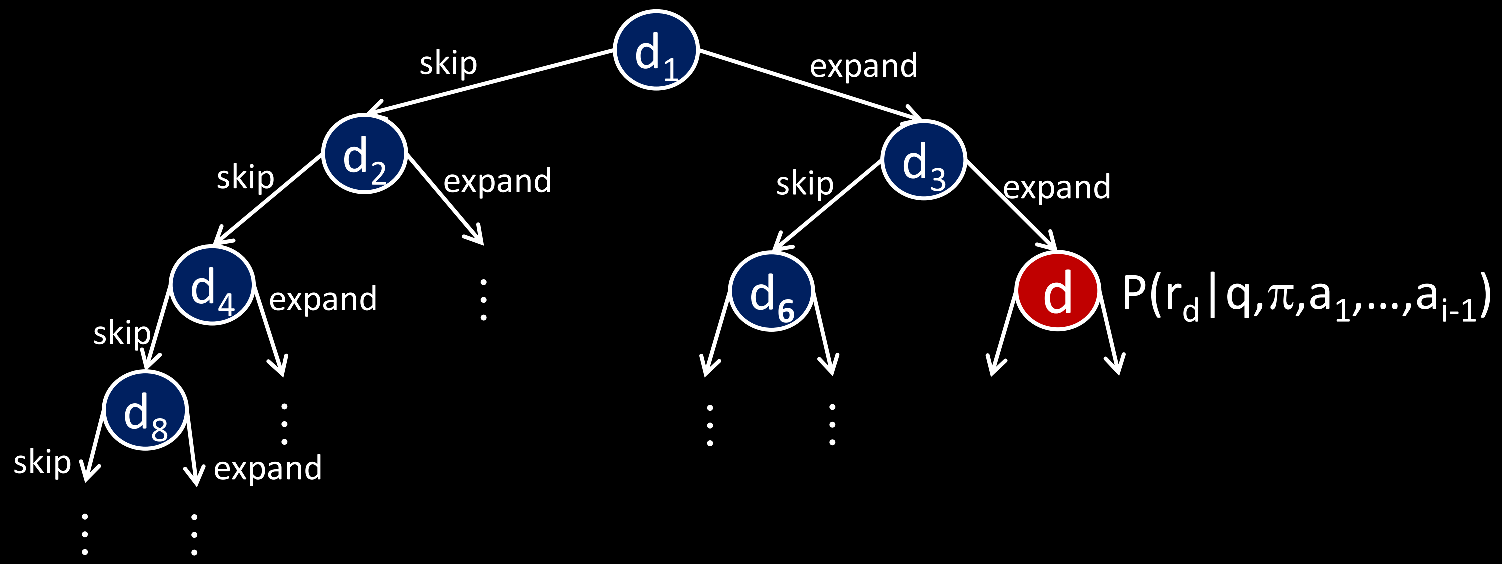
  For modular utility functions, StatMyopic is optimal for constructing static rankings.



… but it is **not** optimal for general utility functions, for example Average Precision.

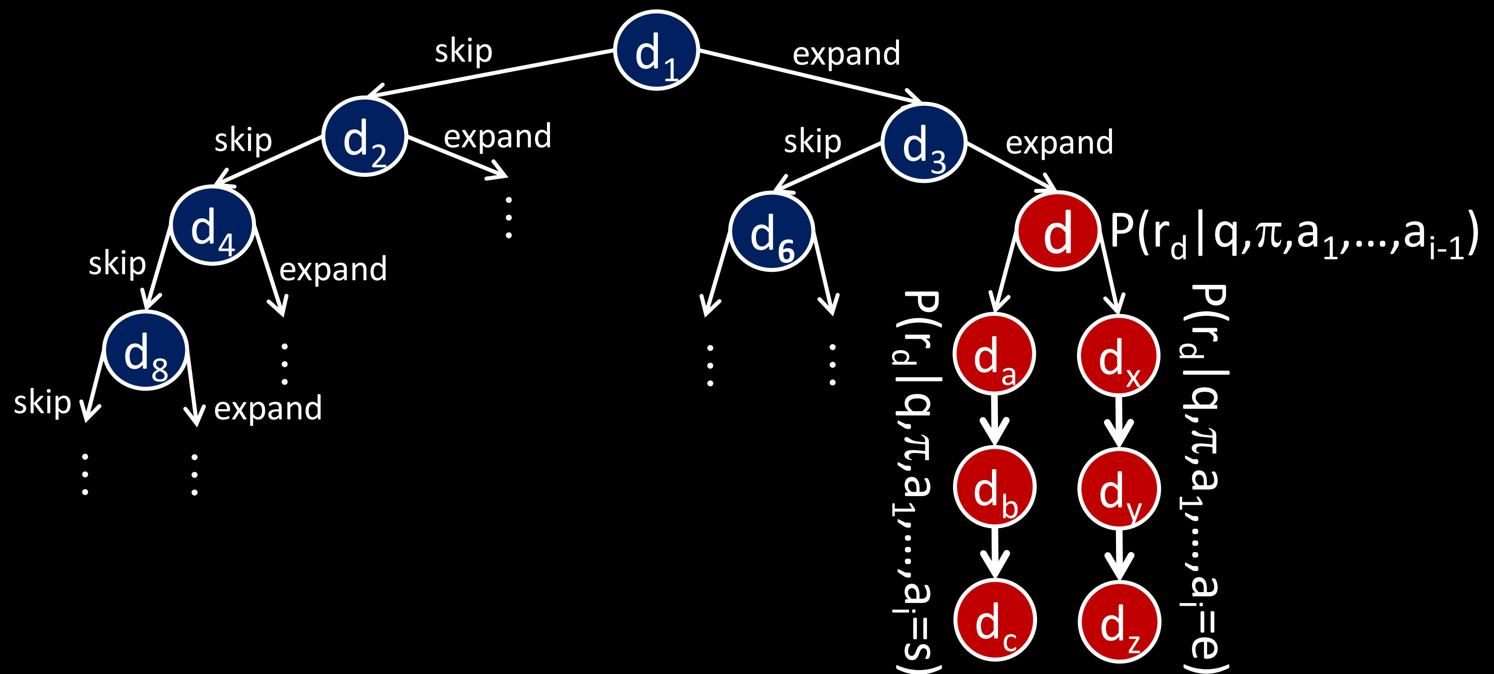
Algorithms for Dynamic Ranking

- *Algorithm 1: DynMyopic(q,D,π,P(u))*
  - Select d to maximize utility for user distribution that is conditioned on past actions
  - recurse on left and right subtree with user distribution updated according to π



Algorithms for Dynamic Ranking

- *Algorithm 2: DynLookahead(q,D,π,P(u))*
  - Select d to maximize conditional utility plus utility of static ranking for left and right subtree
  - recurse on left and right subtree with user distribution updated according to π



Adaptivity Gain

- *Definition: Adaptivity Gain*

  The adaptivity gain of a dynamic ranking algorithm A is

  $$U(A) - U(\text{StatMyopic}).$$

- *Theorem: Non-Negative Adaptivity Gain*

  For modular utility functions and any (known) user behavior policy π, both DynMyopic and DynLookahead have a non-negative adaptivity gain.

… it is possible to construct examples where adaptivity gain is 0, but where the best possible adaptivity gain > 0.

Related Work

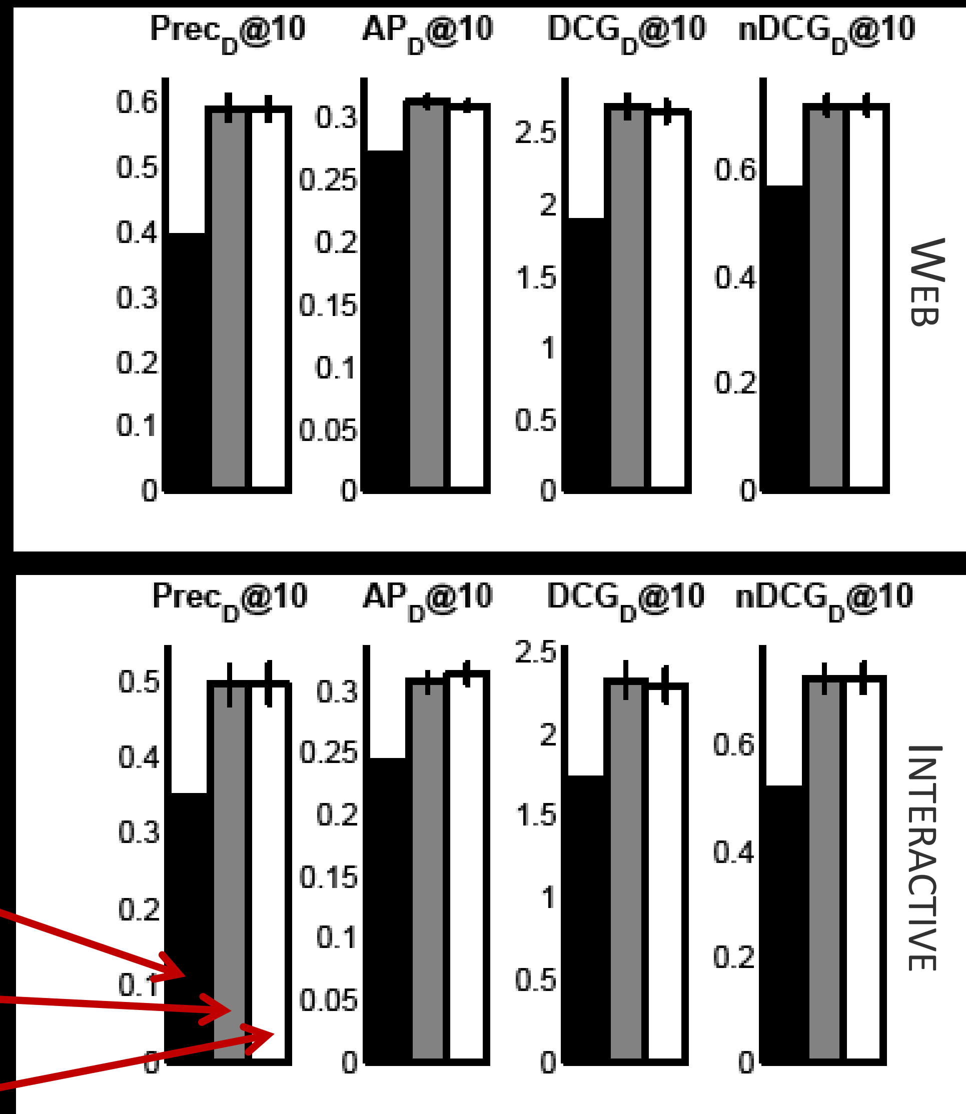
- Interactive Information Retrieval
  - Interface like SurfCanyon.com [Cramer et al., 2009]
  - Simple interaction, like menu browsing
  - Decision-theoretic model [Fuhr, 2008]
- Relevance Feedback
  - Use feedback to refine query, see [Ruthven & Lalmas, 2003]
  - Integrating feedback and result presentation; users “keep state”
  - Exploration, not just exploitation
- Diversified Retrieval
  - Novelty vs. relevance [Carbonell & Goldstein, 1998] [Zhai et al., 2003]
  - Left branch of DynMyopic similar to [Chen & Karger, 2006]

Experiments

- Data:
  - TREC 18 Web Track (WEB)
    - 50 queries
    - 2-8 profiles (i.e. user types) for each query.
    - P(u) uniform.
    - Binary relevance judgments
  - TREC 6-8 Interactive Track (INTERACTIVE)
    - 20 queries
    - 7-56 profiles (i.e. user types) for each query.
    - P(u) proportional to number of relevant documents in profile.
    - Binary relevance judgments

How Large is the Adaptivity Gain?

- Relevance profiles
  - known to the algorithm
- User Behavior
  - π_{det}: users expand relevant docs, skip non-relevant docs.



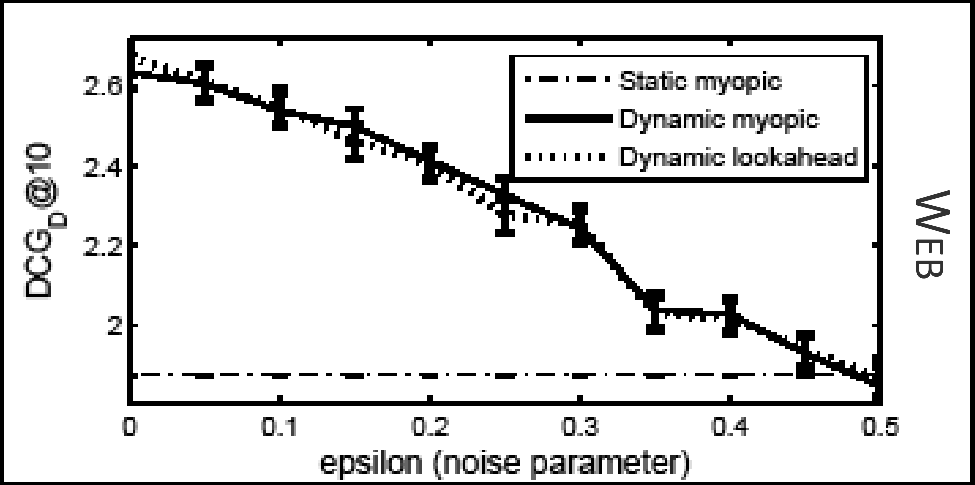
StatMyopic

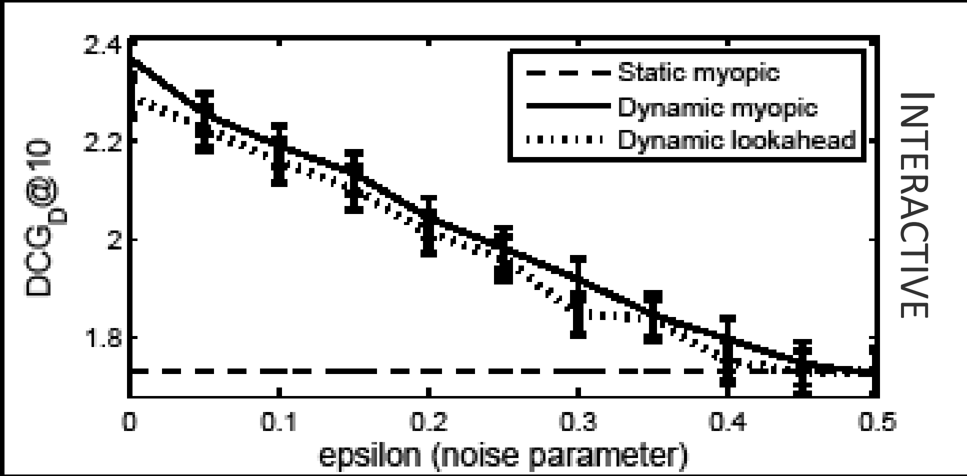
DynMyopic

DynLookahead

Noisy User Behavior

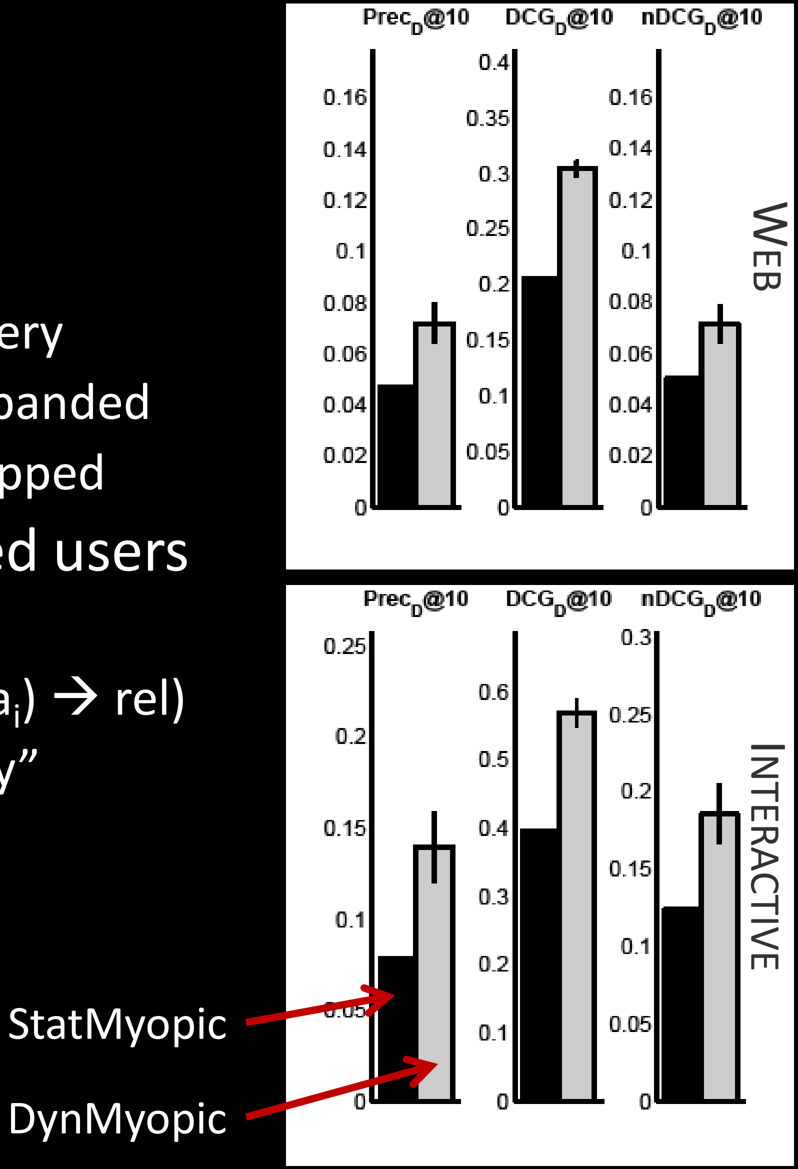
- Relevance profiles
  - known to the algorithm
- User Behavior
  - π_ε: users
    - skip relevant documents with probability ε,
    - exp non-relevant documents with probability ε.





Learning the Relevance Function

- Relevance profiles
  - Learned $P(rel|d,q,\pi,a_1 \ldots a_i)$
  - Logistic regression
    - Features relating doc and query
    - Features relating doc and expanded
    - Features relating doc and skipped
  - Training data from simulated users
    - π_{det}: deterministic users
    - Training examples: $((q,d,a_1 \ldots a_i) \rightarrow rel)$
    - Sampled $a_1 \ldots a_i$ "appropriately"
- Leave-one-query-out cross validation



Conclusions

- Simple interactive ranked retrieval model
  - Combines diversity and recall
  - Evolution, not revolution of user interface; menu browsing
- Theoretical model
  - Provides evaluation measures
  - Guides algorithm design
- Algorithms
  - Two efficient algorithms with non-negative adaptivity gain
  - `http://dynamicranking.joachims.org`
- Directions for Research
  - Approximation algorithms for constructing dynamic ranking trees
  - Using click data for learning $P(r_d|q,\pi,a_1,\ldots,a_{i-1})$
  - More sophisticated user behavior policies π
  - More actions than skip and expand
  - Usability (layout, clicking vs. mousing)

Directions for Research

- Approximation algorithms for constructing dynamic ranking trees
  - Approximation guarantees
  - Partial-information online learning problem
- Learning dynamic ranking models from real data
  - Using observational click data for learning $P(r_d | q, \pi, a_1, \ldots, a_{i-1})$
- More sophisticated user behavior policies π
  - Asymmetric costs for skip and expand / limited depth
  - Verify policy against real user behavior
  - Policies that are not static throughout search
- More actions than skip and expand
  - Backing out of tree
  - Reformulation recommendations
- Usability
  - Result layout
  - Clicking vs. mousing