LEARNING TO RANK

Claudio Lucchese
claudio.lucchese@isti.cnr.it

Learning to Rank approaches

- ***Pointwise***
  - Each query-document pair is associated with a score
  - The objective is to predict such score
    - can be considered a *regression problem*
  - Does not consider the position of a document into the result list
- ***Pairwise***
  - We are given pairwise preferences, d_1 is better than d_2 for query q
  - The objective is to predict a score that preserves such preferences
    - Can be considered a *classification problem*
  - It partially considers the position of a document into the result list
- ***Listwise***
  - We are given the ideal ranking of results for each query
    - NB. It might not be trivial to produce such training set
  - Objective maximize the quality of the resulting ranked list
    - We need some improved approach…

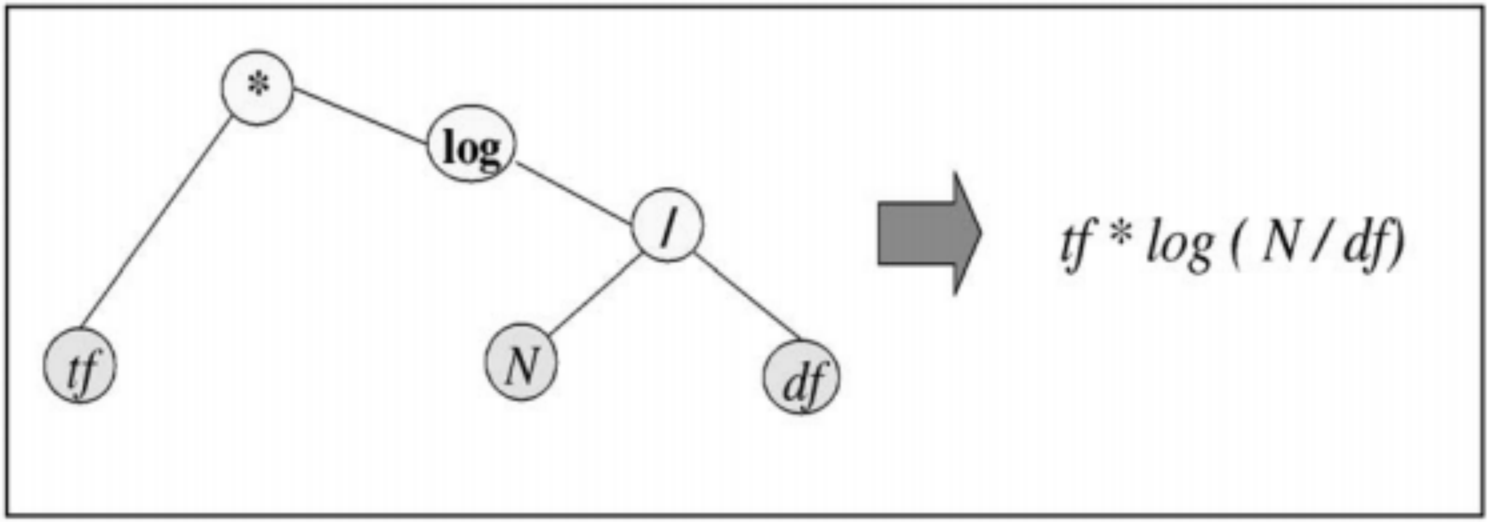
RankNet

$$C = \log(1 + e^Y) = \log\left(1 + e^{h(d_2) - h(d_1)}\right)$$

- What did we get ?
  - C is minimum if all pairs are ranked in the proper order, therefore ***by minimizing C we improve NDCG***
    - this does <u>*not*</u> imply that the *optimal solution for C is the optimal solution for NDCG or other quality measures*
  - we can compute the ***gradient of C***
    - If *h* is differentiable then also *Y* and C are
- We can directly apply steepest descent
  - Just need derivatives of h, i.e. BM25F

Genetic Algorithms

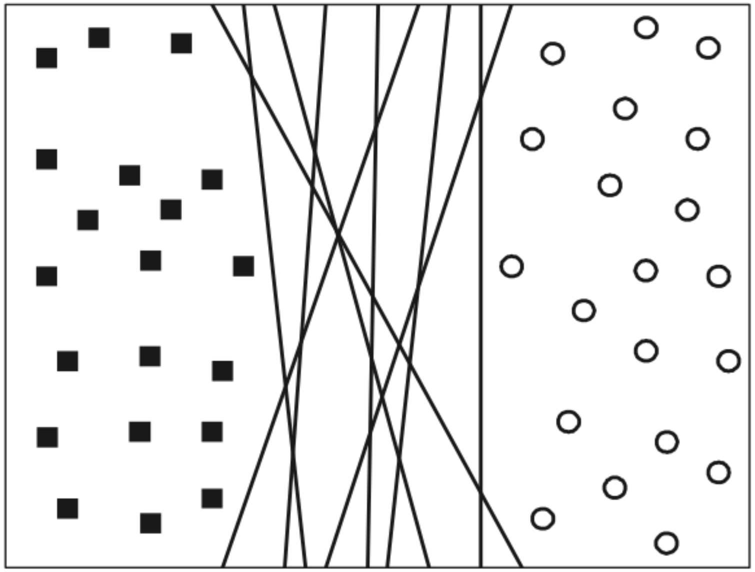
- The trick is in the representation



- Trees can represent complex functions, where nodes are operations and leaves are features
- Crossover is performed by exchanging subtrees at random

Support Vector Machines

- Classification technique, aiming at maximizing the generalization power of its classification model



  - Given the above points in a 2D space, what is the line that best "separates" the squares from the circle?

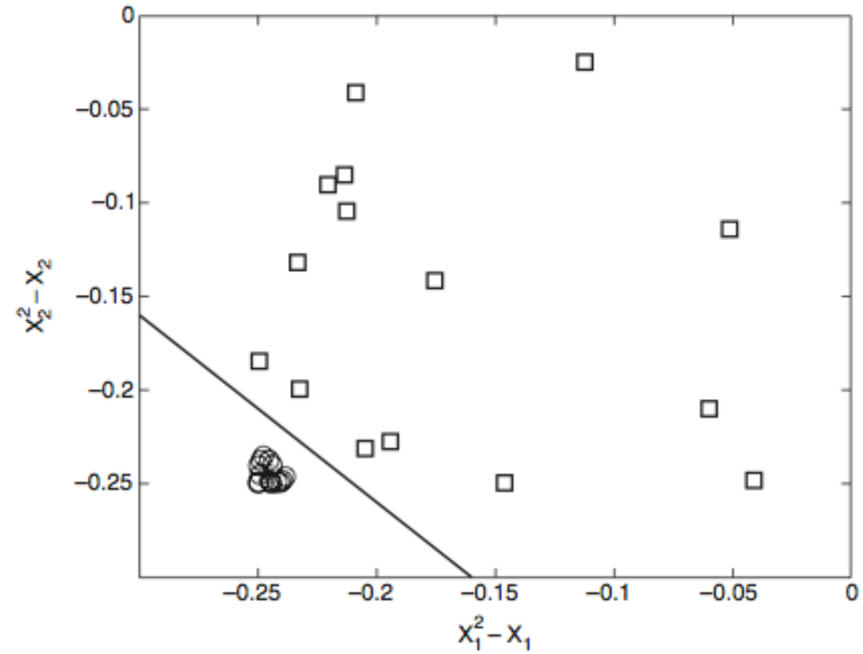
Linear SVM formulation

- Let $y_i \in \{+1,-1\}$ be the class of the *i*-th instance, the (linear) SVM (binary) classification problem is:
  - *Minimize* $\frac{1}{2}\ |w|^2$
  - *Subject to:* $y_i\ (w^T x_i + b) \geq 1$

    or: $y_i\ (w^T x_i + b) - 1 \geq 0$

  - Since the objective function is quadratic, and the constrains are linear in *w* and *b*, this is know to be a *convex optimization problem*.

Nonlinear SVM



- Idea:
  - First transform the data, potentially mapping to a space with higher dimensionality, then use a linear decision boundary as before.
  - Minimize ½ $|w|^2$
  - *Subject to:* $y_i (w^T \boxed{\Phi(x_i)} + b) \geq 1$
  - The dual is: $L_D = \sum_i \lambda_i - \frac{1}{2} \sum_{ij} \lambda_i \lambda_j y_i y_j \Phi(x_i) \Phi(x_j)$

Soft margin

- We need to relax the previous constraints, introducing *slack variables* $\xi_i \geq 0$
  - *Minimize* $\frac{1}{2} \|w\|^2 + \boxed{C \sum \xi_i}$
  - *Subject to:* $y_i (w^T x_i + b) \geq 1 \boxed{- \xi_i}$
    $\xi_i \geq 0$
  - At the same time, this relaxation must be minimized.
  - C defines the trade-off between training error and large margin
  - The problem has the same dual formulation as before, with addition constraint $0 \leq \lambda_i \leq C$

(Linear) Ranking SVM

- In case of a linear combination of features:
  $h(d) = w^T d$

- Our objective is to find *w*, such that:
  - $h(d_i) \geq h(d_j)$
  - $w^T d_i \geq w^T d_j$
  - $w^T (d_i - d_j) \geq 0$

- We approximate by adding *slack variables* ξ and minimizing this "relaxation"
  - given the *k*-th document pair, find the weights *w* such that

$$w^T (d_i - d_j) \geq 1 - \xi_k \qquad \text{with} \quad \xi_k \geq 0$$

and ξ_k is minimum

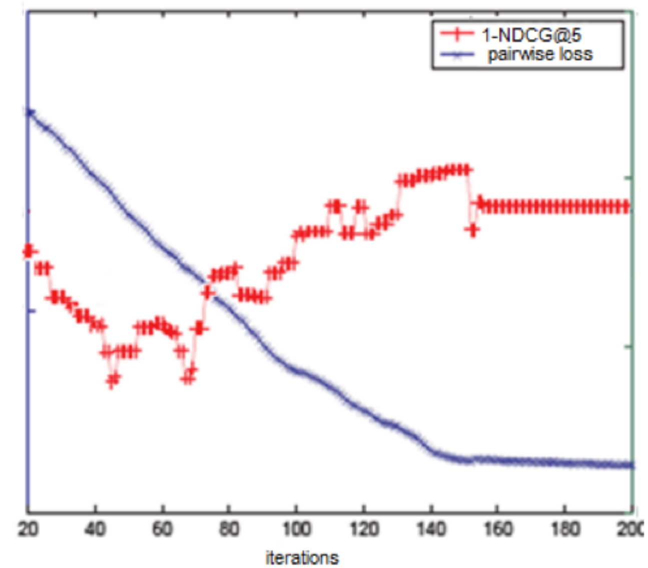
(Linear) Ranking SVM

- The full formulation of the problem is
  - *Minimize* $\frac{1}{2}\,|w|^2 + C \sum_k \xi_k$
  - *Subject to* $w^T(d_i - d_j) \geq 1 - \xi_k$

    $\xi_k \geq 0$
  - where C allows to trade-off error between the margin ($|w|^2$) and the training error ($\sum_k \xi_k$)
- This is an SVM classification problem !
  - Is convex, with no local optima, it can be generalized to non-linear functions of documents features.

Issues of the pairwise approach

- We might not realized that some queries are really badly ranked
- Top result pairs should be more important than other pairs
- In general, the number of document pairs violations, might not be a good indicator

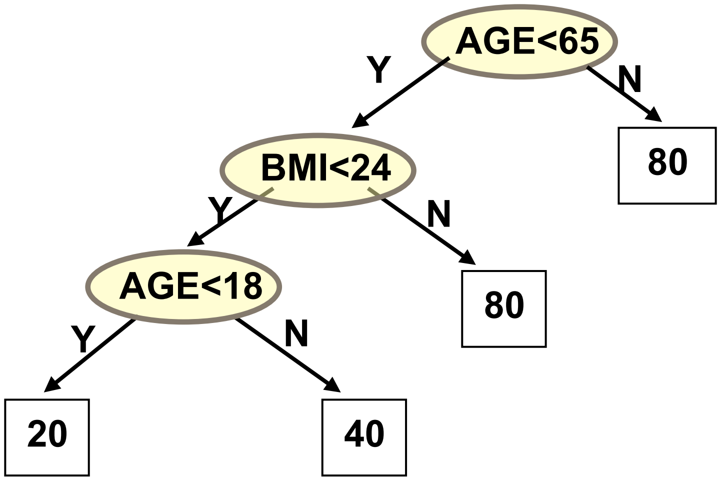


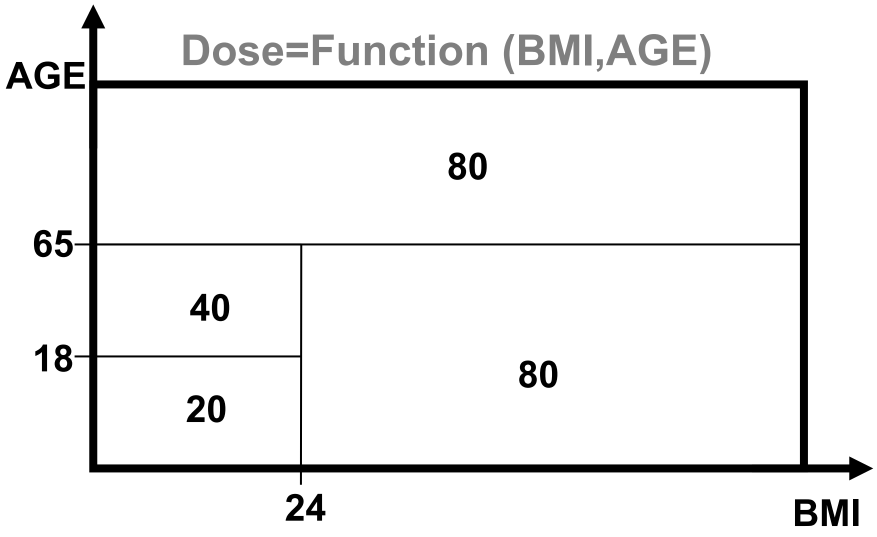
List-wise approach: Lamda-MART

- Goal:
  - Optimize the NDCG score for each query
- Tools:
  - Gradient Boosted Regression Trees
  - A modified cost function, stemming from RankNet

What is Regression Tree ?

- Machine Learning Tool for predicting a continuos variable
  - given features $X=\{X_1, \ldots, X_n\}$ predict variable Y
- A *Regression Tree* is a tree where:
  - an internal node is a predicate on some feature
  - a leaf is the prediction
  - note: every node induces a partitioning/splitting of the data
- A RT is build on the basis of some training set
  - find the tree that best predicts Y on the training data





How to choose the best split ?

- *For each attribute*:
  - *For each* possible predicate, i.e., *splitting criteria*
  - Compute the prediction for the left and right child
    - *Predicted value is the average of the target variable* on the corresponding instances
  - Compute the *goodness of the split*
    - Error reduction, usually measured as Mean Squared Error
    - New error is given by the *average distance* of the target variable from the *new prediction*: *the variance !*
  - *Take the split with the best error reduction*, i.e. *smallest variance*

- Then:
  - *Split* the data according to the chosen split criterion
  - and *repeat recursively* for generating new nodes

- Note:
  - A new node will not degrade prediction

What is a Boosted Regression Tree ? MART (multiple additive regression trees)

- We want to learn a predictor incrementally:

$$F^*(x) = \sum_{m=0}^{M} f_m(x)$$

- Input: a learning sample $\{(x_i, y_i): i=1,\ldots,N\}$
- Initialize
  - *Baseline preticts the average label value*
  - $\hat{y}_0(x) = 1/N \sum_i y_i$; $r_i = y_i$, $i=1,\ldots,N$
- For $t=1$ to M:
  - *Regression tree predicts the residual error*
  - For $i=1$ to N, compute the residuals

    $$r_i \leftarrow r_i - \hat{y}_{m-1}(x_i)$$

  - Build a regression tree from the learning sample $\{(x_i, r_i): i=1,\ldots,N\}$
  - The prediciton of the new regression tree is denoted with $\hat{y}_m$
- Return the model $\hat{y}(x) = \hat{y}_0(x) + \hat{y}_1(x) + \ldots + \hat{y}_M(x)$

- Function f_m should be easy to be learnt:
  - Decision stump: trees with one node and two leaves

What is a Gradient Boosted Regression Tree ?

- We want to learn a predictor incrementally:

$$F^*(x) = \sum_{m=0}^{M} f_m(x)$$

  - where f_m is sufficiently easy to be learnt
    - chosen from a family *H*
    - *E.g. decision stumps, or small trees*
  - each f_i reduces the error/cost function
  - f_0 is an initial guess (e.g., average)
- How to find the best f_i at each step ?
  - We use *steepest descent* and line search to find f_i

Gradient Boosting and Regression Trees

- Let $C(y_i, F_{m-1}(x_i))$ be the *error* in predicting y_i with $F_{m-1}(x_i)$ at the step *m-1*
- To improve $F_{m-1}(x_i)$ we should compute the gradient g_m of C
  - Given the gradient the new approximation should be as follows
  - $F_m(x_i) = F_{m-1}(x_i) - \gamma_m g_m$
- Note that we are looking for a tree being equivalent to the gradient of F_{m-1} !
- Since g_m *may not be in H*, we search for the *best approximation*:
  - Compute the value of gradient of the cost function at each training instance
    - This is independent from the fact that F_{m-1} is a tree
  - Find the tree *h in H* that best approximates g_m
    - This is a simple regression tree learning
- Finally, *line search* is used to find the best weight of the tree
- The new estimated score function F_m is:

$$F_m(x) = F_{m-1}(x) + \rho_m h_m(x)$$

GBRT can optimize any cost function… so which one ?

- Recall the RankNet cost function

$$C = \log(1+e^{Y}) = \log\left(1+e^{h(d_2)-h(d_1)}\right)$$

- Let's denote with *w* the parameters of *h*

$$\frac{\partial C}{\partial w} = \frac{\partial C}{\partial h(d_1)}\frac{\partial h(d_1)}{\partial w} + \frac{\partial C}{\partial h(d_2)}\frac{\partial h(d_2)}{\partial w} = \frac{1}{1+e^{-Y}}\left(\frac{\partial h(d_1)}{\partial w} - \frac{\partial h(d_2)}{\partial w}\right)$$

  - where we define:

$$\lambda_{12} = \frac{1}{1+e^{-Y}}$$

- The update rule of the weights *w* with steepest descent is:

$$\delta w = -\rho \sum_{ij}\left(\lambda_{ij}\frac{\partial h(d_i)}{\partial w} - \lambda_{ij}\frac{\partial h(d_j)}{\partial w}\right)$$

- equivalently

$$\delta w = -\rho \sum_{i} \lambda_i \frac{\partial h(d_i)}{\partial w} \qquad \lambda_i = \sum_{d_i \succ d_j} \lambda_{ij} - \sum_{d_j \succ d_i} \lambda_{ij}$$

What did we get ?

$$\lambda_i = \sum_{d_i \succ d_j} \lambda_{ij} - \sum_{d_j \succ d_i} \lambda_{ij}$$

- λ_i is a single *magic number* for each URL assessing whether it is *well ranked* and *how much far is from it*

- Note that λ_i depends on number of violated pairwise constraints
  - Becasue it comes directly from the RankNet cost

From left to right, the number of pairwise violations decreases from 13 to 7 (good for RankNet)

Black arrows are RankNet Gradients, read are what we want

How to optimize NDCG ?

- Observation 1:
  - GBRT only need to be able to compute gradients of the cost function
- Observation 2:
  - λ_{ij} are exactly the gradients of the cost function w.r.t. the document scoring function *h*
- Conclusion 1:
  - We can plug λ_{ij} into a GBRT so that at each iteration a new tree is found that approximates λ_{ij}
- Observation 2:
  - Since we want to optimize NDCG, we can improve λ_{ij} so that they take into account the change in NDCG due to swapping *i* with *j*
- Result:

$$\lambda_{ij} = \frac{1}{1+e^{-Y}} \left|\Delta_{NDCG}\right| = \frac{1}{1+e^{-Y}} \left(2^{l_i} - 2^{l_j}\right) \left(\log\left(\frac{1}{1+i}\right) - \log\left(\frac{1}{1+j}\right)\right)$$

Lambda-MART

- Input: a learning sample $\{(x_i, y_i): i=1,\ldots,N\}$
- Initialize
  - *Baseline pretícts the average label value*
  - $\hat{y}_0(x) = 1/N \sum_i y_i$; $r_i = y_i,\ i=1,\ldots,N$
- For $t=1$ to M:
  - *Regression tree predicts the corrected lambdas*
  - For $i=1$ to N, compute the *pseudo-residuals*

    $$r_i \leftarrow \lambda_i$$
  - Build a regression tree from the learning sample $\{(x_i, r_i): i=1,\ldots,N\}$
  - The prediciton of the new regression tree is denoted with $\hat{y}_m$
- Return the model $\hat{y}(x) = \hat{y}_0(x) + \hat{y}_1(x) + \ldots + \hat{y}_M(x)$

- Note that the final prediction is not close to y_i, but, since it optimized lambdas, it optimizes the final NDCG.

Performance

	Validation		Test	
	ERR	NDCG	ERR	NDCG
BM25F-SD	0.42598	0.73231	0.42853	0.73214
RankSVM	0.43109	0.75156	0.43680	0.75924
GBDT	0.45625	0.78608	0.46201	0.79013

- Results are from the Yahoo! Learning to rank challenge
- The winner of the challenge used a combination of several Lambda-MART models

How to Exploit User feedback

- Explicit
  - Ask users to rate result
    - (by the page or by the snippet)
- Implicit
  - Process logs to get information about:
    - Clicks
    - Query reformulation
- Fancier…
  - Eye tracking
    - Fixation: spatially stable gaze lasting for approximately 200–300 ms
- Goals:
  - Build a training set
  - Evaluate our search engine

Experiment Set-up

- Phase I:
  - Use Google to answer 10 questions
    - 34 user recruited
  - Is there any rank bias ?
- Phase II:
  - Answer the same questions with a "modified Google"
    - 27 users recruited
  - Modifications:
    - SWAPPED: swap the top 2 results
    - REVERSED: reverse top-10 results

Questions

Table I. Questions Used in the Study and the Average Number of Queries and Clicks per Question and Subject

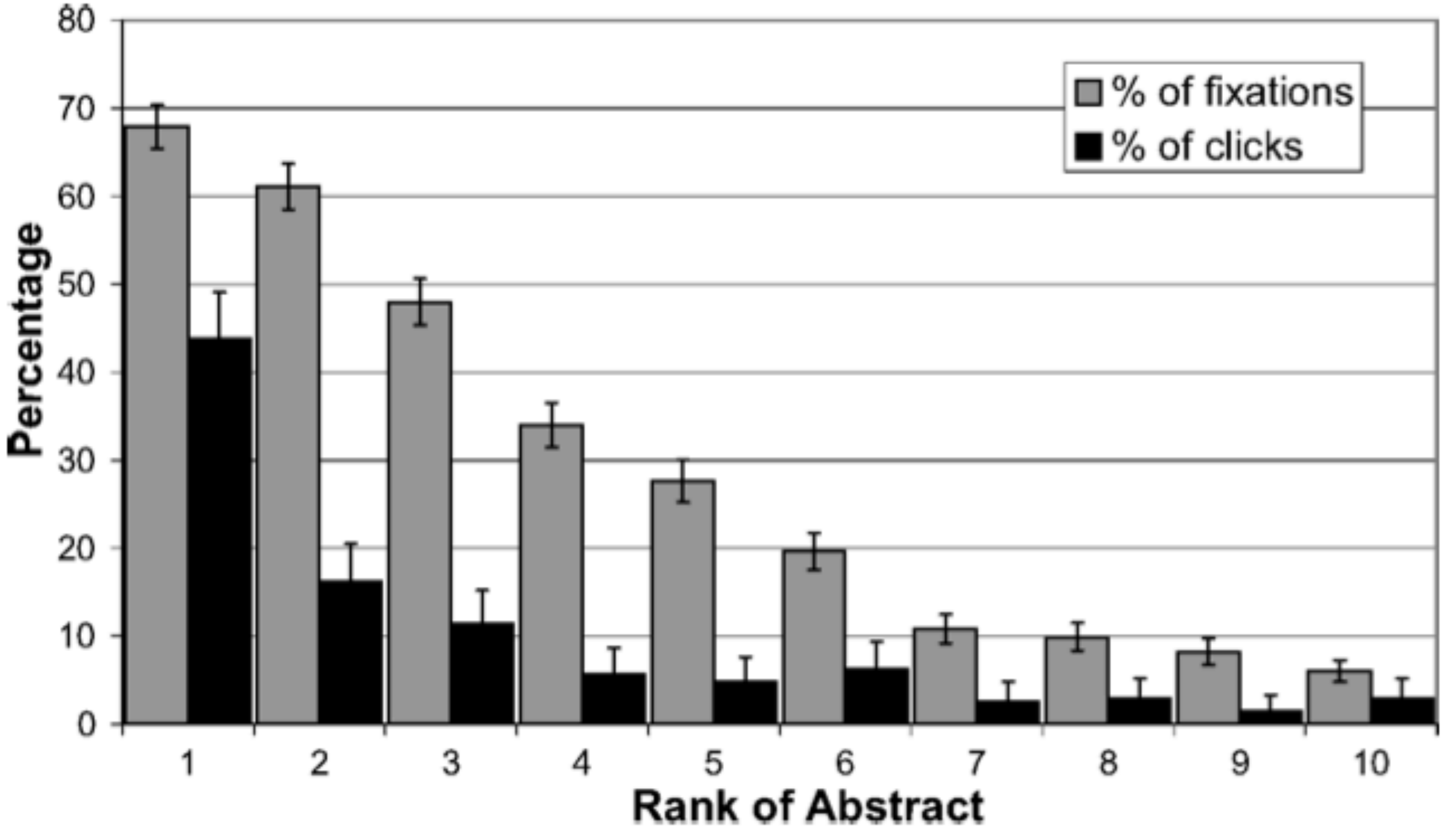
			Phase I		Phase II	
	No.	Question	#Queries	#Clicks	#Queries	#Clicks
Navigational	1.	Find the homepage of Michael Jordan, the statistician.	2.8	1.6	2.6	1.7
	2.	Find the page displaying the route map for Greyhound buses.	1.3	1.5	1.6	1.6
	3.	Find the homepage of the 1000 Acres Dude Ranch.	2.2	2.6	2.2	1.9
	4.	Find the homepage for graduate housing at Carnegie Mellon University.	2.0	1.7	2.2	0.9
	5.	Find the homepage of Emeril—the chef who has a television cooking program.	1.9	1.6	3.0	1.8
Informational	6.	Where is the tallest mountain in New York located?	1.7	2.0	2.0	1.6
	7.	With the heavy coverage of the Democratic presidential primaries, you are excited to cast your vote for a candidate. When are Democratic presidential primaries in New York?	1.6	1.8	1.6	2.1
	8.	Which actor starred as the main character in the original *Time Machine* movie?	1.8	1.8	1.9	1.9
	9.	A friend told you that Mr. Cornell used to live close to campus—near University and Steward Ave. Does anybody live in his house now? If so, who?	2.0	1.5	2.9	1.6
	10.	What is the name of the researcher who discovered the first modern antibiotic?	2.0	2.0	2.3	1.6

- Phase I: 1.9 queries per question, 0.9 clicks per query
- Phase II: 2.2 queries per question, 0.8 clicks per query

Explicit Feedback

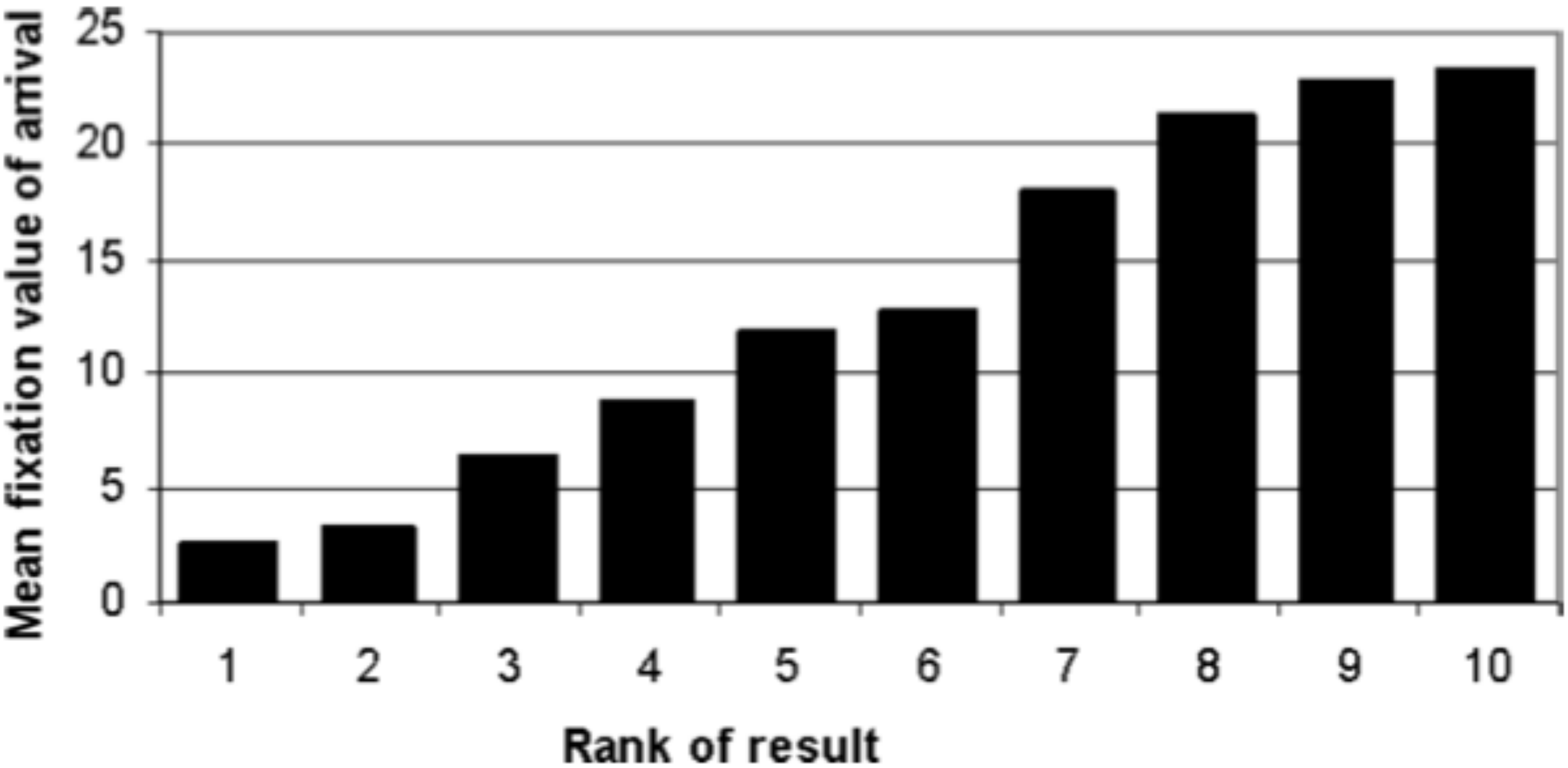
- Phase I:
  - “order the results by how promising their abstracts are for leading to information that is relevant to answering the question “
- Phase II:
  - same as Phase I
  - Assessment of results by looking at the webpage without any provided snippet

Which Links Did Users View and Click?



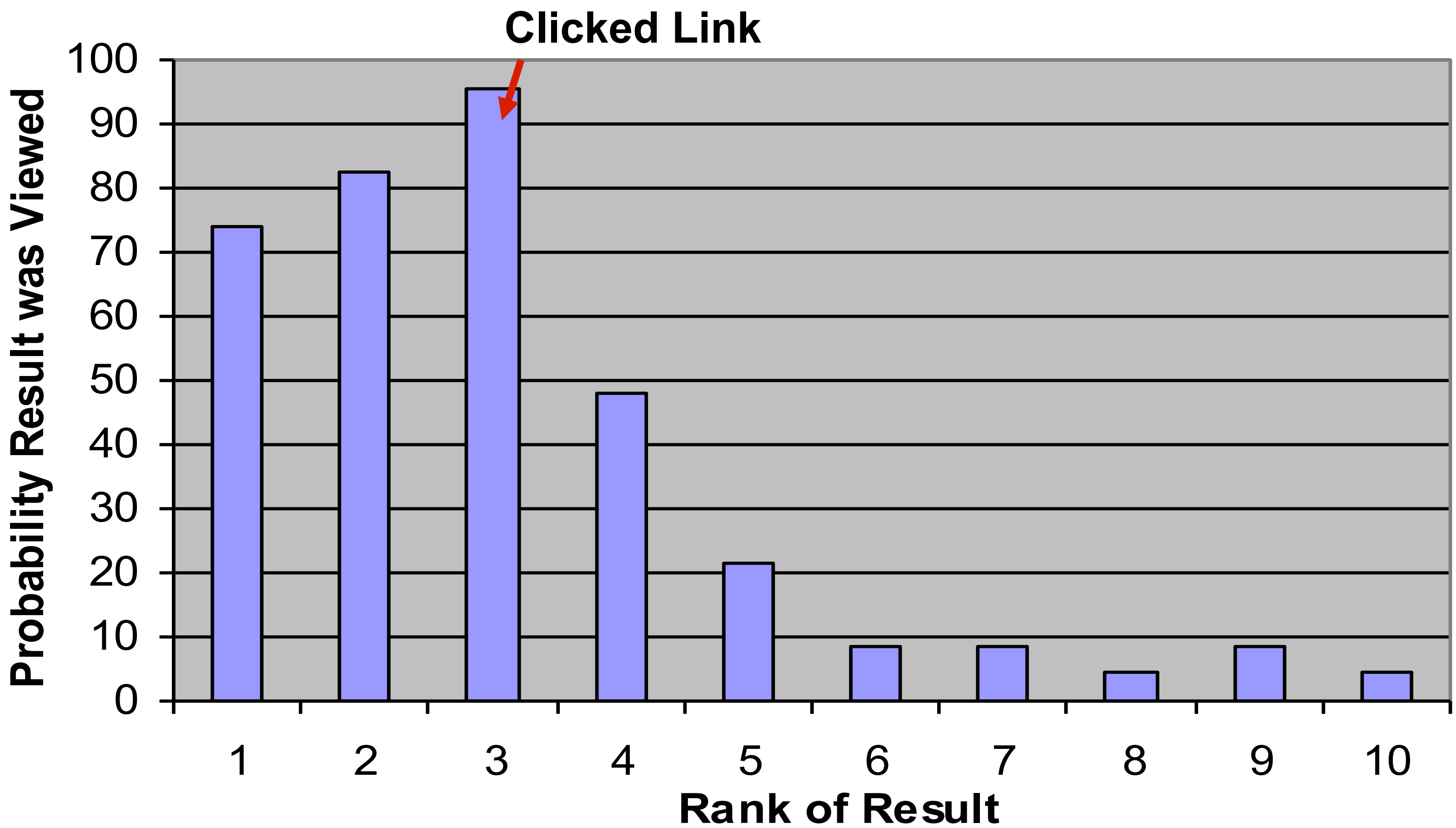
- First result receives a large number of clicks w.r.t. to the number of fixations
- There is drop after page scroll

Did Users Scan Links from Top to Bottom?



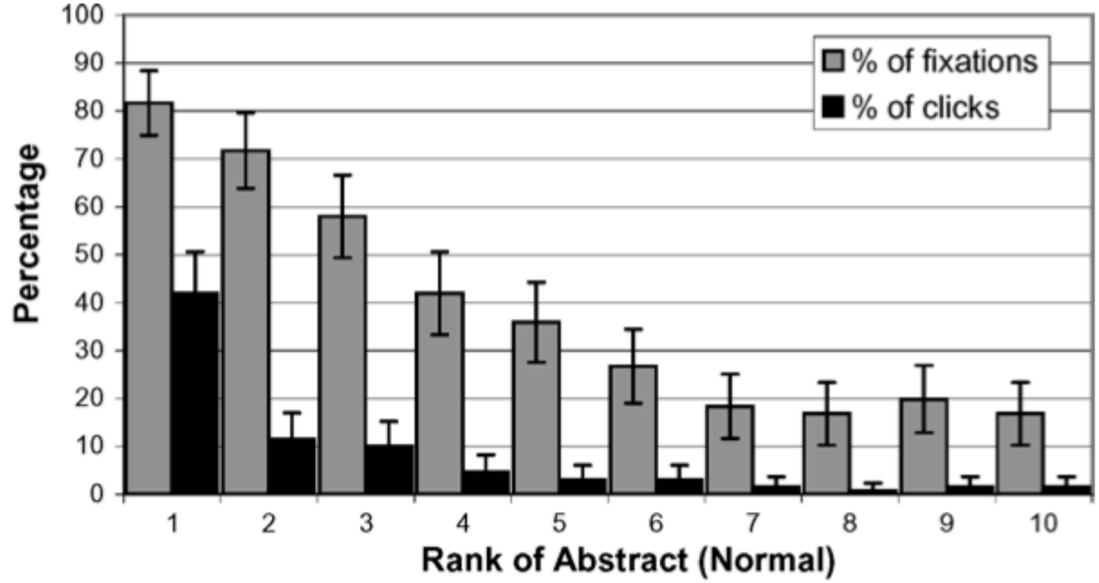
- Yes, but the first 2 results are seen almost at the same time
- Scroll is after the 6th result

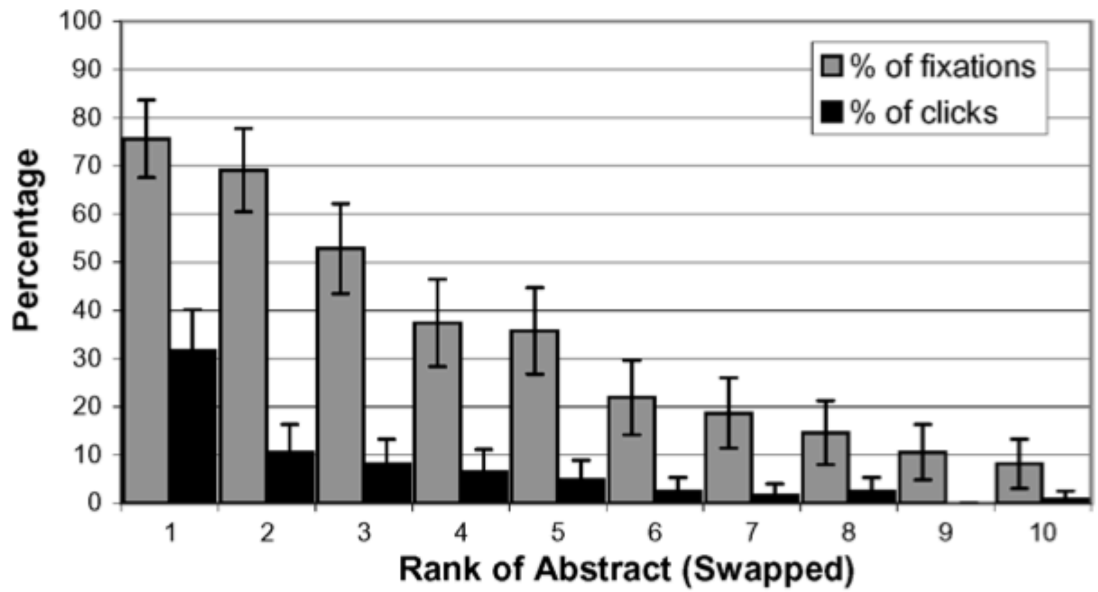
Which Links Did Users Evaluate Before Clicking?



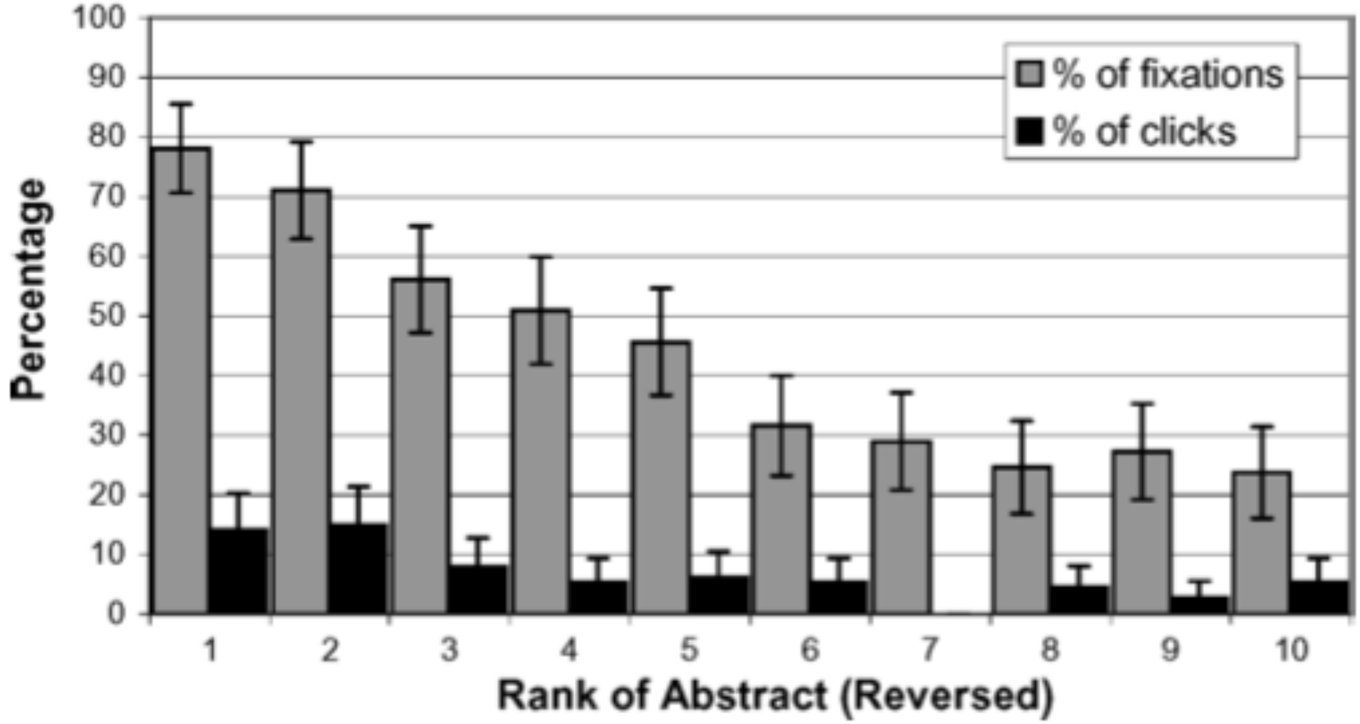
- Users check most of the results above the click
- Almost no attention below the click
- An exception is the first link below the click

Does Relevance Influence User Decisions?





Does Relevance Influence User Decisions?



- Average number of clicks changes from 2.1 to 2.45
- The quality of the system impact on the clicks
- *Trust bias* and *quality bias* make it difficult to use clicks as an absolute measure of result quality

Are Clicks Relative Relevance Judgments Within One Results Page?

- Can we use clicks to compare results ?
- Idea:
  - exploit clicked and non clicked results

- ***Strategy 1**: CLICK > SKIP ABOVE*
- Example:
  - l_1^* l_2 l_3^* l_4 l_5^* l_6 l_7
  - $l_3 > l_2$, $l_5 > l_4$, $l_5 > l_2$

- Measure the goodness of these constraints as the ratio of agreement with relevance judgments

Are Clicks Relative Relevance Judgments Within One Results Page?

- Idea:
  - Latest click is the most important
- ***Strategy 2****: LAST CLICK > SKIP ABOVE*
- Example:
  - l_1* l_2 l_3* l_4 l_5* l_6 l_7
  - $l_5 > l_4$, $l_5 > l_2$
- Idea:
  - Earlier clicks are less important
- ***Strategy 3****: CLICK > EARLIER CLICK*
- Example:
  - l_1* l_2 l_3* l_4 l_5* l_6 l_7 (l_3 then l_1 then l_5)
  - $l_1 > l_3$, $l_5 > l_1$, $l_5 > l_3$

Are Clicks Relative Relevance Judgments Within One Results Page?

- Idea:
  - Previous result receives lot of attention
- ***Strategy 4***: *CLICK > SKIP PREVIOUS*
- Example:
  - $l_1{}^*$ l_2 $l_3{}^*$ l_4 $l_5{}^*$ l_6 l_7
  - $l_3>l_2$, $l_5>l_4$
- Idea:
  - Next result receives lot of attention
- ***Strategy 5***: *CLICK > NO-CLICK NEXT*
- Example:
  - $l_1{}^*$ l_2 $l_3{}^*$ l_4 $l_5{}^*$ l_6 l_7
  - $l_1>l_2$, $l_3>l_4$, $l_5>l_6$

Explicit Feedback Data Strategy	p/q	Abstracts Phase I Normal	Abstracts Phase II Normal	Swapped	Reversed	All	Pages Phase II All
Interjudge agreem.	N/A	89.5	N/A	N/A	N/A	82.5	86.4
Click > Skip Above	1.37	80.8 ± 3.6	88.0 ± 9.5	79.6 ± 8.9	83.0 ± 6.7	83.1 ± 4.4	78.2 ± 5.6
LastClick > SkipAbove	1.18	83.1 ± 3.8	89.7 ± 9.8	77.9 ± 9.9	84.6 ± 6.9	83.8 ± 4.6	80.9 ± 5.1
Click > Earlier Click	0.20	67.2 ± 12.3	75.0 ± 25.8	36.8 ± 22.9	28.6 ± 27.5	46.9 ± 13.9	64.3 ± 15.4
Click > Skip Previous	0.37	82.3 ± 7.3	88.9 ± 24.1	80.0 ± 18.0	79.5 ± 15.4	81.6 ± 9.5	80.7 ± 9.6
Click > No Click Next	0.68	84.1 ± 4.9	75.6 ± 14.5	66.7 ± 13.1	70.0 ± 15.7	70.4 ± 8.0	67.4 ± 8.2

- *CLICK > SKIP ABOVE*: performs well, close to the judge agreement
- *LAST CLICK > SKIP ABOVE: slightly improves*
- *CLICK > EARLIER CLICK: not performing well*
- *CLICK > SKIP PREVIOUS: No statistically significant difference with CLICK > SKIP ABOVE*
- *CLICK > NO-CLICK NEXT: is it useful ?*

Are Clicks Relative Relevance Judgments Within a Query Chain?

- Observations:
  - Clicked top queries are not very involved in the generated frequencies
  - Users run sequence of queries before satisfying their information need

- ***Strategy 1***: *CLICK > SKIP EARLIER*
- ***Strategy 2***: *LAST CLICK > SKIP EARLIER*
- ***Strategy 3***: *CLICK > CLICK EARLIER*
- ***Strategy 4***: *CLICK > TOP 1 NO CLICK EARLIER*
- ***Strategy 5***: *CLICK > TOP 2 NO CLICK EARLIER*
- ***Strategy 6***: *TOP 1 > TOP 1 EARLIER*

Are Clicks Relative Relevance Judgments Within a Query Chain?

Explicit Feedback Data Strategy	p/q	Abstracts Phase II Normal	Abstracts Phase II Swapped	Abstracts Phase II Reversed	Abstracts Phase II All	Pages Phase II All
Click > Skip Earlier QC	0.49	84.5 ± 16.4	71.1 ± 17.0	54.6 ± 18.1	70.2 ± 9.7	68.0 ± 8.4
Last Click > Skip Earlier QC	0.33	77.3 ± 20.6	80.8 ± 20.2	42.1 ± 24.4	68.7 ± 12.6	66.2 ± 12.2
Click > Click Earlier QC	0.30	61.9 ± 23.5	51.2 ± 17.1	35.3 ± 26.4	50.6 ± 11.4	65.8 ± 11.8
Click > TopOne NoClickEarl. QC	0.35	86.4 ± 21.2	77.3 ± 15.1	92.6 ± 16.9	83.9 ± 9.1	85.4 ± 8.7
Click > TopTwo NoClickEarl. QC	0.70	88.9 ± 12.9	80.0 ± 10.1	86.8 ± 12.1	84.2 ± 6.1	84.5 ± 6.1
TopOne > TopOne Earlier QC	0.84	65.3 ± 15.2	68.2 ± 12.7	75.6 ± 15.1	69.4 ± 7.8	69.4 ± 7.9

- The performance of *CLICK > TOP 2 NO CLICK EARLIER* suggest that query reformulation is a strong evidence of document poor quality

Software tools

- RankLib:
  - http://sourceforge.net/p/lemur/wiki/RankLib/

Usage: java -jar RankLib.jar <Params>	
Params:	
[+] Training (+ tuning and evaluation)	
-train <file>	Training data
-ranker <type>	Specify which ranking algorithm to use
	0: MART (gradient boosted regression tree)
	1: RankNet
	2: RankBoost
	3: AdaRank
	4: Coordinate Ascent
	6: LambdaMART
	7: ListNet
	8: Random Forests

Conclusions

- Machine learning frameworks are necessary for modern web search engines
- Creating a training dataset is expensive
  - Potentially requires users to evaluate a large number of queries and results
- Click data can be successfully transformed in pair-wise preferences:
  - To estimate the quality of the system
  - To create a training set of a learning-to-rank approach
- Several approaches have been developed
  - They succeed in the non trivial task of optimizing complex IT evaluation measures such as NDCG.

The End

claudio.lucchese@isti.cnr.it