Learning User Interaction Models for Predicting Web Search Result Preferences

Eugene Agichtein Eric Brill Susan Dumais Robert Ragno

Presented by

JiaLei Wang Seok Hyun Jin

Introduction

- Traditional approach to ranking for web search
  - Features that describe a candidate page
  - Supervised learning methods
  - Dependent on explicit relevance
- Use ***implicit relevance feedback***
  - Clickthrough data
  - Scroll time
  - Reading time

- **How can we model user's behavior? Which implicit features correlate to explicit ratings?**
- **Given implicit feedback, how can we effectively use them to produce reliable preference?**

Introduction : Limitations of Existing Methods

- Don’t make extensive use of ***implicit feedback***
  - Clickthrough, dwell time
  - Cheap and abundant
- Don’t necessarily generalize well for real-world web search
  - Web search is not controlled
    - “Users” may act irrationally, maliciously or may not even be human
      - Not all users are “experts”

Introduction :
How can we address these limitations?

- **How can we model user behavior? Which implicit features correlate to explicit ratings?**
- **Given implicit features, how can we effectively use them to determine preference?**

- Use of a distributional model of user behavior
  - Aggregated behavior of large number of users
  - Allows self-correct for noise

- Extension of strategies to include richer set of features
  - Partial to more descriptive model of user behavior
    - Pre and Post-search user behavior

Learning User Behavior Model

- As we noted earlier, real web search user behavior can be “noisy”.
- Hence, instead of treating each user as a reliable “expert”, we use statistics to infer relevance information from many unreliable data of user inputs.
- Approach: Model user web search behavior as :

relevance information	+	background noise	=	user behavior

Learning User Behavior Model : Case study in click distributions

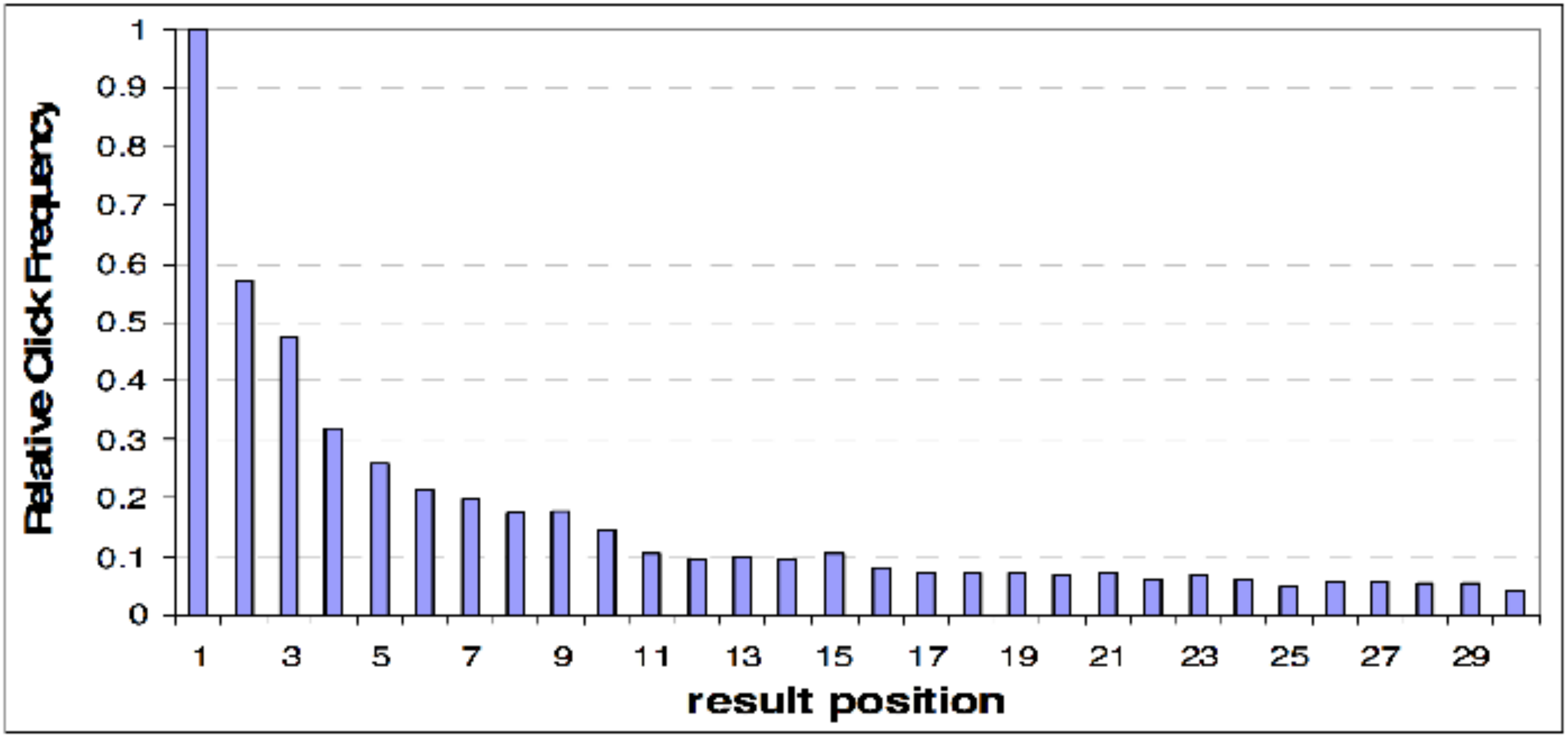


Figure 3.1: Relative click frequency for top 30 result positions over 3,500 queries and 120,000 searches.

Learning User Behavior Model : Case study in click distribution

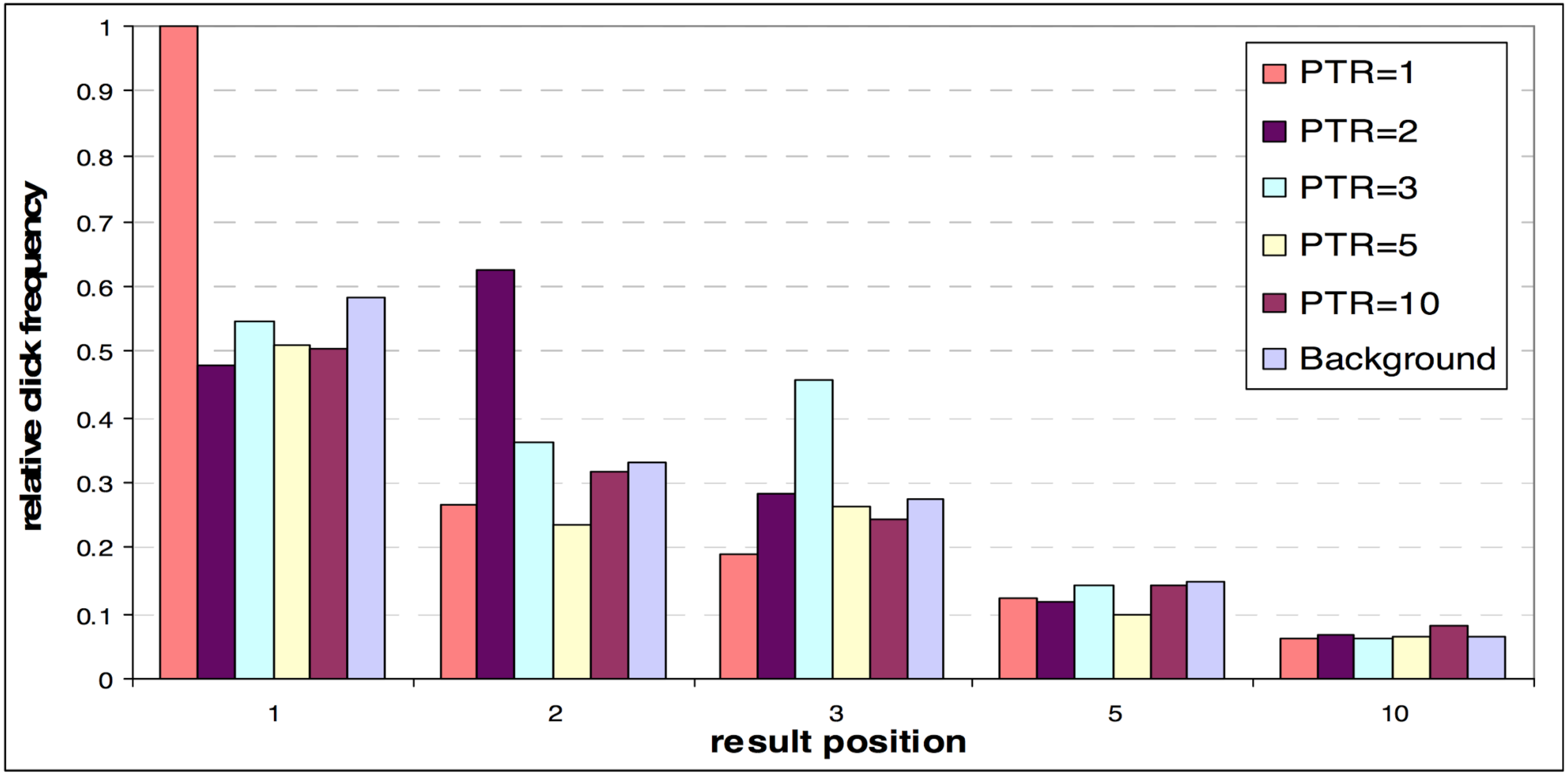


Figure 3.2: Relative click frequency for queries with varying PTR (Position of Top Relevant document).

Learning User Behavior Model

- Activity:
  - How do you interpret relevance result from previous distribution?

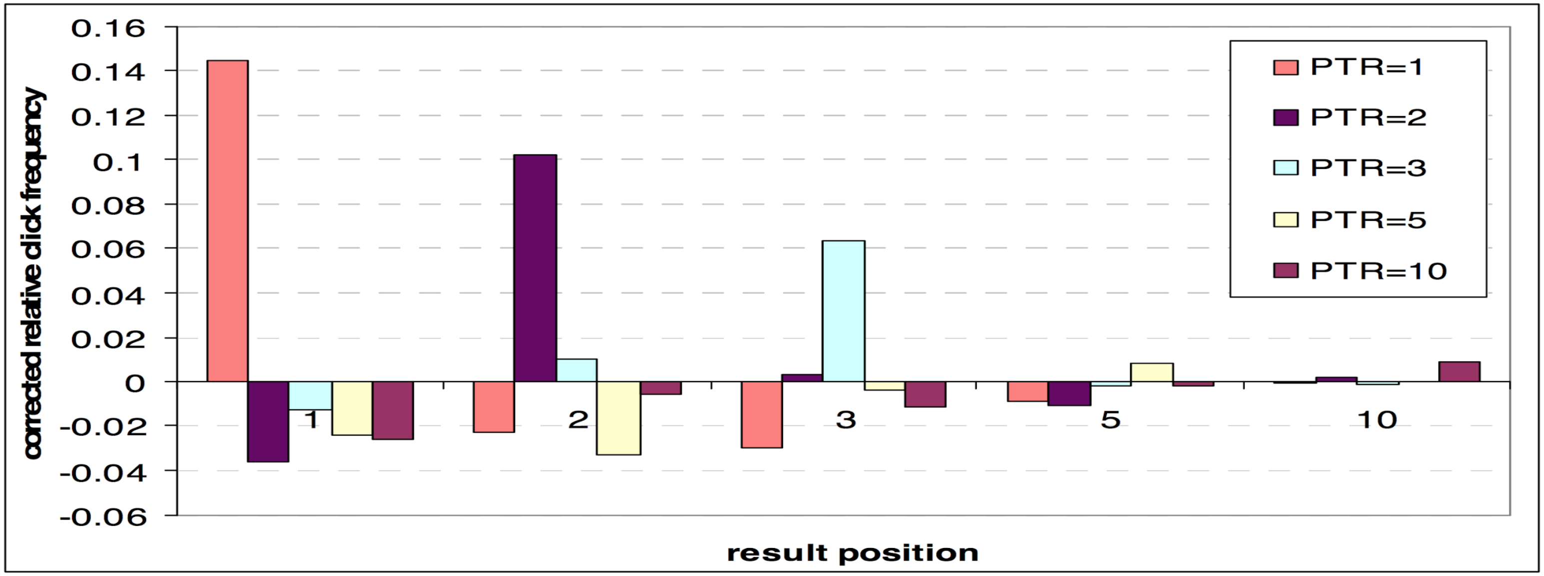


Figure 3.3: Relative corrected click frequency for relevant documents with varying PTR (Position of Top Relevant).

Learning User Behavior Model : Robust user behavior model

- Post-search activities are comprised of clicks, page dwell time, clicks from search, etc.
- We have just shown how the ‘relevance-driven’ click distribution can be recovered from the biased observed distribution.
- We conjecture that for other aspects of user behavior, we can do something similar. Observed value o of a feature f for query q and result r can be expressed as

$$o(q,r,f) = C(r,f) + rel(q,r,f)$$

- where $C(r,f)$ is the ‘background’ distribution

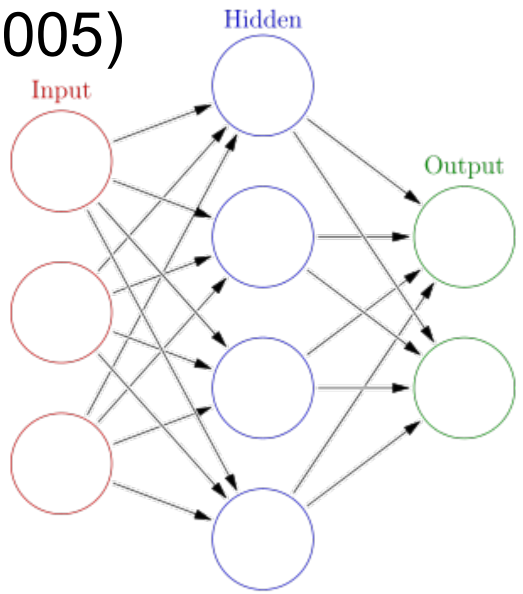
Learning User Behavior Model : Features representing user behavior

Query-text features	
TitleOverlap	Fraction of shared words between query and title
SummaryOverlap	Fraction of shared words between query and summary
QueryURLOverlap	Fraction of shared words between query and URL
QueryDomainOverlap	Fraction of shared words between query and domain
QueryLength	Number of tokens in query
QueryNextOverlap	Average fraction of words shared with next query
Browsing features	
TimeOnPage	Page dwell time
CumulativeTimeOnPage	Cumulative time for all subsequent pages after search
TimeOnDomain	Cumulative dwell time for this domain
TimeOnShortUrl	Cumulative time on URL prefix, dropping parameters
IsFollowedLink	1 if followed link to result, 0 otherwise
IsExactUrlMatch	0 if aggressive normalization used, 1 otherwise
IsRedirected	1 if initial URL same as final URL, 0 otherwise
IsPathFromSearch	1 if only followed links after query, 0 otherwise
ClicksFromSearch	Number of hops to reach page from query
AverageDwellTime	Average time on page for this query
DwellTimeDeviation	Deviation from overall average dwell time on page
CumulativeDeviation	Deviation from average cumulative time on page
DomainDeviation	Deviation from average time on domain
ShortURLDeviation	Deviation from average time on short URL
Clickthrough features	
Position	Position of the URL in Current ranking
ClickFrequency	Number of clicks for this query, URL pair
ClickRelativeFrequency	Relative frequency of a click for this query and URL
ClickDeviation	Deviation from expected click frequency
IsNextClicked	1 if there is a click on next position, 0 otherwise
IsPreviousClicked	1 if there is a click on previous position, 0 otherwise
IsClickAbove	1 if there is a click above, 0 otherwise
IsClickBelow	1 if there is click below, 0 otherwise

derived feature

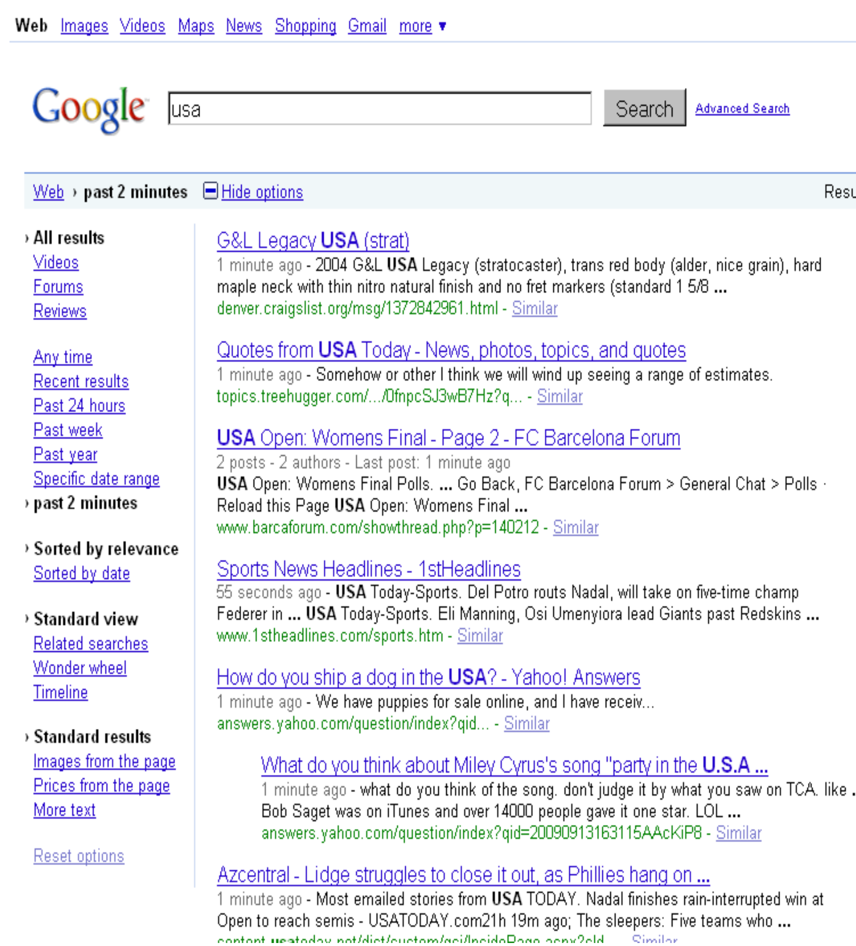
Learning User Behavior Model : Learning a predictive behavior model

- Instead of heuristics or insights, we use supervised learning to map features to user preferences.
    - Advantage: We can always mine more data instead of relying on intuition and limited lab evidence.
- Training data: query/URL pair, explicit label by expert.
- Training method : RankNet (Burges et al. 2005)
    - Scalable neural net training
    - Pairwise preference
    - Use gradient descent to rank



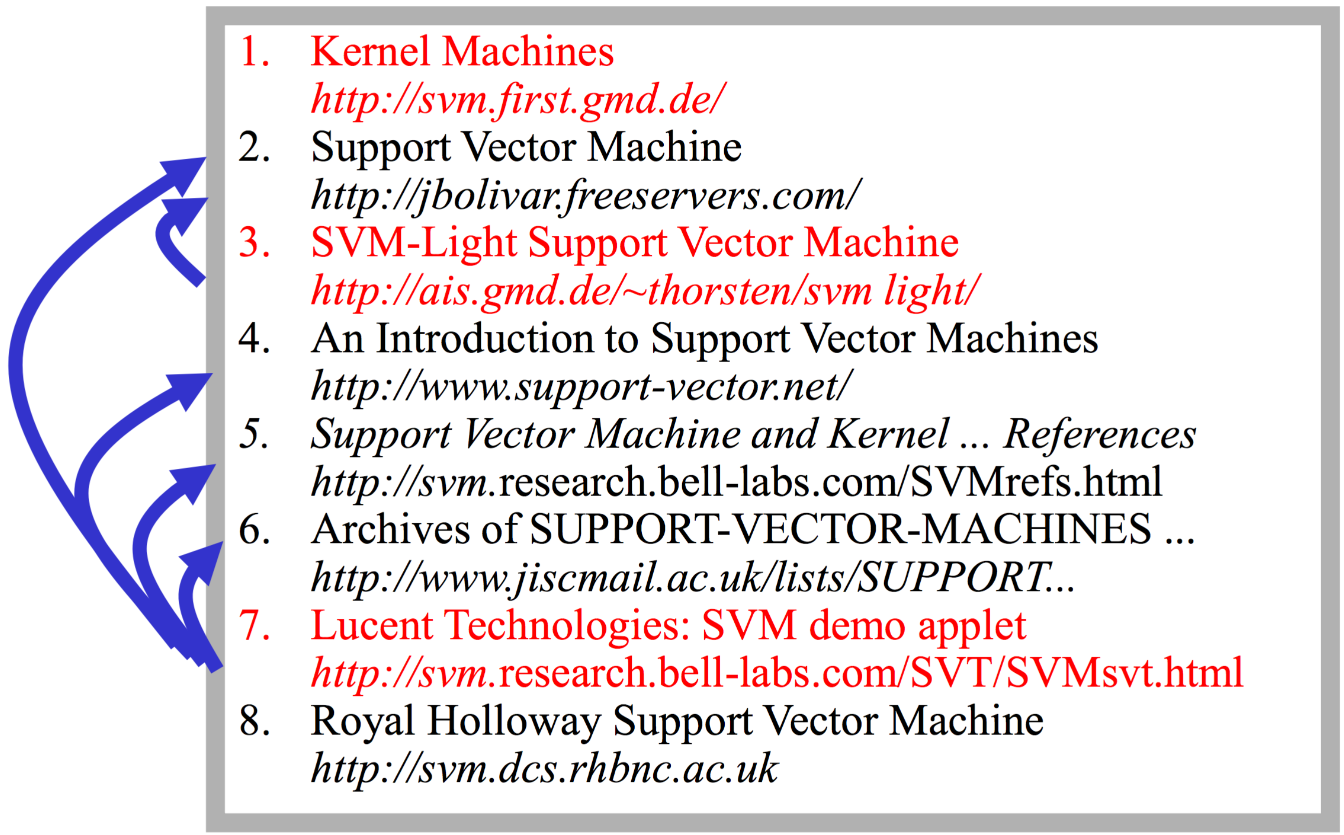
Predicting User Preferences : Baseline Model

- Baseline Model (“current”)
  - A state-of-the-art page ranking system currently used by a major web search engine.
  - The algorithm ranks results based on hundreds of features such as query to document similarity, query to anchor text similarity and intrinsic page quality.



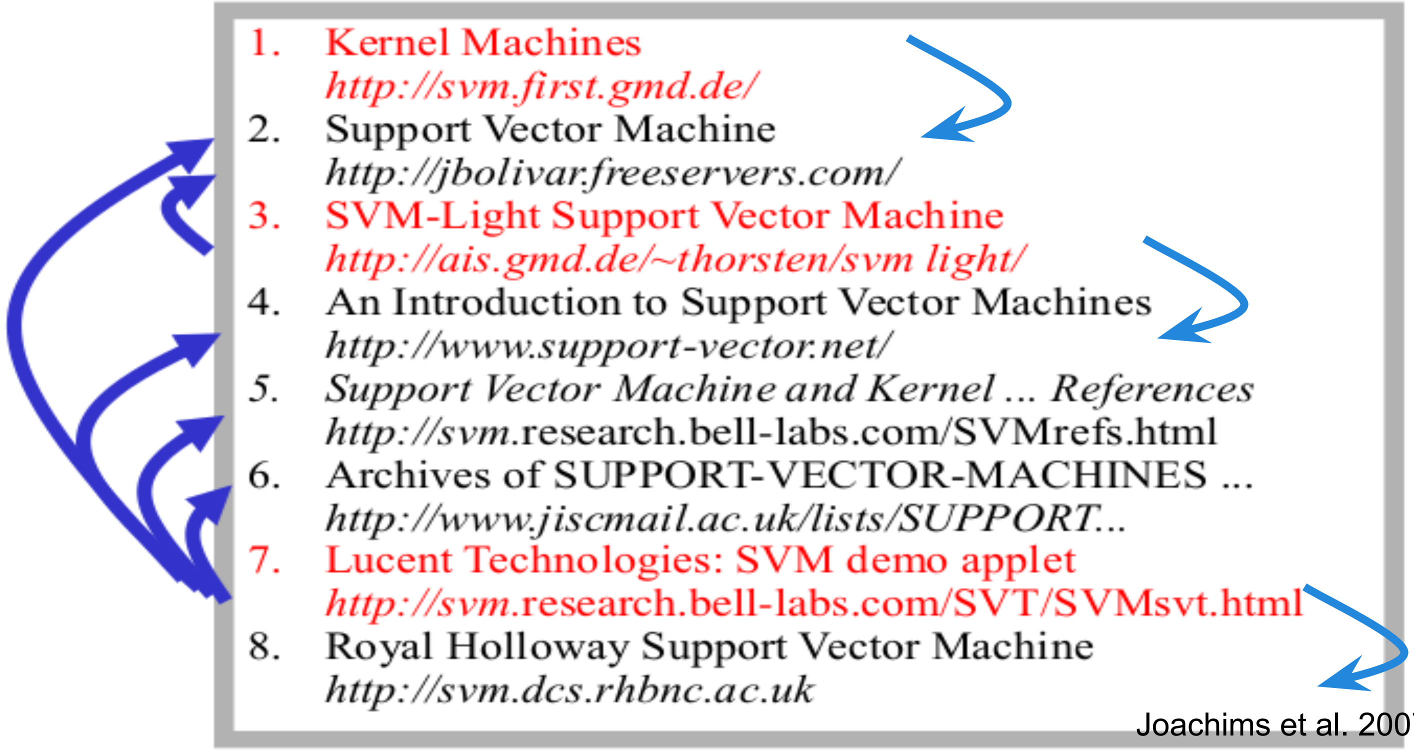
Predicting User Preferences : Clickthrough Model

- Clickthrough Model (Joachims et al. 2007)
  - Strategy SA (Skip Above):



Predicting User Preferences : Clickthrough Model

- Clickthrough Model (Joachims et al. 2007)
  - Strategy SA+N (Skip Above + Skip Next):



Joachims et al. 2007

Predicting User Preferences : Clickthrough Model

- Clickthrough Model with filtering
  - Strategy CD (deviation d): Given query, compute observed click frequency distribution $o(r,p)$

$$dev(r,p) = o(r,p) - C(p)$$

  - If $dev(r,p) > d$, retain the click as input to SA+N strategy

Predicting User Preferences : Clickthrough and General User Model

- Clickthrough Model with filtering
  - Strategy CDiff(margin m) : For each pair of results r_i, r_j predict preference of r_i over r_j iff
    - $dev(r_i, p_i) - dev(r_j, p_j) > m$
  - Strategy CD + CDiff (deviation d, margin m): CDiff and CD are complimentary. CDiff is a generalization of the clickthrough frequency model of CD, while ignoring the positional information used in CD.
- General User Behavior Model
  - User Behavior Strategy: Supervised learning model based on direct & derived features described in previous slide.

Experimental Setup: Methods Compared and Datasets

- Methods compared:

Current	SA	CD	UserBehavior
	SA+N	CDiff	
		CD+CDiff	

- 3500 queries randomly sampled
  - Top 10 results for each query manually rated by experts
  - Defined 3 subsets
    - **Q1 :** Queries with ***at least 1 click*** (3500 queries)
    - **Q10 :** Queries with ***at least 10 clicks*** (1300 queries)
    - **Q20 :** Queries with ***at least 20 clicks*** (1000 queries)

Experimental Setup : Evaluation Methodology and Metrics

- Evaluation based on pairwise agreement with explicit
- Query Precision(q) =

$$\frac{\#\{pref : pref \in prediction(q) \wedge pref \in explicit\}}{\#prediction(q)}$$

  - Fraction of pairs predicted that agree with human ratings
- Query Recall(q) =

$$\frac{\#\{pref : pref \in prediction(q) \wedge pref \in explicit\}}{\#explicit}$$

  - Fraction of human-rated preferences predicted correctly
- Average Query Precision/Recall for evaluation

Experimental Setup : More on Metrics

Deviation : $dev(r,p) > \mathbf{d}$

Margin : $dev(r_i,p_i) - dev(r_j,p_j) > \mathbf{m}$

d and ***m*** as tradeoff between Query Precision and Recall

Activity 2 :
What effect will changing **d** and **m** (both increase/decrease) have on query precision and query recall? Why?

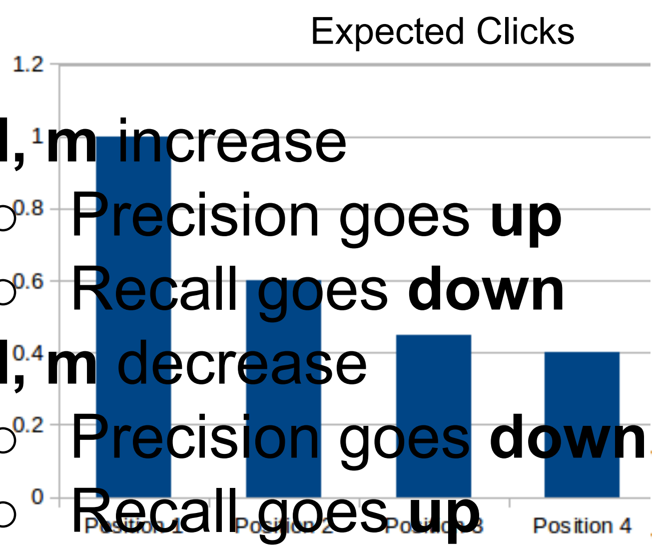
- Query Precision(q) = $\frac{\#\{pref : pref \in prediction(q) \wedge pref \in explicit\}}{\#prediction(q)}$
- Query Recall(q) = $\frac{\#\{pref : pref \in prediction(q) \wedge pref \in explicit\}}{\#explicit}$

Experimental Setup : More on Metrics

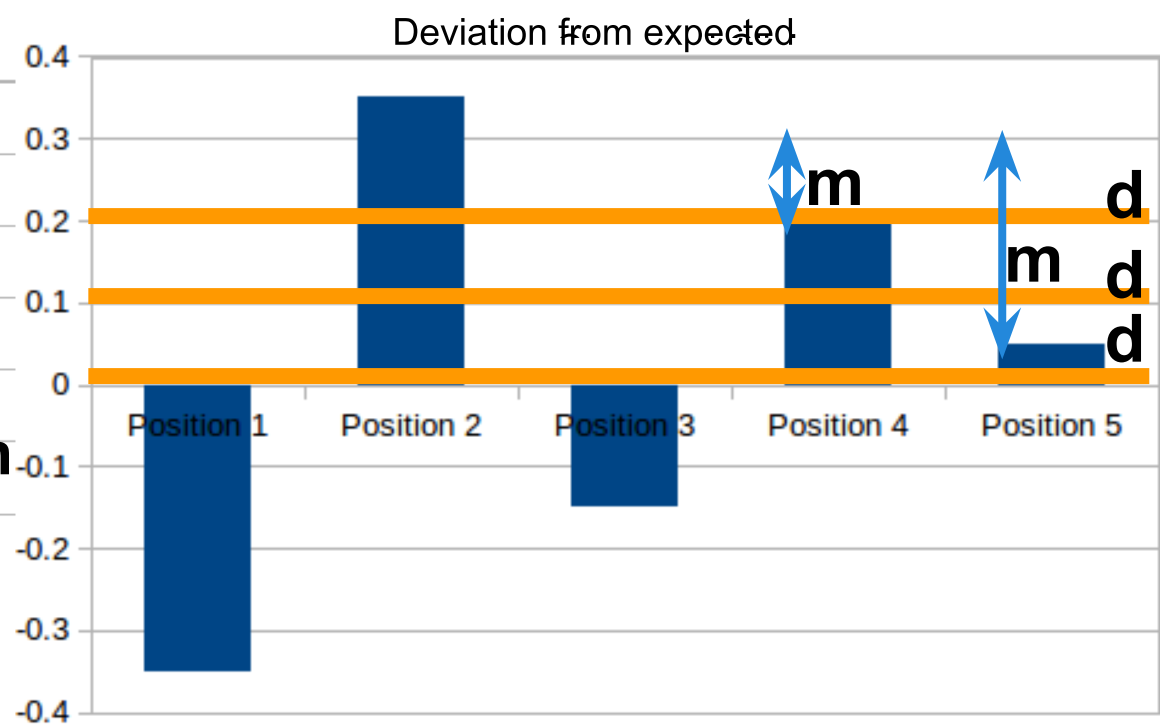
Deviation : $$dev(r, p) > \mathbf{d}$$

Margin : $$dev(r_i, p_i) - dev(r_j, p_j) > \mathbf{m}$$

d and ***m*** as tradeoff between Query Precision and Recall



- **d, m** increase
  - Precision goes **up**
  - Recall goes **down**
- **d, m** decrease
  - Precision goes **down**
  - Recall goes **up**



Experimental Setup: Results

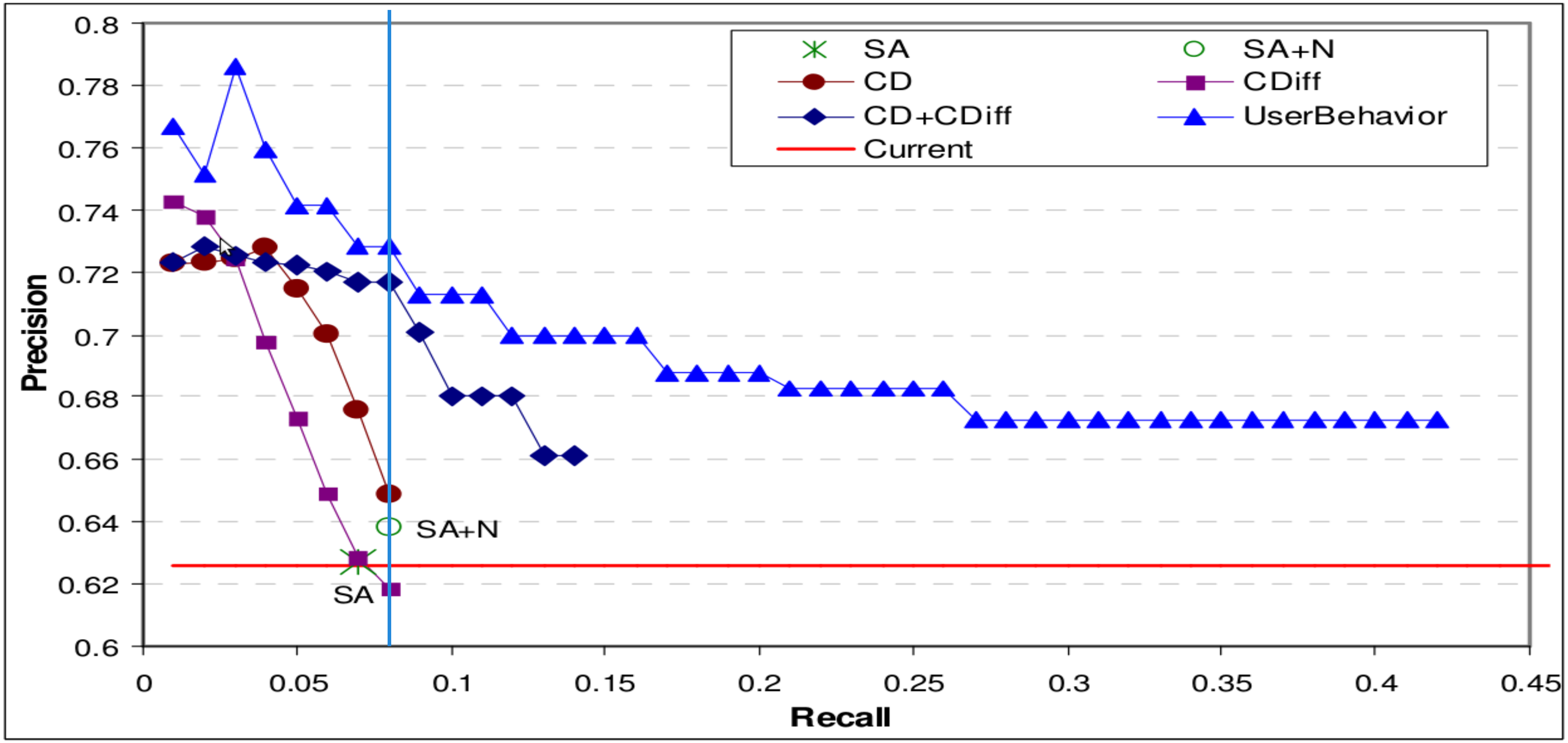


Figure 6.1: Precision vs. Recall of SA, SA+N, CD, CDiff, CD+CDiff, UserBehavior, and Current relevance prediction methods over the Q1 dataset.

Experimental Setup: Results

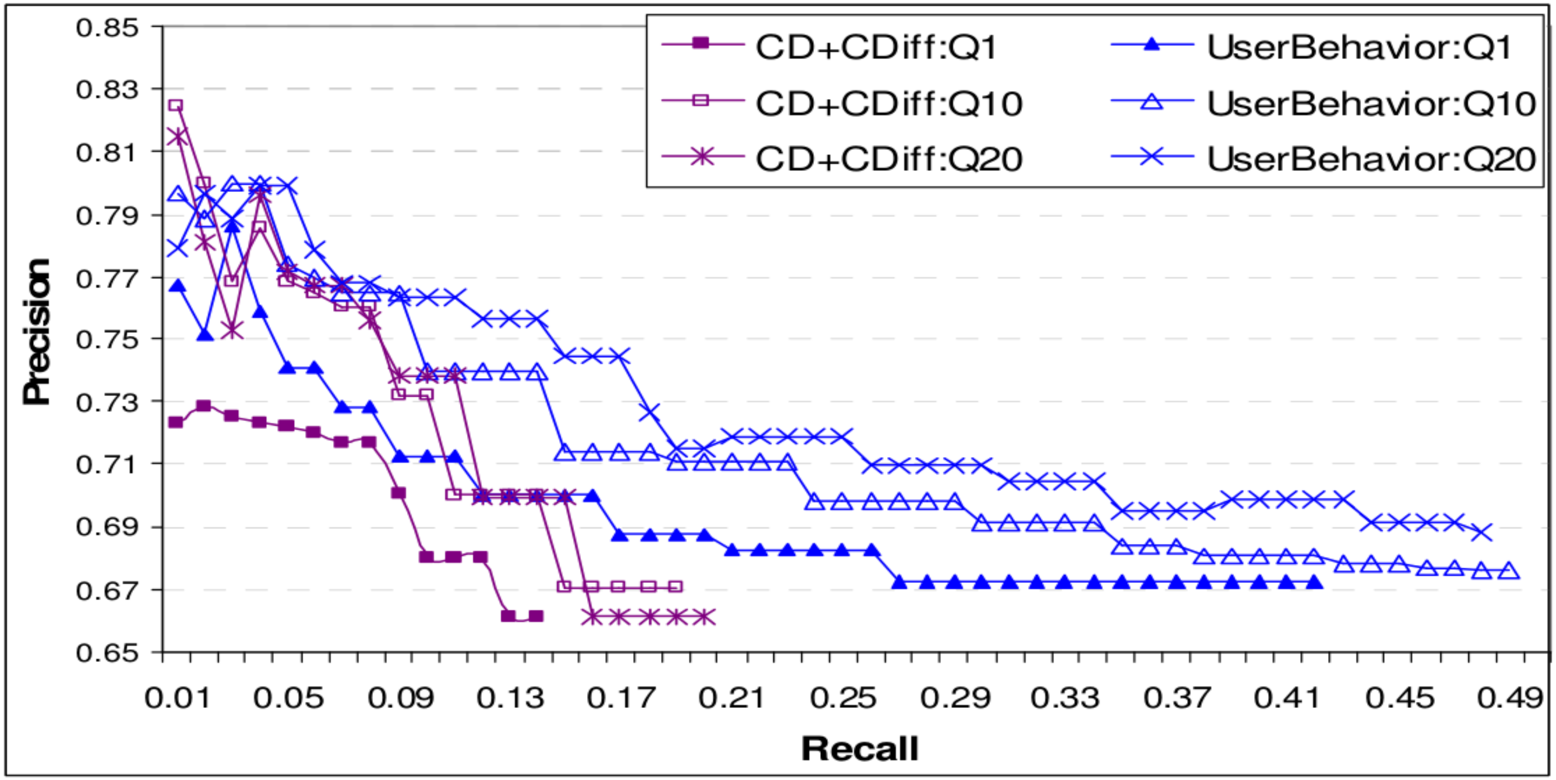


Figure 6.3: Recall vs. Precision of CD+CDiff and UserBehavior for query sets Q1, Q10, and Q20 (queries with at least 1, at least 10, and at least 20 clicks respectively).

Experimental Setup: Results

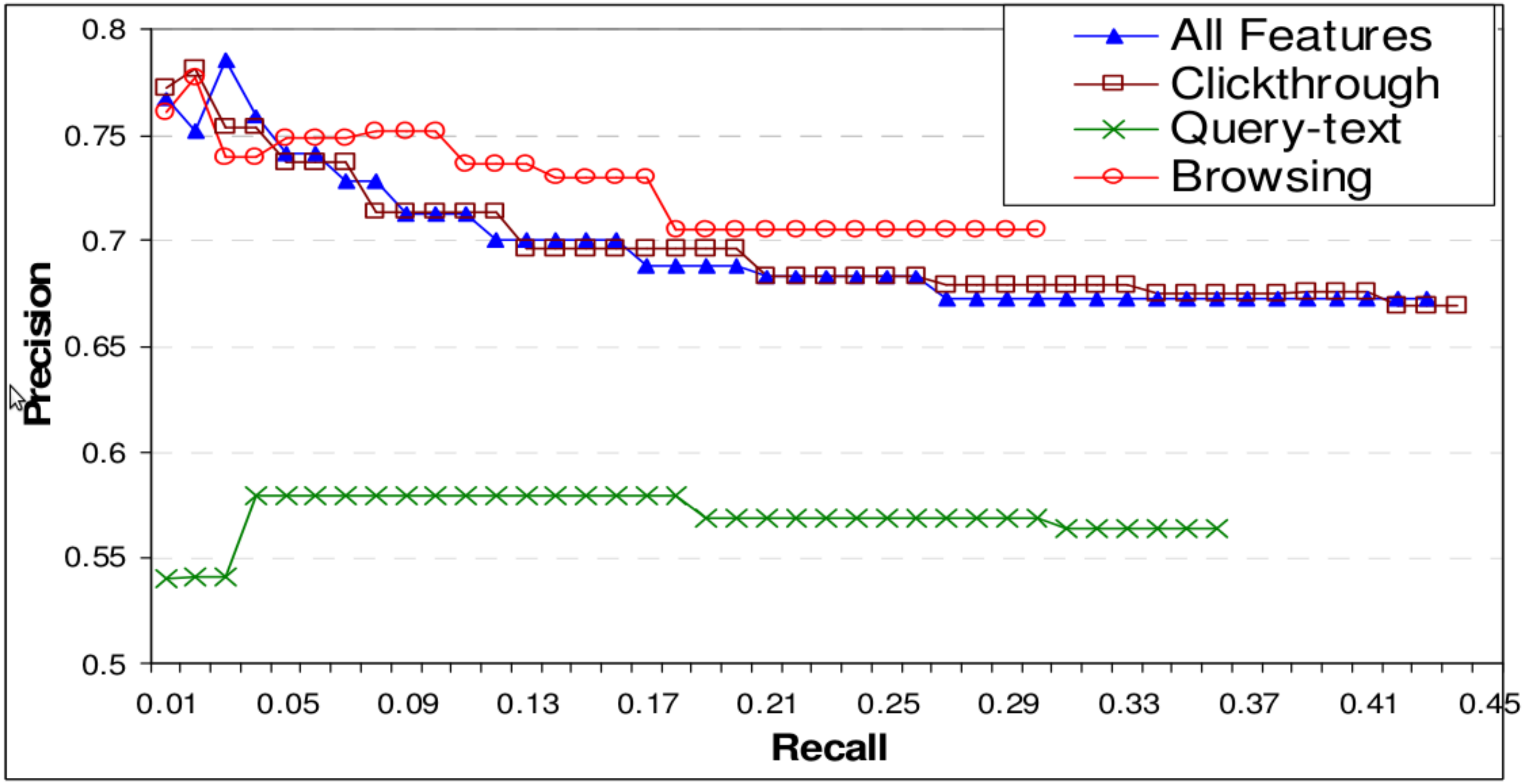


Figure 6.2: Precision vs. recall for predicting relevance with each group of features individually.

Conclusion

- Observed a wide range of strategies:
    - SA, SA+N
    - CD, CDiff
        - Considers “background noise”
    - UserBehavior
        - Richer features
- Accounting for the “background noise” before applying clickthrough strategies can improve accuracy.
- Using richer features that include user behavior before and after search lead to increased accuracy.