Optimizing Search Engines using *Clickthrough Data*

Presented by -

Kajal Miyan

Seminar Series, 891

Michigan state University

**Slides adopted from presentations of*
Thorsten Joachims (author) and
Shui-Lung Chuang,

The Goal Reiterated

"The meaning of life? Wait a minute. I'll try to find it on the internet."

H.L. Schwadron for *Barron's*

The Reference Papers

Evaluating Retrieval Performance using Clickthrough Data
Technical Report, Cornell U., 2002

Optimizing Search Engines using Clickthrough Data
KDD-2002

Thorsten Joachims
Department of Computer Science
Cornell University

Outline

- Things about clickthrough data
- Evaluating search engines using clickthrough data
- Optimizing search engines using clickthrough data
- SVM-light
- Open issues
- Conclusion
- Discussion

Inspiration

- Search Engines.
- Vs Learning Search Engines
- Recent use of SVM-light with astonishing results.

Heterogeneity and Homogeneity in Web Search

- Different users but similar behavior patterns.

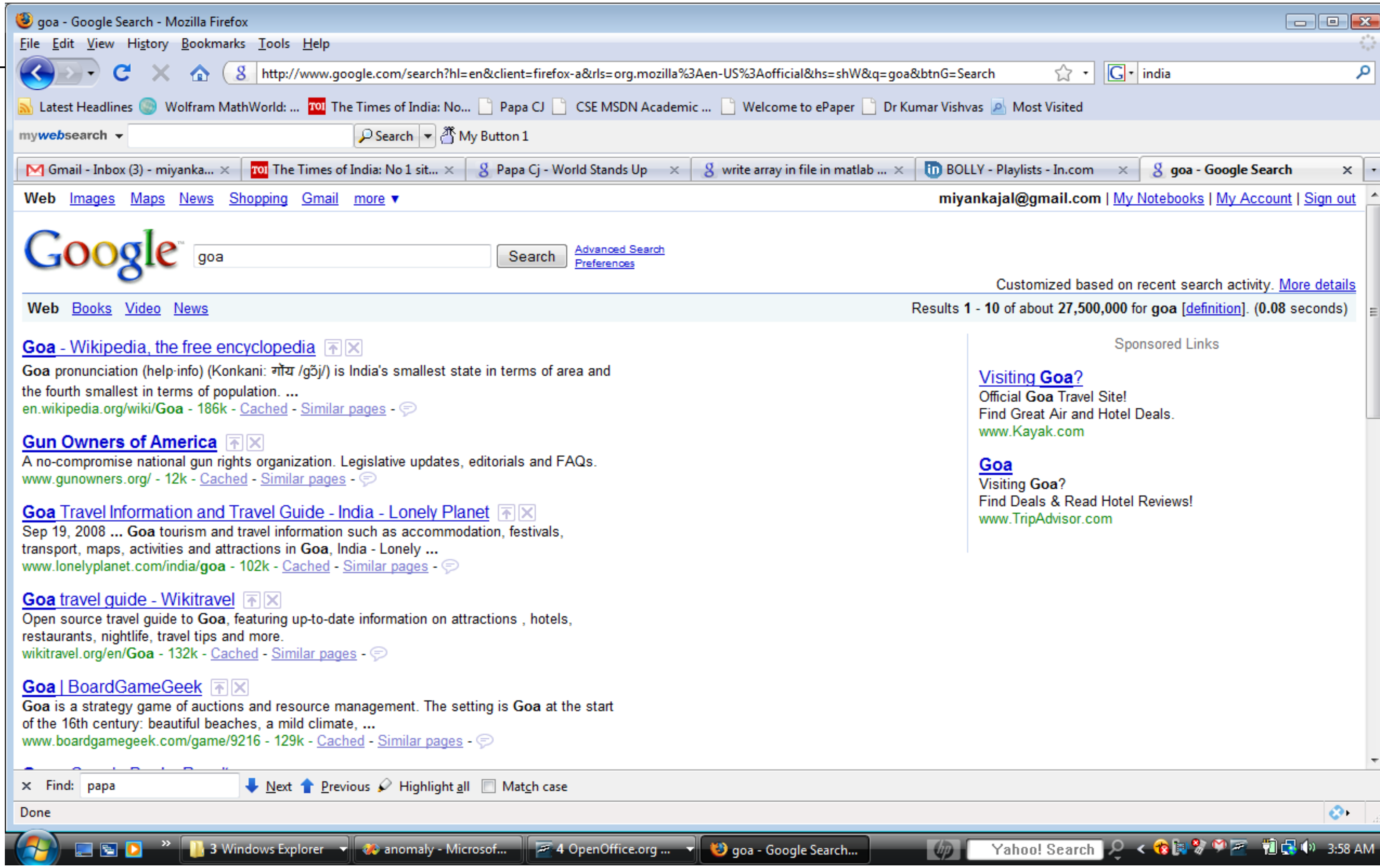
1 screens per query	85.2%	max screens per query	78496
2 screens per query	7.5%	2nd most screens	5108
3 screens per query	3.0%	stddev of screens/query	1.39
> 3 screens per query	4.3%	avg screens per query	3.74

Table 8: Statistics concerning the characteristics of result screen requests in sessions

Problem Definition

- Optimization of web search results ranking
  - Ranking algorithms mainly based on similarity
    - Similarity between query keywords and page text keywords
    - Similarity between pages (Page Rank)
  - No consideration for user personal preferences

» Goa – Tourist Destination in India… I want to visit



- Room for improvement by incorporating user behaviour data: **user feedback**
  - Use of previous implicit search information to enhance result ranking

Types of user feedback

- Explicit feedback
    - User explicitly judges relevance of results with the query
        - Costly in time and resources
        - Costly for user → limited effectiveness
        - Direct evaluation of results
- Implicit feedback
    - Extracted from log files
        - Large amount of information
        - Indirect evaluation of results through click behaviour

Implicit feedback (Categories)

- Clicked results
  - **Absolute relevance**:
    - Clicked result -> Relevant
      - Risky: poor user behaviour quality
    - Percentage of result clicks for a query
    - Frequently clicked groups of results
    - Links followed from result page
  - **Relative relevance**:
    - Clicked result -> More relevant than non-clicked
      - More reliable
- Time
  - Between clicks
    - E.g., fast clicks → maybe bad results
  - Spent on a result page
    - E.g., a lot of time → relevant page
  - First click, scroll
    - E.g., maybe confusing results

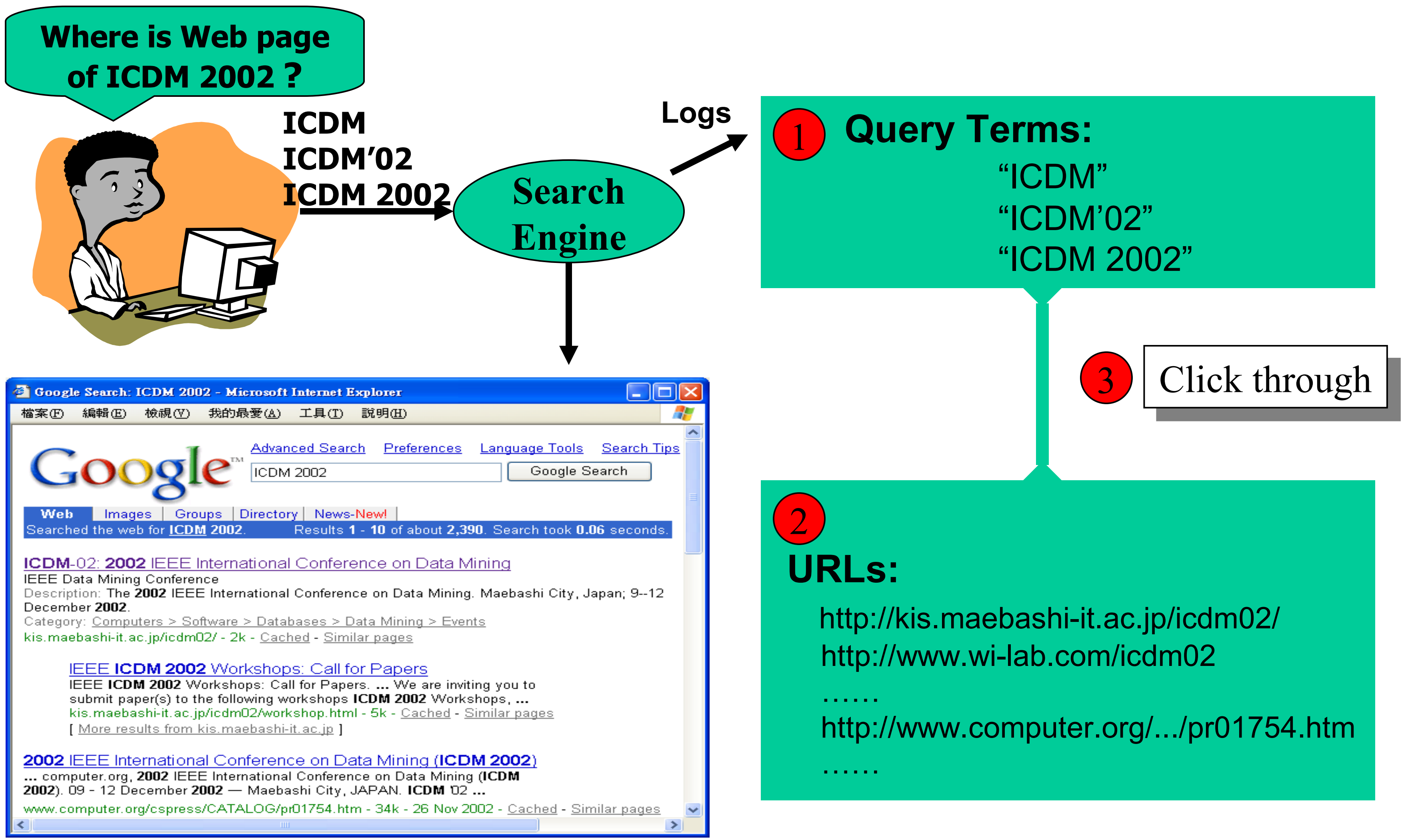
Implicit feedback (Categories)

- Query chains: sequences of reformulated queries to improve results of initial search
  - Detection:
    - Query similarity
    - Result sets similarity
    - Time
  - Connection between results of different queries:
    - Enhancement of a bad query with another from the query chain
    - Comparison of result relevance between different query results
- Scrolling
  - Time (quality of results)
  - Scrolling behaviour (quality of results)
  - Percentage of page (results viewed)
- Other features
  - Save, print, copy etc of result page → maybe relevant
  - Exit type (e.g. close window → poor results)

Joachims approach

- Clickthrough data
  - Relative relevance
  - Indicated by user behaviour study
- Method
  - Training of svm functions
  - Training input: inequations of query result rankings
  - Trained function: weight vector for features examined
  - Use of trained vector to give weights to examined features
- Experiments
  - Comparison of method with existing search engines

Search Engine Logs



Clickthrough Data

- Clickthrough data can be thought of as triplets **(q,r,c)**
  - the query **q**
  - the ranking **r** presented to the user
  - the set **c** of links the user clicked on
- E.g.,

q

support vector machine

r

1. **Kernel Machines**
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. **SVM-Light Support Vector Machine**
   http://ais.gmd.de/~thorsten/svm_light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel Methods References
   http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
   http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html
7. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
   http://www.support-vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
    http://www.cs.wisc.edu/dmi/lsvm

c

link1
link3
link7

➢ Clickthough data provide *users' feedback for relevance judgment*

A Mechanism to Record Clickthrough Data

Each query is assigned a unique ID which is stored in the query-log along with the query words and the presented ranking. The links on the results-page presented to the user do not lead directly to the suggested document, but point to a proxy server. These links encode the query-ID and the URL of the suggested document. When the user clicks on the link, the proxy-server records the URL and the query-ID in the click-log. The proxy then uses the *HTTP Location* command to forward the user to the target URL. This process can be made transparent to the user and does not influence system performance.

- query-log: the query words, the presented ranking
- click-log: query-ID, clicked URL (via a proxy server)

- This process should be made transparent to the user
- This process should not influence system performance

The Problem to start with

Which search engine provides better results:
Google or MSNSearch?

- A problem of statistical inference: hypothesis test
- Users are only rarely willing to give explicit feedback
- Clickthrough data **seem** to provide users' implicit feedback; Is them suitable for relevance judgment? I.e., **Click $\Rightarrow$ Relevance** ?

EXP1: Regular Clickthrough Data

Experiment Setup 1 (REGULAR CLICKTHROUGH DATA)
The user types a query into a unified interface and the query is sent to both search engines A and B. One of the returned rankings is selected at random and it is presented to the user. The ranks of the links the user clicked on are recorded.

Average clickrank

	retrieval function		
	bxx	tfc	hand-tuned
avg. clickrank	6.26±1.14	6.18±1.33	6.04± 0.92

- Clicks heavily depend on the ranking (presentation bias)

EXP2: Unbiased Clickthrough Data

- The criteria to get unbiased clickthrough data for comparing search engines
  - **Blind test**: The interface should hide the random variables underlying the hypothesis test to avoid biasing the user's response
  - **Click ⇒ preference**: The interface should be designed so that a click demonstrates a particular judgment of the user
  - **Low usability impact**: The interface should not substantially lower the productivity of the user

Experiment Setup 2 (Unbiased Clickthrough Data)

The user types a query into a unified interface. The query is sent to both search engines A and B. The returned rankings are mixed so that at any point the top l links of the combined ranking contain the top k_a and k_b links from rankings A and B, $|k_a - k_b| \leq 1$. The combined ranking is presented to the user and the ranks of the links the user clicked on are recorded.

EXP2: Unbiased Clickthrough Data

Google Results:

1. Kernel Machines
   http : //svm.first.gmd.de/
2. SVM-Light Support Vector Machine
   http : //ais.gmd.de/.../svm_light/
3. Support Vector Machine ... References
   http : //svm.....com/SVMrefs.html
4. Lucent Technologies: SVM demo applet
   http : //svm.....com/SVT/SVMsvt.html
5. Royal Holloway Support Vector Machine
   http : //svm.dcs.rhbnc.ac.uk/
6. Support Vector Machine - The Software
   http: //wwwsupport—vectornet/softwarehtml
7. Support Vector Machine - Tutorial
   http://wwwsupport—vectornet/tutorialhtml
8. Support Vector Machine
   http : //jbolivar.freeservers.com/

MSNSearch Results:

1. Kernel Machines
   http : //svm.first.gmd.de/
2. Support Vector Machine
   http : //jbolivar.freeservers.com/
3. An Introduction to Support Vector Machines
   http : //www.support — vector.net/
4. Archives of SUPPORT-VECTOR- ...
   http : //www.jiscmail.ac.uk/lists/...
5. SVM-Light Support Vector Machine
   http : //ais.gmd.de/.../svm_light/
6. Support Vector Machine - The Software
   http: //wwwsupport—vectornet/softwarehtml
7. Lagrangian Support Vector Machine Home Page
   http : //www.cs.wisc.edu/dmi/lsvm
8. A Support ... - Bennett, Blue (ResearchIndex)
   http : //citeseer.../bennett97support.html

Combined Results:

1. **Kernel Machines**
   http : //svm.first.gmd.de/
2. Support Vector Machine
   http : //jbolivar.freeservers.com/
3. **SVM-Light Support Vector Machine**
   http : //ais.gmd.de/ ~ thorsten/svm_light/
4. An Introduction to Support Vector Machines
   http : //www.support — vector.net/
5. Support Vector Machine and Kernel Methods References
   http : //svm.research.bell — labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
   http : //www.jiscmail.ac.uk/lists/SUPPORT—VECTOR—MACHINES.html
7. **Lucent Technologies: SVM demo applet**
   http : //svm.research.bell — labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http : //svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
   http : //www.support — vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
    http : //www.cs.wisc.edu/dmi/lsvm

- Top ***l*** links of the combined ranking containing the top ***ka*** and ***kb*** links from rankings A and B; |***ka-kb***|≤1.

Computing the Combined Ranking

Algorithm 1 (Combine Rankings)

Input: ranking $A = (a_1, a_2, \ldots)$, ranking $B = (b_1, b_2, \ldots)$
Call: combine$(A,B,0,0,\emptyset)$
Output: combined ranking D

```
combine(A,B,k_a,k_b,D) {
        if(k_a = k_b) {
                if(A[k_a + 1] ∉ D) { D := D + A[k_a + 1]; }
                combine(A,B,k_a + 1,k_b,D);
        }
        else {
                if(B[k_b + 1] ∉ D) { D := D + B[k_b + 1]; }
                combine(A,B,k_a,k_b + 1,D);
        }
}
```

Theorem 1 *Algorithm 1 always produces a combined ranking $D = (d_1, d_2, \ldots)$ from $A = (a_1, a_2, \ldots)$ and $B = (b_1, b_2, \ldots)$ so that for all n*

$$\{d_1, \ldots, d_n\} = \{a_1, \ldots, a_{k_a}\} \cup \{b_1, \ldots, b_{k_b}\} \tag{1}$$

with $k_b \leq k_a \leq k_b + 1$.

Experiment

- Google v.s. MSNSearch
- Experiment data gathered from 3 users, 9/25–10/18, 2001
  - 180 queries and 211 clicks (1.17 clicks/query, 2.31 words/query)
- The top *k* links for each query are manually judged for the relevance
- Questions to examine:
  - Does the clickthrough evaluation agree with the manual relevance judgments?
  - Click ⇒ Preference?

Theoretical Analysis: Assumption 1

Assumption 1 *Given a ranking in which the user encounters r relevant links and n non-relevant links before he stops browsing. Denote with c the number of links the user clicks on, whereas c_r of these links are relevant and c_n are non-relevant. Further denote with r_a and r_b the number of relevant links in the top k of rankings A and B respectively. It holds that*

$$\mathcal{E}\left(\frac{C_r}{R\,C}|r_a - r_b\right) - \mathcal{E}\left(\frac{C_n}{N\,C}|r_a - r_b\right) = \epsilon > 0 \tag{6}$$

for some $\epsilon > 0$ and all differences between r_a and r_b with non-zero probability. $\mathcal{E}(\cdot)$ denotes the expectation.

- Intuitively, this assumption formalizes that users click on a relevant link more frequently than on a non-relevant link by a difference of ε.

Theoretical Analysis: Assumption 2

Assumption 2

$$\begin{aligned}
\mathcal{E}(C_{a,r}|c_r, c_n, r_a, n_a, r_b, n_b, r, n) &= c_r \frac{r_a}{r} && (7)\\
\mathcal{E}(C_{a,n}|c_r, c_n, r_a, n_a, r_b, n_b, r, n) &= c_n \frac{n_a}{n} && (8)\\
\mathcal{E}(C_{b,r}|c_r, c_n, r_a, n_a, r_b, n_b, r, n) &= c_r \frac{r_b}{r} && (9)\\
\mathcal{E}(C_{b,n}|c_r, c_n, r_a, n_a, r_b, n_b, r, n) &= c_n \frac{n_b}{n} && (10)
\end{aligned}$$

- Intuitively, the assumption states that the only reason for a user clicking on a particular link is due to the relevance of the link, but not to other influence factors connected with a particular retrieval function.

Statistical Hypothesis Test

- Two-tailed paired t-test
- binomial sign test

Please refer to the paper if you have interest

Clickthrough vs. Relevance

Comparison using pairwise clickthrough data.

A	B	$c_a > c_b$ (A better)	$c_a < c_b$ (B better)	$c_a = c_b > 0$ (tie)	$c_a = c_b = 0$	total
Google	MSNSearch	34	20	46	23	123
Google	Default	18	1	3	12	34
MSNSearch	Default	17	2	1	4	24

Google vs. MSNSearch	(77% vs. 63%)
Google vs. Default	(85% vs. 18%)
MSNSearch vs. Default	(91% vs. 12%)

Comparison using manual relevance judgments.

A	B	$r_a > r_b$ (A better)	$r_a < r_b$ (B better)	$r_a = r_b > 0$ (tie)	$r_a = r_b = 0$	total
Google	MSNSearch	26	17	51	29	123
Google	Default	19	1	1	13	34
MSNSearch	Default	15	1	0	8	24

Is Assumption 1 Valid?

- Assumption 1: User clicks on more relevant links than non-relevant links on average

Let I_d be the set of queries with $r_a - r_b = d$ and $d \neq 0$.

$$\hat{\epsilon}_d = \frac{1}{I_d} \sum_{I_d} \frac{c_r}{c\,r} - \frac{1}{I_d} \sum_{I_d} \frac{c_n}{c\,n}$$

		$R_a - R_b$	
A	B	-1	+1
Google	MSNSearch	0.73 ± 0.11	0.71 ± 0.09
Google	Default	—	0.76 ± 0.08
MSNSearch	Default	—	0.85 ± 0.07

Is Assumption 2 Valid?

$$
\begin{aligned}
\mathcal{E}(C_r \frac{R_a}{R}) &= \mathcal{E}(C_{r,a}) \\
\mathcal{E}(C_r \frac{R_b}{R}) &= \mathcal{E}(C_{r,b}) \\
\mathcal{E}(C_n \frac{N_a}{N}) &= \mathcal{E}(C_{n,a}) \\
\mathcal{E}(C_n \frac{N_b}{N}) &= \mathcal{E}(C_{n,b})
\end{aligned}
$$

		C_{ra}			C_{rb}			C_{na}			C_{nb}		
A	B	exp		obs	exp		obs	exp		obs	exp		obs
Google	MSNSearch	75.9	$\approx$	78	67.8	$\approx$	67	23.0	$\approx$	26	22.8	$\approx$	22
Google	Default	21.0	$\approx$	21	3.0	$\approx$	3	6.7	$\approx$	10	8.9	$\approx$	8
MSNSearch	Default	15.0	$\approx$	15	1.0	$\approx$	1	5.3	$\approx$	9	5.4	$\approx$	3

An Illustrative Scenario

support vector machine

1. **Kernel Machines**
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. **SVM-Light Support Vector Machine**
   http://ais.gmd.de/~thorsten/svm_light/
4. An Introduction to Support Vector Machines
   http://www.support-vector.net/
5. Support Vector Machine and Kernel Methods References
   http://svm.research.bell-labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
   http://www.jiscmail.ac.uk/lists/SUPPORT-VECTOR-MACHINES.html
7. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell-labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
   http://www.support-vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
    http://www.cs.wisc.edu/dmi/lsvm

Click ≠ Absolute Relevance Judgment

Clickthrough data as a triplet **(q,r,c)**

- The presented ranking **r** depends on the query **q** as determined by the retrieval function of search engine
- The set **c** of clicked-on links depends on both query **q** and the presented ranking **r**
  - E.g., Highly ranked links have advantages to be clicked

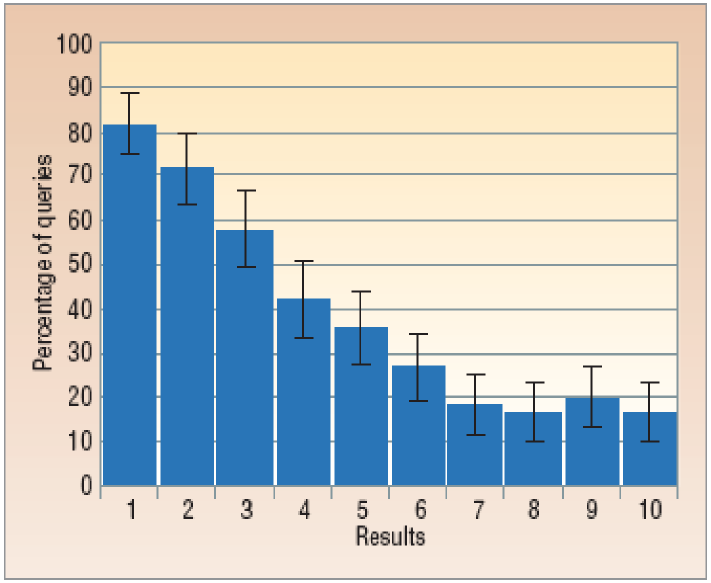
- A click on a particular link cannot be seen as an absolute relevance judgment

Average clickrank for three retrieval functions

	retrieval function		
	bxx	tfc	hand-tuned
avg. clickrank	6.26 ± 1.14	6.18 ± 1.33	6.04 ± 0.92

Clickthrough data (Studies)

- Why not absolute relevance?
  - User behaviour influenced by initial ranking
  - Study: Rank and viewership
    - Percentage of queries where a user viewed the search result presented at a particular rank
  - Conclusion: Most times users view only the few first results



Ranking and NOT Classification

- Classification into Relevant or non-Relevant results
- Neither possible nor feasible
- Ranking should be used.
- SVM-light

Click = Relative Preference Judgment

- Assuming that the user scanned the ranking from top to bottom
  - E.g., c = {link1,link3,link7} (r*: the ranking preferred by the user)

link3 <r* link2

link7 <r* link2
link7 <r* link4
link7 <r* link5
link7 <r* link6

ALGORITHM 1. (EXTRACTING PREFERENCE FEEDBACK FROM CLICKTHROUGH)
For a ranking $(link_1, link_2, link_3, \ldots)$ *and a set* C *containing the ranks of the clicked-on links, extract a preference example*

$$link_i <_{r^*} link_j$$

for all pairs $1 \leq j < i$, *with* $i \in C$ *and* $j \notin C$.

A Framework for Learning Retrieval Fun.

- r^* is optimal ordering, $r_{f(q)}$ is the ordering of retrieval function f on query q; r^* and $r_{f(q)}$ are binary relations over D×D, where D={d1,…,dm} is the document collection
  - e.g., di <r dj, then (di,dj)∈ r
- Kendall's τ (vs. average precision)

$$\tau(\mathrm{r}_a, \mathrm{r}_b) = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{m}{2}}$$

P: # of concordant pairs
Q: # of discordant pairs

- For a fixed but unknown distribution Pr(q,r*), the goal is to learn a retrieval function with the maximum expected Kendall's τ

$$\tau_P(\mathrm{f}) = \int \tau(\mathrm{r}_{\mathrm{f(q)}}, \mathrm{r}^*)\, d\Pr(\mathrm{q}, \mathrm{r}^*)$$

An SVM Algo. for Learning Ranking Fun.

- Given an independently and identically distributed training sample *S* of size *n* containing queries q with their target ranking r*

$$(\mathrm{q}_1, \mathrm{r}_1^*), (\mathrm{q}_2, \mathrm{r}_2^*), ..., (\mathrm{q}_n, \mathrm{r}_n^*).$$

- The learner will select a ranking function f from a family of ranking functions *F* that maximize the empirical τ

$$\tau_S(\mathrm{f}) = \frac{1}{n} \sum_{i=1}^{n} \tau(\mathrm{r}_{\mathrm{f}(\mathrm{q}_i)}, \mathrm{r}_i^*).$$

The Ranking SVM Algorithm

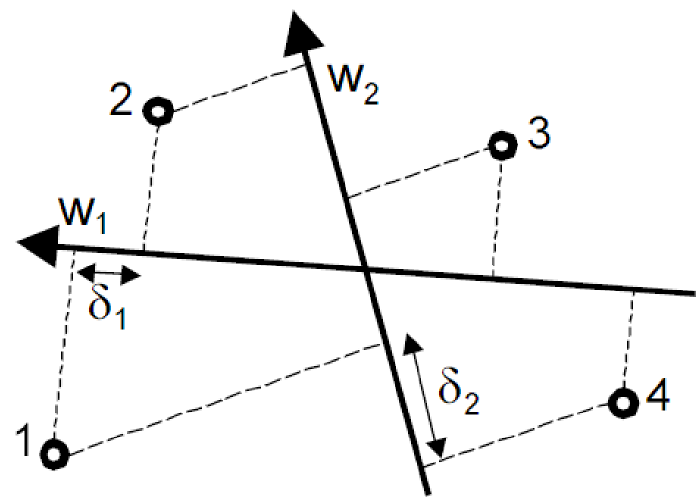
- Consider the class of linear ranking functions

$$(\mathrm{d}_i, \mathrm{d}_j) \in \mathrm{f}_{\vec{w}}(\mathrm{q}) \Longleftrightarrow \vec{w}\Phi(\mathrm{q}, \mathrm{d}_i) > \vec{w}\Phi(\mathrm{q}, \mathrm{d}_j).$$

  w is a weight vector that is adjusted by learning,
  Φ(q,d) is a mapping onto features describing the match between q and d

- The goal is to find a ***w*** so that max number of following inequalities is fulfilled

$$\forall (\mathrm{d}_i, \mathrm{d}_j) \in \mathrm{r}_1^* : \quad \vec{w}\Phi(\mathrm{q}_1, \mathrm{d}_i) > \vec{w}\Phi(\mathrm{q}_1, \mathrm{d}_j)$$
$$\cdots$$
$$\forall (\mathrm{d}_i, \mathrm{d}_j) \in \mathrm{r}_n^* : \quad \vec{w}\Phi(\mathrm{q}_n, \mathrm{d}_i) > \vec{w}\Phi(\mathrm{q}_n, \mathrm{d}_j)$$



The Categorization SVM

Learning a hypothesis h such that

$$P(error(h)) \leq train\_error(h) + complexity(h)$$

Example:

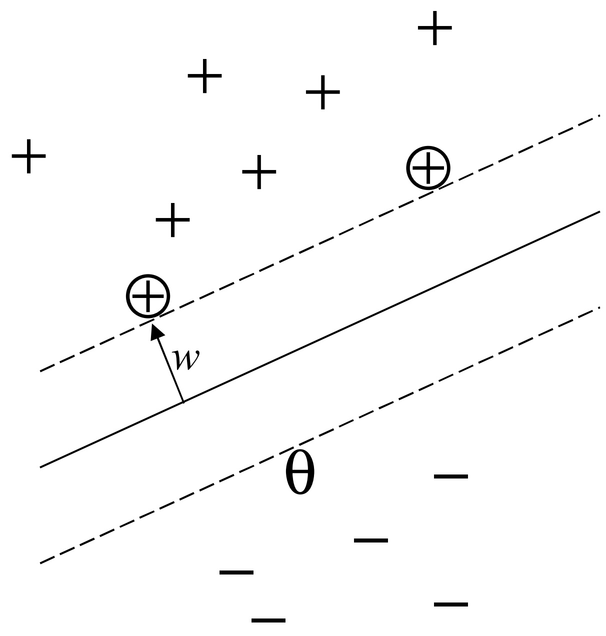
$$h(\vec{d}) = sign\{\vec{w} \bullet \vec{d} + b\} = \begin{cases} +1 & if\ \vec{w} \bullet \vec{d} + b > 0 \\ -1 & else \end{cases}$$

Learning ⇨ Optimization problem

Minimize: $\frac{1}{2} \vec{w} \bullet \vec{w}$

so that: $\forall_i : y_i[\vec{w} \bullet \vec{d}_i + b] \geq 1$

where $y_i = +1\ (-1)$ if d_i is in class + (–)



The Ranking SVM Algorithm *(cont.)*

OPTIMIZATION PROBLEM 1. (RANKING SVM)

$$\textit{minimize:} \quad V(\vec{w}, \vec{\xi}) = \frac{1}{2}\, \vec{w} \cdot \vec{w} + C \sum \xi_{i,j,k} \tag{12}$$

subject to:

$$\begin{aligned}
&\forall (d_i, d_j) \in r_1^* : \vec{w}\Phi(q_1, d_i) \geq \vec{w}\Phi(q_1, d_j) + 1 - \xi_{i,j,1} \\
&\qquad\qquad \ldots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (13) \\
&\forall (d_i, d_j) \in r_n^* : \vec{w}\Phi(q_n, d_i) \geq \vec{w}\Phi(q_n, d_j) + 1 - \xi_{i,j,n}
\end{aligned}$$

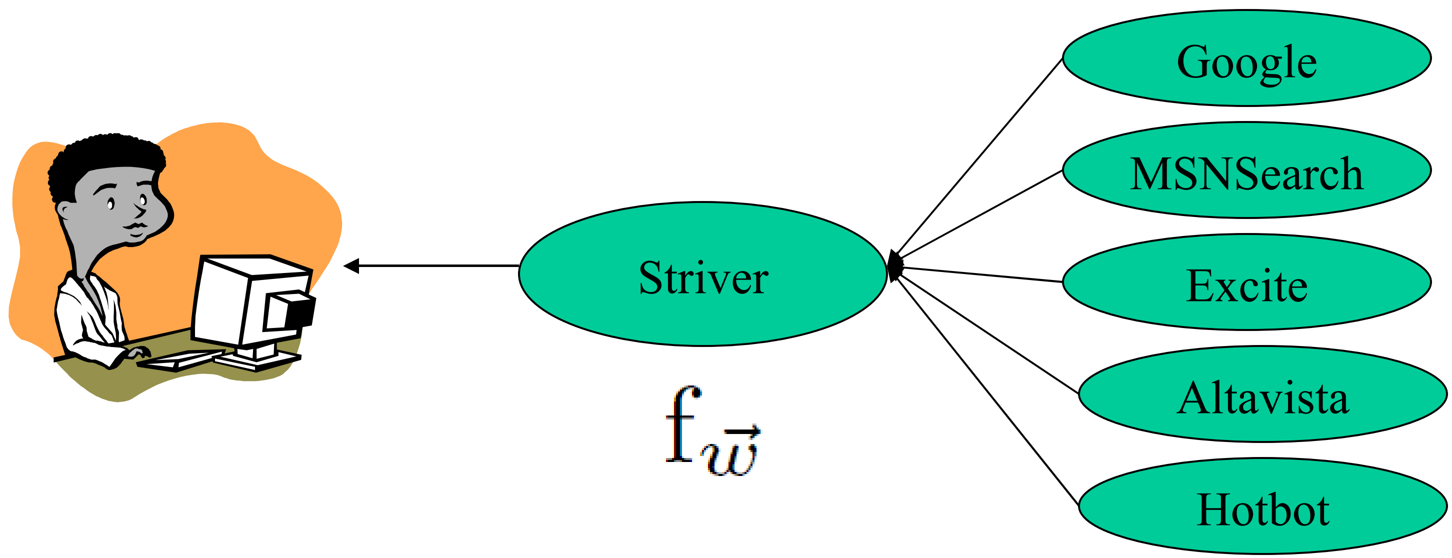
$$\forall i \forall j \forall k : \xi_{i,j,k} \geq 0 \tag{14}$$

- Optimization Problem 1 is convex and has no local optima. By rearranging the constraints as

$$\vec{w}\,(\Phi(\mathrm{q}_k, \mathrm{d}_i) - \Phi(\mathrm{q}_k, \mathrm{d}_j)) \geq 1 - \xi_{i,j,k},$$

  - A classification SVM on vectors $\Phi(\mathrm{q}_k, \mathrm{d}_i) - \Phi(\mathrm{q}_k, \mathrm{d}_j)$

Experiment Setup: Meta Search



Features

1. Rank in other search engines (38 features total):

   rank_X: 100 minus rank in $X \in$ {Google, MSN-Search, Altavista, Hotbot, Excite} divided by 100 (minimum 0)

   top1_X: ranked #1 in $X \in$ {Google, MSNSearch, Altavista, Hotbot, Excite} (binary $\{0,1\}$)

   top10_X: ranked in top 10 in $X \in$ {Google, MSN-Search, Altavista, Hotbot, Excite} (binary $\{0,1\}$)

   top50_X: ranked in top 50 in $X \in$ {Google, MSN-Search, Altavista, Hotbot, Excite} (binary $\{0,1\}$)

   top1count_X: ranked #1 in X of the 5 search engines

   top10count_X: ranked in top 10 in X of the 5 search engines

   top50count_X: ranked in top 50 in X of the 5 search engines

2. Query/Content Match (3 features total):

   query_url_cosine: cosine between URL-words and query (range $[0,1]$)

   query_abstract_cosine: cosine between title-words and query (range $[0,1]$)

   domain_name_in_query: query contains domain-name from URL (binary $\{0,1\}$)

3. Popularity-Attributes ($\sim$ 20.000 features total):

   url_length: length of URL in characters divided by 30

   country_X: country code X of URL (binary attribute $\{0,1\}$ for each country code)

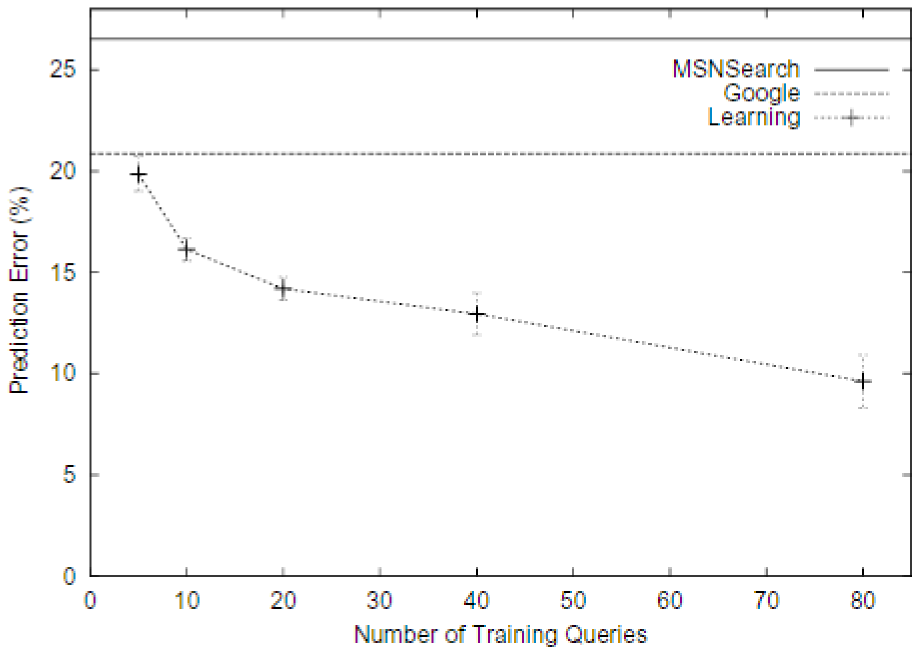
   domain_X: domain X of URL (binary attribute $\{0,1\}$ for each domain name)

   abstract_contains_home: word "home" appears in URL or title (binary attribute $\{0,1\}$)

   url_contains_tilde: URL contains "$\sim$" (binary attribute $\{0,1\}$)

   url_X: URL X as an atom (binary attribute $\{0,1\}$)

Experiment Results



Comparison	more clicks on learned	less clicks on learned	tie (with clicks)	no clicks	total
Learned vs. Google	29	13	27	19	88
Learned vs. MSNSearch	18	4	7	11	40
Learned vs. Toprank	21	9	11	11	52

Learned Weights of Features

weight	feature
0.60	query_abstract_cosine
0.48	top10_google
0.24	query_url_cosine
0.24	top1count_1
0.24	top10_msnsearch
0.22	host_citeseer
0.21	domain_nec
0.19	top10count_3
0.17	top1_google
0.17	country_de
...	
0.16	abstract_contains_home
0.16	top1_hotbot
...	
0.14	domain_name_in_query
...	
-0.13	domain_tu-bs
-0.15	country_fi
-0.16	top50count_4
-0.17	url_length
-0.32	top10count_0
-0.38	top1count_0

Open issues

- Trade-off between amount of training data and homogeneity
- Clustering algorithms to find homogenous groups of users
- Adaptation to the properties of a particular document collection. Shipping off-the-shelf SE that learns after deployment
- Incremental online learning/feedback algorithm
- Protection from spamming???
- Personalizing my search choices!!!

Possible extensions

- Utilization of time feedback
  - How long was the user browsing a clicked page
- Other types of feedback
  - Scrolling
  - Exit type
- Combination with absolute relevance clickthrough feedback
  - Percentage of result clicks for a query
  - Links followed from result page
- Query chains
  - Improvement of detection method
- Link association rules
  - For frequently clicked groups of results
- Query/links clustering
- Constant training of ranking functions

Conclusions

- The first work (evaluating search engines) is crucial
  - The feasibility of using clickthrough data in evaluating retrieval performance has been verified
  - The clickthrough data (less effort) perform as well as manual relevance judgment (more effort) in this task.
- The second (SVM) shows an interesting work on clickthrough data
- Negative comments
  - The approaches have not been justified in a larger scale, so whether the techniques are workable in real cases is still uncertain.
  - That perhaps is the reason that though the paper has been out since 2002, Google is still in business...

Discussion

- Is MILN similarly capable???