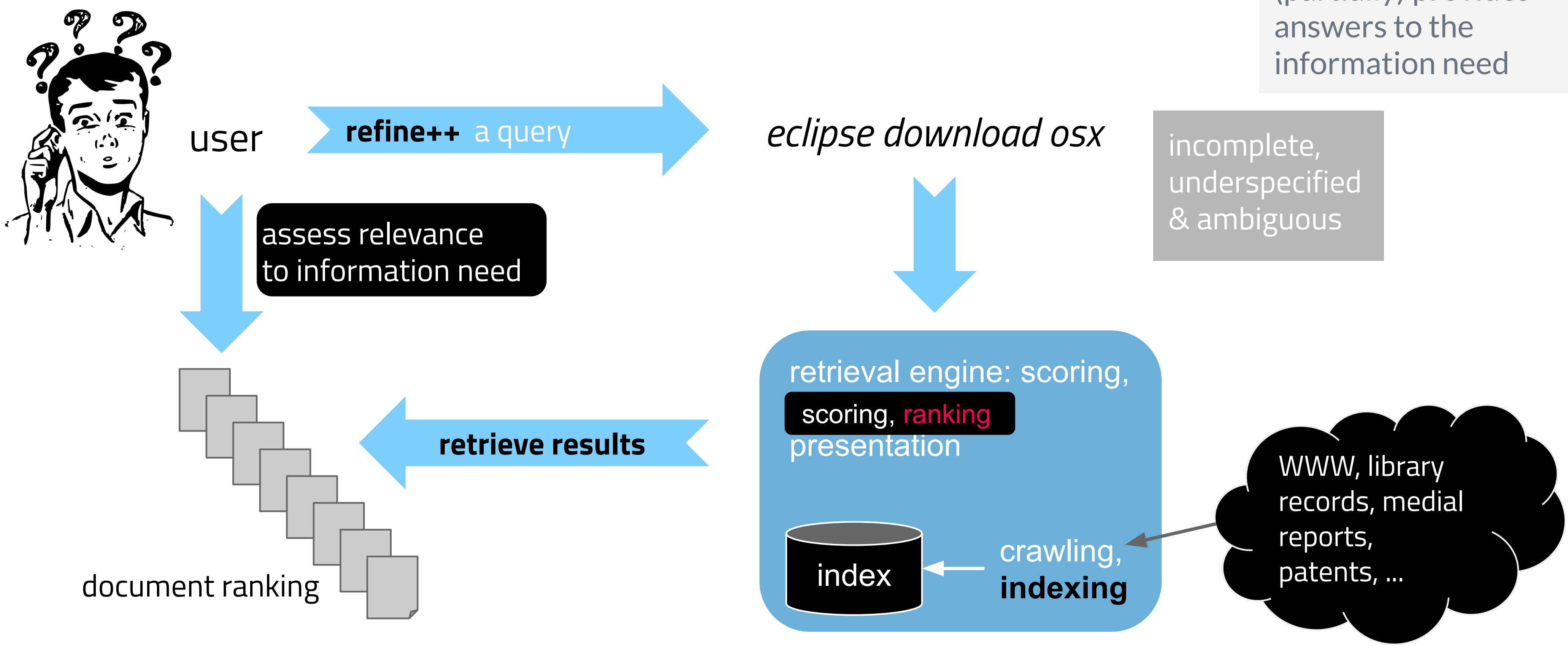
IN4325
Learning to Rank (L2R)

Claudia Hauff (WIS, TU Delft)

The big picture

The essence of IR

Information need: *Looks like I need Eclipse for this job. Where can I download the latest beta version for macOS Sierra?*



Information need
Topic the user wants to know more about

Query
Translation of need into an input for the search engine

Relevance
A document is relevant if it (partially) provides answers to the information need

Tutorial: https://www.nowpublishers.com/article/Details/INR-016

L2R

Conventional ranking models in IR

Query-dependent models

Vector space model

Boolean model

BM25

Language Modeling

...

Query-independent models

PageRank

TrustRank

Spaminess

Readability

...

How can we combine a large number of models (continuously proposed in the literature) to obtain an even better model?

Overview

Learning-to-rank

in the broad sense are all methods that use machine learning to solve the problem of ranking

(e.g. relevance feedback, hyperparameter tuning of BM25 …)

Learning-to-rank

in the narrow sense are all methods that learn the optimal way to combine **features** extracted from query-document pairs through **discriminative** training

Learns the conditional probability distribution $P(y|x)$.

Generative training learns $P(x,y)$ instead.

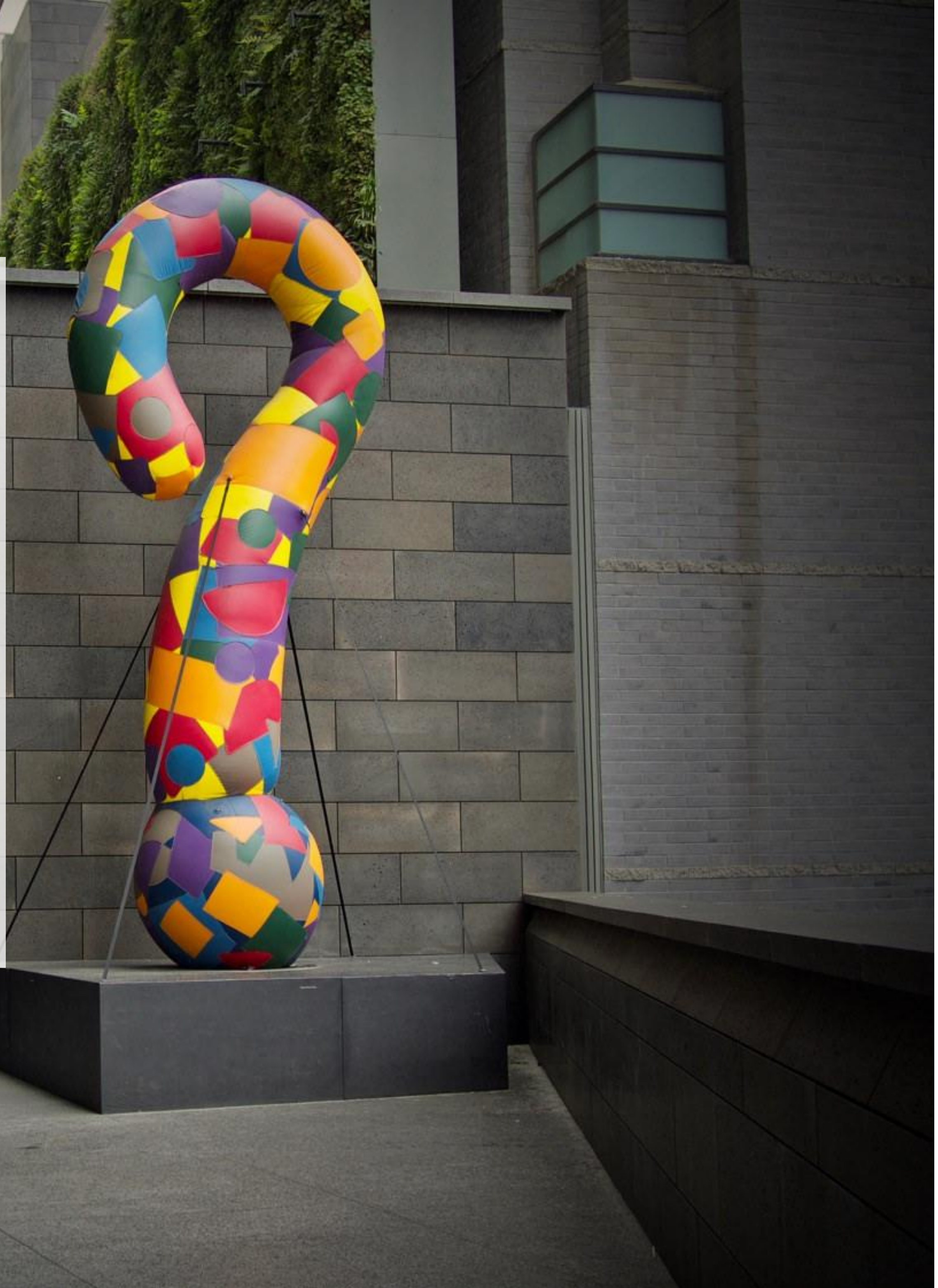
Given your machine learning background, how would you go about using machine learning to **learn a ranking function**?

Ignore deep learning for now ....

What is the input?

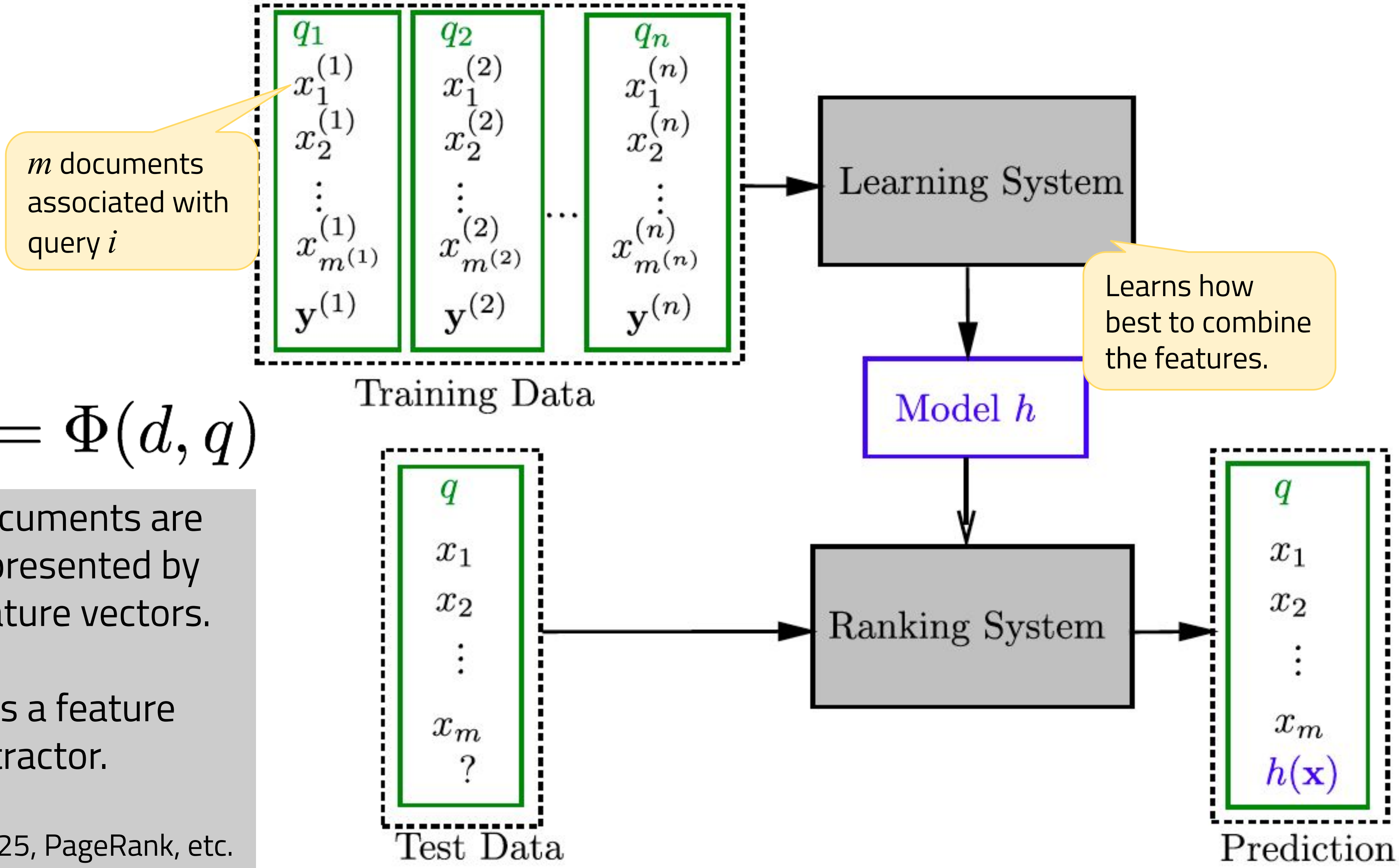
What is the output?

What is your success metric?



L2R setup

Supervised or unsupervised



$$x = \Phi(d, q)$$

Documents are represented by feature vectors.

Φ is a feature extractor.

BM25, PageRank, etc. can be a feature.

Image: https://www.nowpublishers.com/article/Details/INR-016

Document judgment strategies

with respect to a query (topic)

- Specifying whether a document is relevant (binary) or specifying a degree of relevance (`Fair > Bad`)

- Specifying whether a document is *more* relevant than another one (relative preference)

- Specifying the partial or total order of documents (a set of permutations)

UQV100: many queries per topic

You have heard quite a lot about cheap computing as being the way of the future, including one recent model called a Raspberry Pi. You start thinking about buying one, and wonder how much they cost.

amazon raspberry pi

best deal raspberry pi computer

buy Raspberry Pi

buying a raspberry pi price

cheap Raspberry Pi

cost of raspberry pi computing model

how much a Raspberry Pi?

how much Raspberry Pi

Pi cost

price comparions for 'Raspberry Pi' computer

100 topics of the 2013/14 TREC Web track

10,835 queries were collected from 263 crowd workers

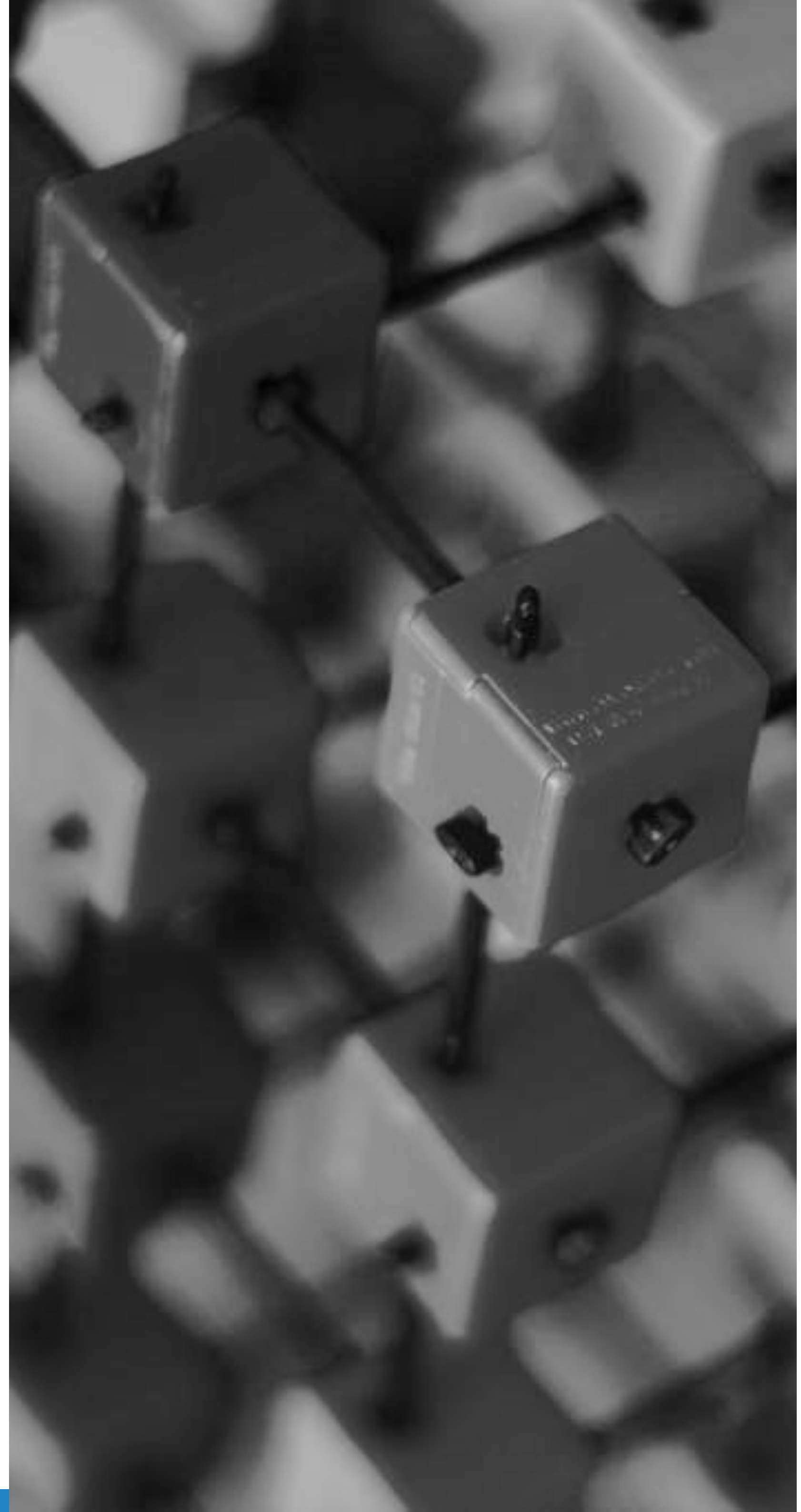
Relevance judgments on a depth pool of 10 (based on Indri-BM25)

UQV100: many queries per topic

LETOR features

LEarning **TO R**ank for Information Retrieval

TF(Term frequency) of body	BM25 of whole document
TF of anchor	LMIR.ABS of body
TF of title	LMIR.ABS of anchor
TF of URL	LMIR.ABS of title
TF of whole document	LMIR.ABS of URL
IDF(Inverse document frequency) of body	LMIR.ABS of whole document
IDF of anchor	LMIR.DIR of body
IDF of title	LMIR.DIR of anchor
IDF of URL	LMIR.DIR of title
IDF of whole document	LMIR.DIR of URL
TF*IDF of body	LMIR.DIR of whole document
TF*IDF of anchor	LMIR.JM of body
TF*IDF of title	LMIR.JM of anchor
TF*IDF of URL	LMIR.JM of title
TF*IDF of whole document	LMIR.JM of URL
DL(Document length) of body	LMIR.JM of whole document
DL of anchor	PageRank
DL of title	Inlink number
DL of URL	Outlink number
DL of whole document	Number of slash in URL
BM25 of body	Length of URL
BM25 of anchor	Number of child page
BM25 of title	
BM25 of URL	



https://arxiv.org/ftp/arxiv/papers/1306/1306.2597.pdf

Approaches

Input space (feature vectors)
Output space (learning target)
Hypothesis space
Loss function (prediction vs. ground truth)

Pointwise approach

Each document for itself.

Input space: feature vector of each doc.

Output space: relevance degree of each document

Hypothesis space: functions that take a doc. feature vector as input and output a relevance degree

Regression or classification loss.

Pairwise approach

Each doc. pair for itself.

Input space: feature vectors of a pair of docs

Output space: pairwise preferences

Hypothesis space: functions that take a document pair as input and outputs their relative order

Loss function considers the relative order between the two docs.

Listwise approach

Designed for ranking

Input space:

$$\mathbf{x} = \{x_j\}_{j=1}^{m}$$

Output space: (1) relevance degrees of all documents, (2) ranked list of documents

Hypothesis space: functions that take $\boldsymbol{x}$ as input and produce (1) or (2)

Loss function considers (1) or (2)

Approaches

Input space (feature vectors)
Output space (learning target)
Hypothesis space
Loss function (prediction vs. ground truth)

Pointwise approach

Each document for itself.

Input space: feature vector of each doc.

Output space: relevance degree of each document

Hypothesis space: functions th... doc. feature... input and ou... relevance degree

Regression or classification loss.

How to rank a whole set of documents? Another step is needed.

Pairwise approach

Each doc. pair for itself.

Input space: feature vectors of a pair of docs

Output space: pairwise preferences

Hypothesis space: functions that take a ...t pair as input ...uts their ...rder

Loss function considers the relative order between the two docs.

Listwise approach

Designed for ranking.

Input space:

$$\mathbf{x} = \{x_j\}_{j=1}^{m}$$

Output space: (1) relevance degrees of all documents, (2) ranked list of documents

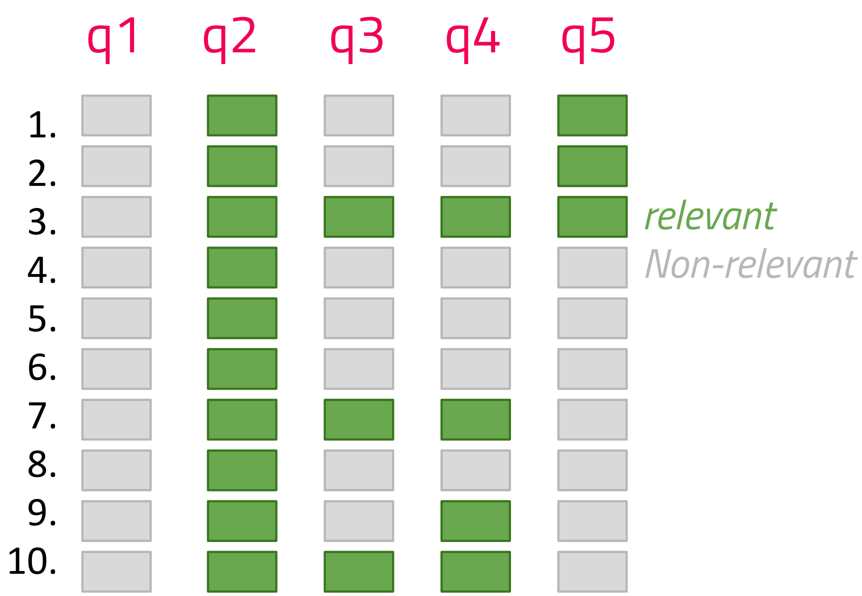
Hypothesis space: functions that take $\boldsymbol{x}$ as input and produce (1) or (2)

Loss function considers (1) or (2)

Document ranking is easy.

Different loss functions but one and the same evaluation metric (e.g. MAP)

Mean Average Precision



AvP 0.0 1.0 0.09 0.13 0.3
(assume R=10)

MAP = 0.364

One system, five queries

Given a set of queries, the average effectiveness is the mean over AvP.

MAP remains one of the most commonly employed retrieval evaluation measure to this day.

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} \frac{\sum_{k=1}^{s} P@k \times rel(k)}{R}$$

Recall

Normalized Discounted Cumulative Gain (NDCG)

Standard Web search queries are short (2-3 terms), e.g. “cheap internet”, “dinosaurs”, “solar panels”

Graded relevance scales needed (e.g. 0-3); NDCG measures the “gain” of documents

Relevance score assessors gave D at query j

Normalization so that a perfect ranking at k for query j is 1

$$NDCG(Q,k) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} Z_{kj} \sum_{m=1}^{k} \frac{2^{R(j,m)}-1}{\log_2(1+m)}$$

Assumptions:

- Highly relevant documents are more valuable than marginally relevant documents
- The greater the ranked position of a relevant document, the less valuable it is for the user
    - Few users go further than the first 10 blue links
    - Probability of reaching the document is lower
    - Users have limited time
    - Users may have seen the information in the document already

Key questions

- How do the proposed algorithms **differ**? What are their **strengths** and **weaknesses**?
- What are the **theoretical issues** for ranking that we should focus on?
- Which of the proposed learning to rank algorithms perform best empirically?

L2R categorization

	SVM	Boosting	Neural net	Others
Pointwise		McRank		PRank
Pairwise	RankSVM	RankBoost, LambdaMART, GBRank	RankNet, LambdaRank, FRank	
Listwise	SVM MAP	AdaRank	ListNet	SoftRank, SmoothRank

Tax et al. (2015): "*ListNet, SmoothRank, FenchelRank, FSMRank, LRUF and LARF are Pareto optimal learning to rank methods*"

[3] Tax, Niek, Sander Bockting, and Djoerd Hiemstra. "A cross-benchmark comparison of **87 learning to rank methods.**" Information processing & management 51.6 (2015): 757-772.

Pointwise approach

Direct application of standard supervised ML.

Pointwise categories

- Regression based algorithms

  Real-valued relevance scores

- Classification based algorithms

  Non-ordered categories

- Ordinal regression based algorithms

  Variables with a natural categorical ordering

Polynomial regression function

Main issue of regression?

Qualitative judgment is encoded quantitatively

Given q and associated $\mathbf{x} = \{x_j\}_{j=1}^{m}$,

let the ground truth label be:

doc judged non-relevant

doc judged relevant

binary: $\vec{y_j} = (0, 1)$ *or* $\vec{y_j} = (1, 0)$

ordered categories: $\vec{y_j} = (0, 0, .., 1, ..0)$

doc judged belonging to a category (e.g. 'Fair' or 'Good')

Predictor of kth element in the ground truth vec.

Scoring function: $\vec{f} = (f_1, f_2, ..., \boxed{f_k})$ *with*

Tth feature in feature vector j

$$f_k(x_j) = w_{k,0} + w_{k,1} \times x_{j,1} + ... + w_{k,T} \times \boxed{x_{j,T}}$$
$$+ \boxed{w_{k,T+1}} \times x_{j,1}^2 + w_{k,T+2} \times x_{j,1} \times x_{j,2} + ...$$

Combination coefficient

Loss function: $L(\vec{f}; x_j, \vec{y_j}) = ||\vec{y_j} - \vec{f}(x_j)||^2$

Pairwise approach

Pairwise

Many algorithms have been proposed, e.g.

- RankNet
- RankBoost
- Ranking SVM
- LambdaRank
- ...

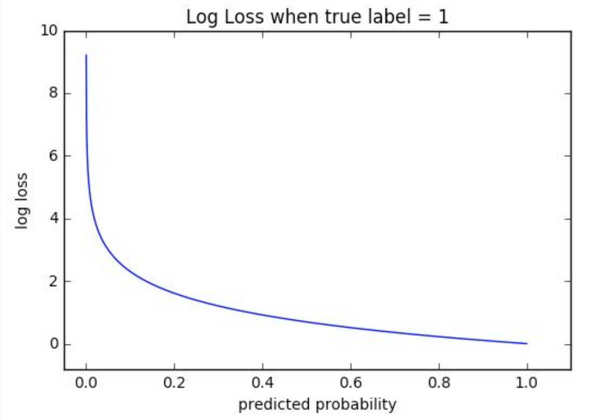
Focus: **relative ordering** of pairs of documents

Ranking problem **reduced** to a classification problem
(goal: minimize #misclassified pairs)

Training data:

$$\{(x_1, x_2, +1), (x_1, x_3, -1), \ldots, (x_i, x_j, +1), \ldots\}$$

RankNet



$$Given\ q\ and\ two\ documents\ x_u\ and\ x_v,$$

$$\textbf{modeled prob: } \boxed{P_{u,v}(f)} = \frac{\exp(f(x_u) - f(x_v))}{1 + \exp(f(x_u) - f(x_v))}$$

Based on the diff. between the two documents' scores

Shallow neural network learns scoring function f; gradient descent as optimization alg

$$\boxed{L(f; x_u, x_v, y_{u,v})} = -\boxed{\bar{P}_{u,v}} \log P_{u,v}(f) - (1 - \boxed{\bar{P}_{u,v}}) \log(1 - P_{u,v}(f))$$

Cross-entropy loss

Target probability:

$\bar{P}_{u,v} = 1,\ if\ y_{u,v} = 1;$

$\bar{P}_{u,v} = 0\ otherwise$

Why is convexity important?

RankNet - loss function issue

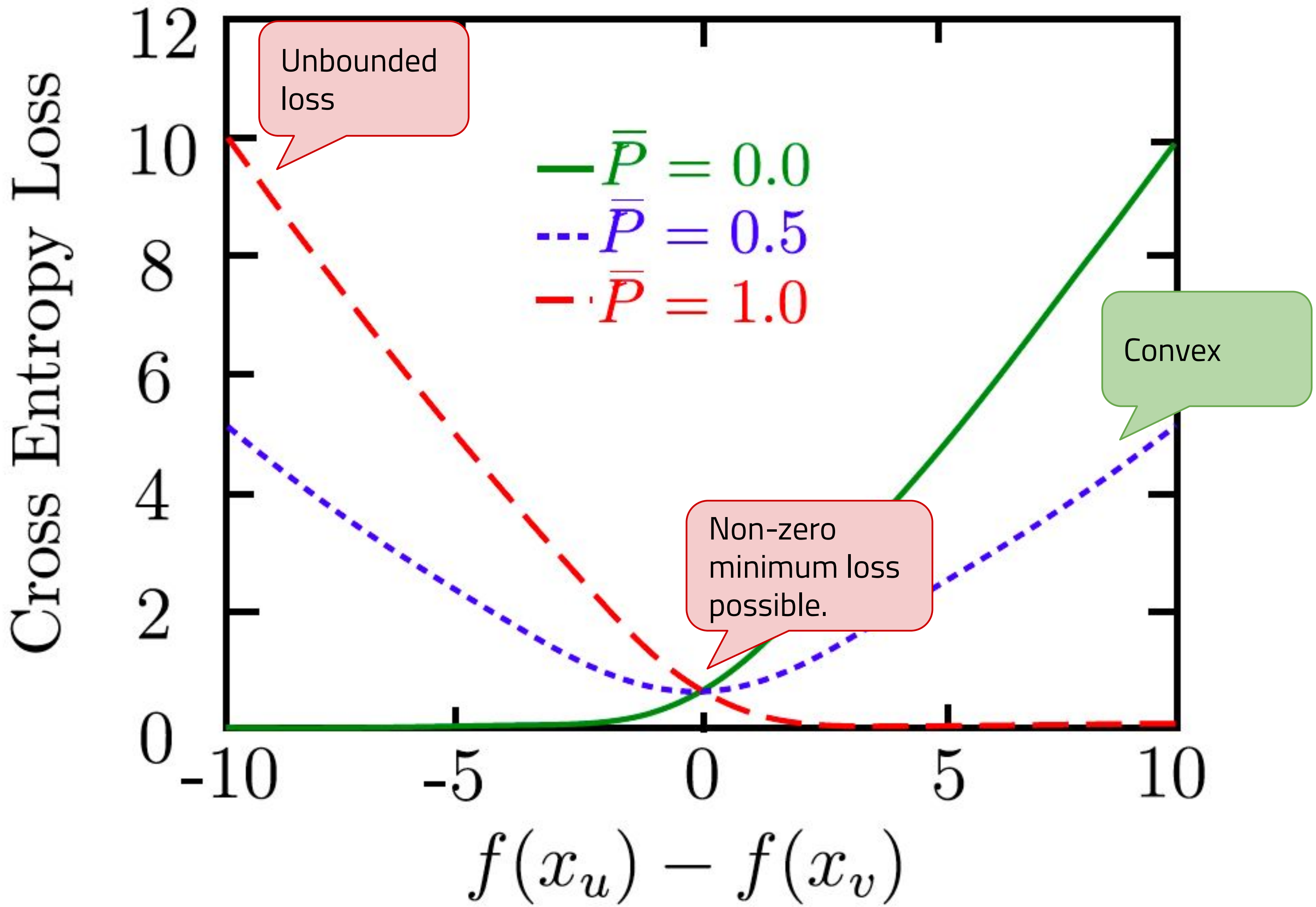


Image: https://www.nowpublishers.com/article/Details/INR-016

FRank: Fidelity loss

$$L(f; x_u, x_v, y_{u,v}) = 1 - \sqrt{\bar{P}_{u,v} P_{u,v}(f)} - \sqrt{(1 - \bar{P}_{u,v})(1 - P_{u,v}(f))}$$

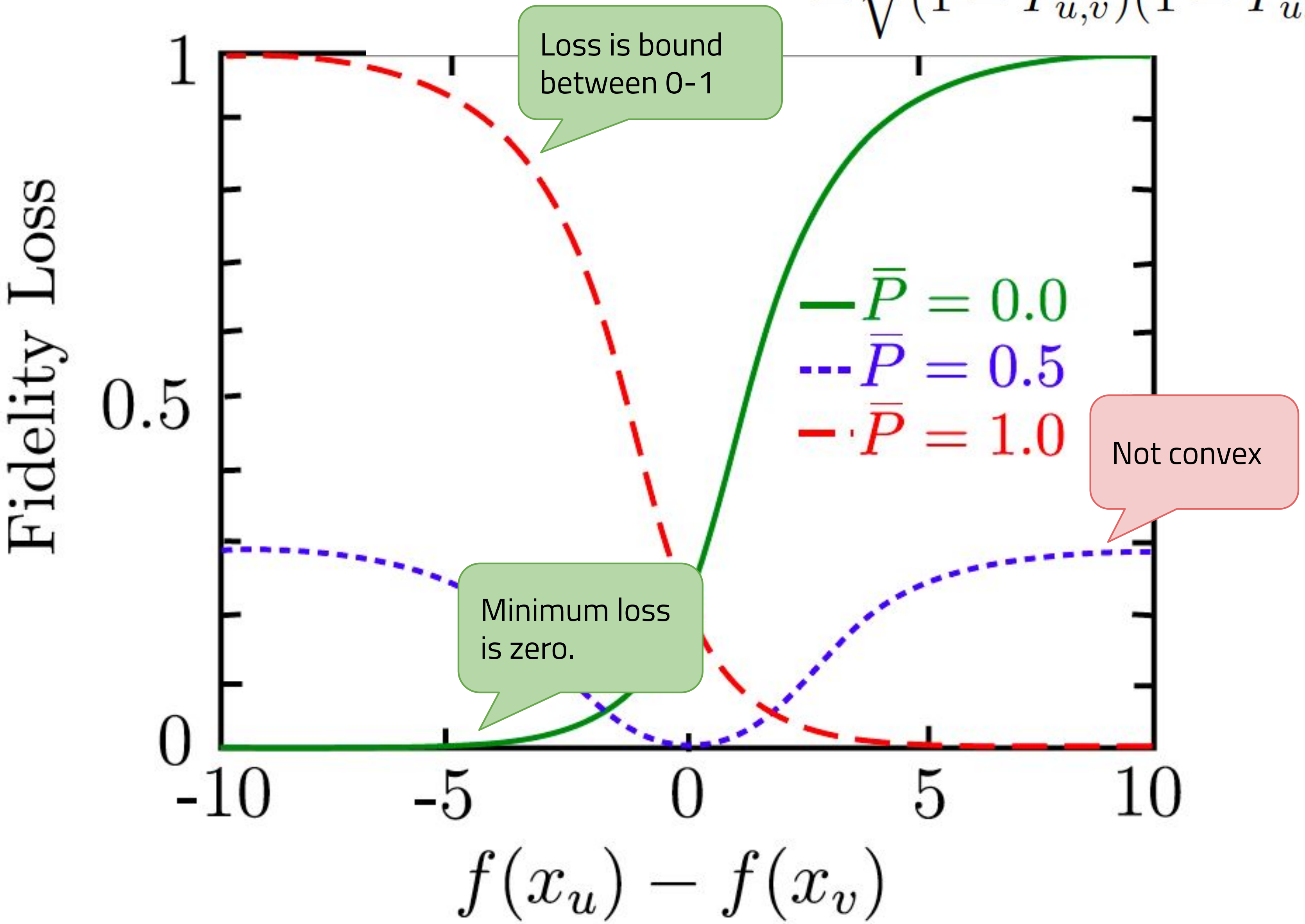
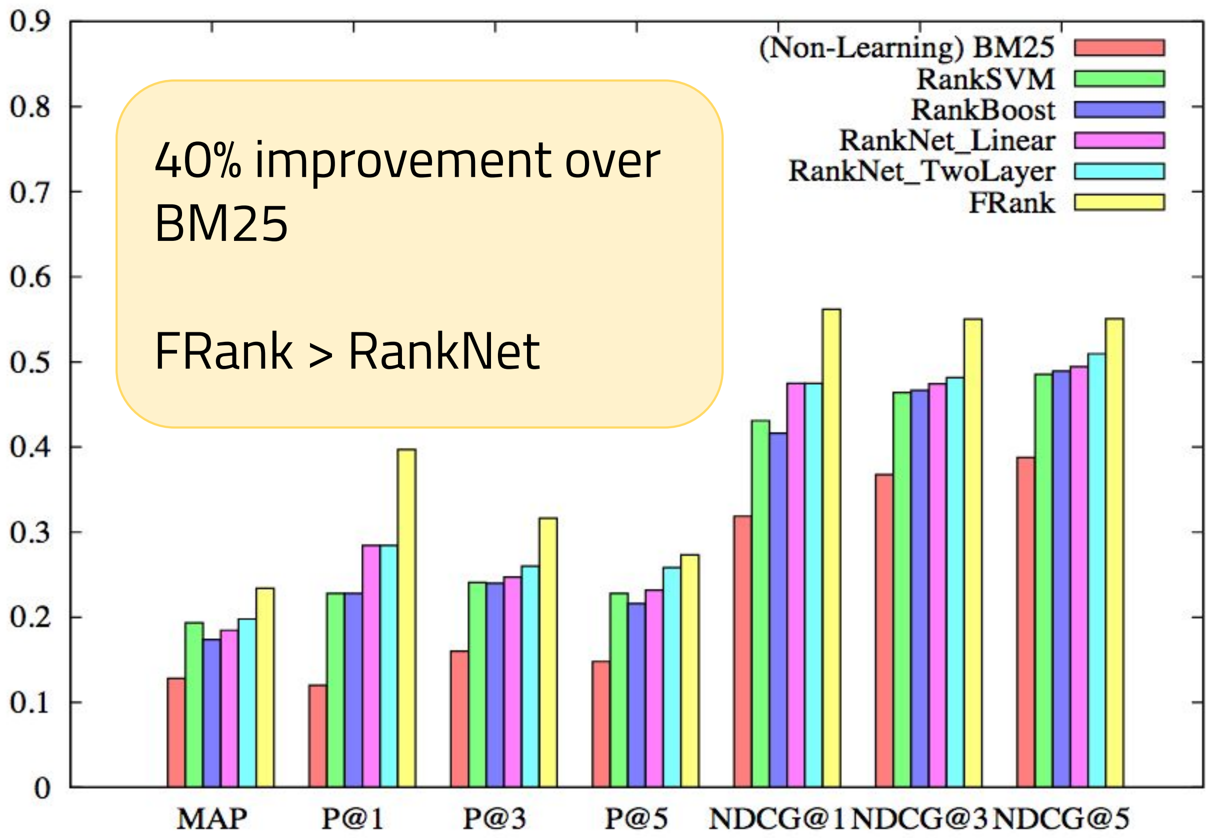


Image: https://www.nowpublishers.com/article/Details/INR-016

Experimentally: RankNet vs. FRank



TREC topic distillation aims at finding key resources which are high-quality pages for certain topics.

Corpus: 1M `.gov` pages, 14 features per document

50 topics (between 1 and 86 relevant docs per topic)

4-fold cross validation

Is training of the pairwise approach slower/faster than the pointwise one?

Image: https://dl.acm.org/citation.cfm?id=1277741.1277808

Document pair issue

Document pairs only make it into the training set if their relevance degrees differ

Worst case: number of pairs quadratic in the document number

Queries differ widely in the number of pairs they generate

Loss will be dominated by queries with many pairs (requires query-level normalization)

Highly skewed distribution of pairs/queries (based on a dataset from a web search engine)

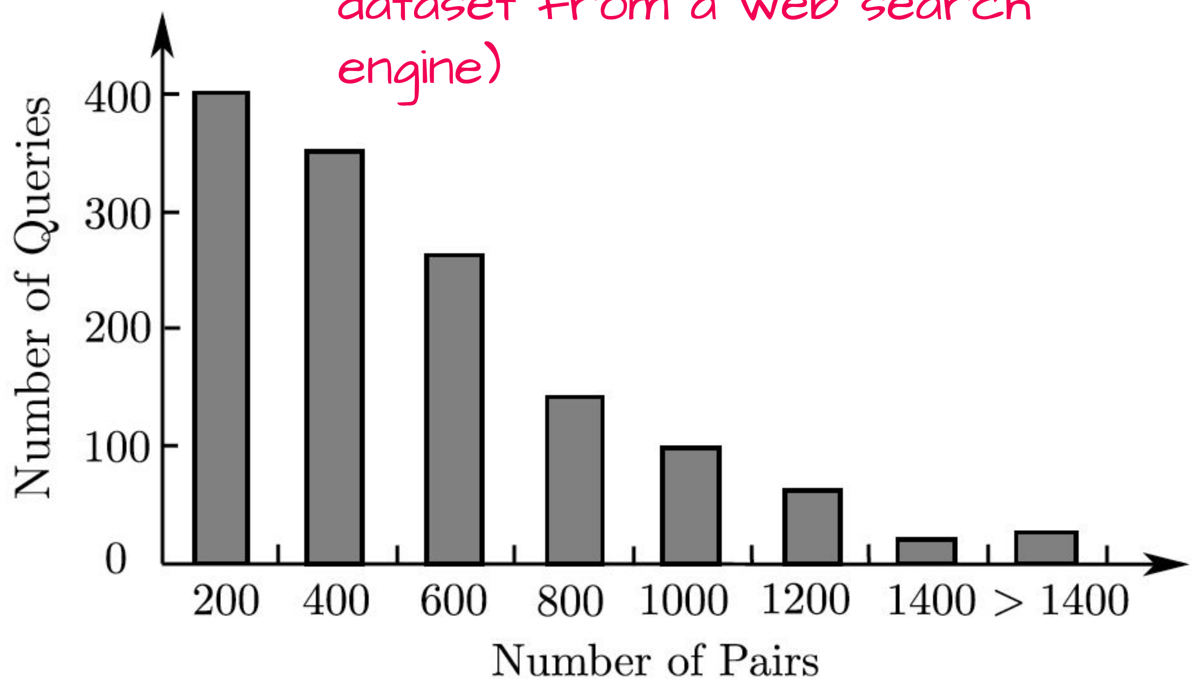


Image: https://www.nowpublishers.com/article/Details/INR-016

Beyond RankNet ...

>> LambdaRank

- Insight: neural net training requires only the *gradients* of the cost function
- Heuristic rules on how the cost changes if document rankings are swapped

>> LambdaMART

- Gradient boosted decision trees

Burges, Christopher JC. "From ranknet to lambdarank to lambdamart: An overview." Learning 11.23-581 (2010): 81.

Listwise approach

Listwise

1) Optimize a continuous & differentiable approximation. of an IR metric.
2) Optimize a continuous & differentiable bound of an IR metric.
3) Choose optimization approach that can handle complex objectives.

Direct optimization

Output space contains relevance degrees of all docs associated with q.

Loss function **optimizes an IR metric.** Not easy as MAP, NDCG, ... are non-continuous & non-differentiable.

Most optimization techniques are designed for continuous and differentiable functions.

Examples: SoftRank, AdaRank

Permutation-based

The output space contains the permutation of the documents associated with q.

The loss function measures the difference between the permutation given by the hypothesis and the ground truth permutation.

Examples: ListNet, ListMLE

ListNet

Required: a loss function that considers the document list

Idea: define two probability distributions, one on the hypothesized and one on the reference ranking. Use a metric that **compares the two probability distributions** as loss function.

Given scoring function f

and document relevance scores $\mathbf{s} = \{s_j\}_{j=1}^{m}$, *where* $s_j = f(x_j)$,

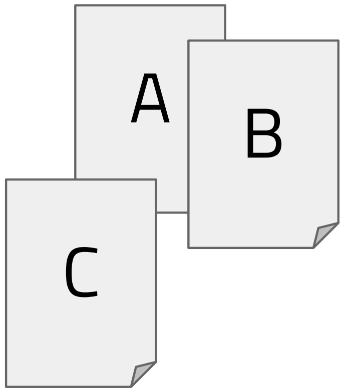
define a prob. for each possible permutation π *of the documents:*

$$\underbrace{P(\pi|\mathbf{s})}_{\text{permutation probability}} = \prod_{j=1}^{m} \underbrace{\frac{\varphi(s_{\pi^{-1}(j)})}{\sum_{u=j}^{m} \varphi(s_{\pi^{-1}(u)})}}_{\text{conditional probability}}$$

doc. ranked at jth position in the permutation

transformation function (e.g. exponential)

ListNet permutation example



Given 3 documents for query q, what is the probability of the ranking permutation A-B-C?

$$P_\pi = P_1 \times P_2 \times P_3$$

$$P_1 = \frac{\varphi(s_A)}{\varphi(s_A) + \varphi(s_B) + \varphi(s_C)}.$$

Probability of doc A being ranked at the top. Determined by comparing A's score to B's and C's scores.

$$P_2 = \frac{\varphi(s_B)}{\varphi(s_B) + \varphi(s_C)}$$

Probability of B being ranked at position 2, given that A has been ranked already.

$$P_3 = 1$$

Only C is left.

ListNet

ListNet defines the permutation probability distribution based on the scores given by the scoring function.

The **reference permutation probability distribution** is based on the ground truth labels.

KL divergence between both distributions to define the listwise ranking loss:

$$L(f;\mathbf{x},\pi_y) = D(P(\pi \mid \varphi(f(w,\mathbf{x}))) \| P_y(\pi))$$

Shallow neural network learns scoring function f; gradient descent as optimization alg

Practical issue: training over all possible permutations of a list is impractical ($m!$ permutations of size m)

Benchmarks & practice

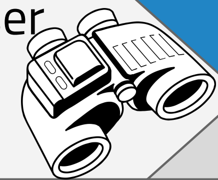
	Queries	Doc.	Rel.	Feat.	Year
LETOR 3.0 – Gov	575	568 k	2	64	2008
LETOR 3.0 – Ohsumed	106	16 k	3	45	2008
LETOR 4.0	2,476	85 k	3	46	2009
Yandex	20,267	213 k	5	245	2009
Yahoo!	36,251	883 k	5	700	2010
Microsoft	31,531	3,771 k	5	136	2010

Burges et al. (2011) used a linear combination of 12 ranking models, 8 of which were LambdaMART (Burges, 2010) boosted tree models, 2 of which were LambdaRank neural nets, and 2 of which were logistic regression models. While LambdaRank was originally instantiated using neural nets, LambdaMART implements the same ideas using the boosted-tree style MART algorithm, which itself may be viewed as a gradient descent algorithm. Four of the LambdaMART rankers (and one of the nets) were trained using the ERR measure, and four (and the other net) were trained using NDCG. Extended training sets were also generated by randomly deleting feature vectors for each query.

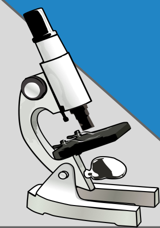
Chapelle, Olivier, and Yi Chang. "Yahoo! learning to rank challenge overview." Proceedings of the learning to rank challenge. 2011.

Implicit feedback

Incorporating user behaviour information

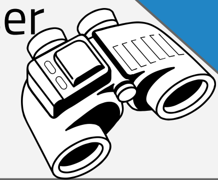


Evaluating the accuracy of implicit feedback (clicks)



Implicit feedback

Incorporating user behaviour information

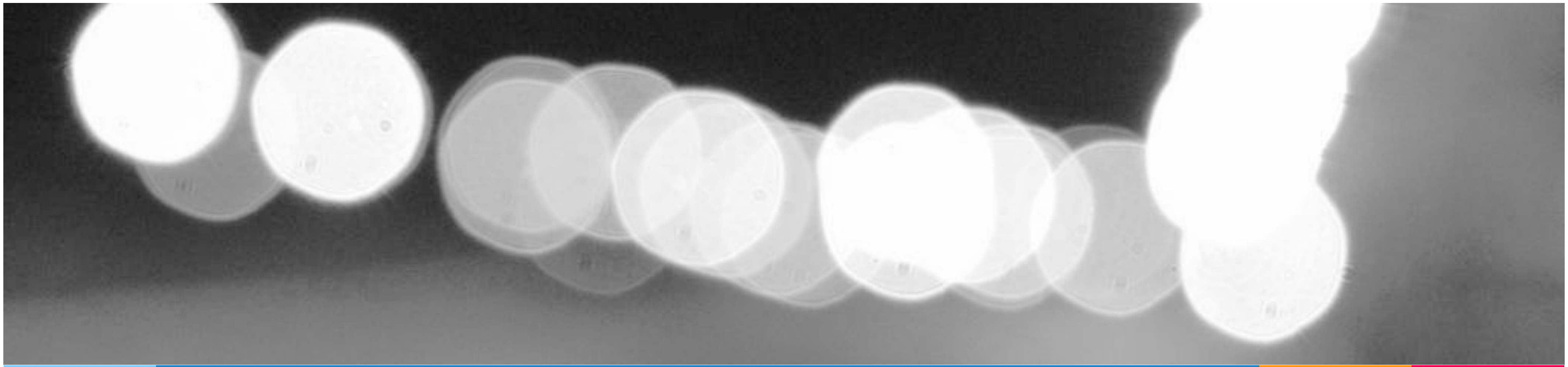


Explicit vs. implicit

Learning to rank, BM25, LM ...

- Need **training data** to effectively learn the models' (hyper)parameters
- Often explicit relevance judgments are used

Explicit qrels are extremely **expensive** to accumulate (can become outdated quickly for dynamic collections)



TREC-8 numbers
(ran in 1999)

50

Topics (ad hoc task)

86,830

Pooled documents (k=100)

129

Systems

723

Assessor hours

At $20 an hour, that amounts to $14,460.

And thus, We are still using the TREC-8 corpus to this day for experiments!

Potential source: clickthrough data

RQ: How effective is **implicit feedback** in practice (i.e. in a large-scale operational environment)?

- Web search engines use thousands of features and are heavily tuned
- Tuning is a continuous process

RQ: How can implicit feedback be **combined** with the existing ranking produced by the search system?

Insight: Instead of treating a user as a reliable "expert", aggregate information from multiple, unreliable search session traces

https://dl.acm.org/citation.cfm?id=1148177

Clickthrough data as independent evidence

1) Retrieve an *initial ranking* (low cost)
2) Assign an expected relevance/user satisfaction score based on previous interactions
3) Merge the **rank orders** of the original and implicit feedback (IF) based ranking
4) Order results by the merge score

Empirically found to work well

original rank of doc d

$$S_M(d, I_d, O_d, w_I) = \begin{cases} \frac{1}{O_d+1}, & \textit{if no IF exists} \\ w_I \frac{1}{I_d+1} + \frac{1}{O_d+1}, & \textit{otherwise} \end{cases}$$

implicit rank of doc d

influence of implicit feedback

If the influence of IF is extremely high, clicked results are simply ranked above unclicked results.

https://dl.acm.org/citation.cfm?id=1148177

Clickthrough data in L2R

1) Derive a set of features from implicit feedback
2) At runtime, the search engine needs to fetch the implicit feedback features associated with each `(query,URL)` pair

L2R needs to be robust to missing values: *long tail* issue

Here: **RankNet**

- Neural net based tuning algorithm that optimizes feature weights to best match explicitly provided **pairwise** user preferences
- Has both train- and run-time efficiency
- Aggregate `(query,URL)` pair features across all instances in the session logs

https://dl.acm.org/citation.cfm?id=1148177

How does a search engine get this data?

Features

Different types of user action features

Directly observed vs. derived features (derivations)

Browsing behaviour after the result has been clicked

Snippet based features are included as users often determine relevance based on snippet information

Clickthrough features	
Position	Position of the URL in Current ranking
ClickFrequency	Number of clicks for this query, URL pair
ClickProbability	Probability of a click for this query and URL
ClickDeviation	Deviation from expected click probability
IsNextClicked	1 if clicked on next position, 0 otherwise
IsPreviousClicked	1 if clicked on previous position, 0 otherwise
IsClickAbove	1 if there is a click above, 0 otherwise
IsClickBelow	1 if there is click below, 0 otherwise
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, no 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 average dwell time on page
CumulativeDeviation	Deviation from average cumulative dwell time
DomainDeviation	Deviation from average dwell time on domain
Query-text features	
TitleOverlap	Words shared between query and title
SummaryOverlap	Words shared between query and snippet
QueryURLOverlap	Words shared between query and URL
QueryDomainOverlap	Words shared between query and URL domain
QueryLength	Number of tokens in query
QueryNextOverlap	[illegible]

https://dl.acm.org/citation.cfm?id=1148177

Evaluation

How should we split the set into training/val/test set? By time? By query? By user? By ...?

Random sample of queries from a Microsoft query log with associated results and traces of user actions

8 weeks of user interactions with 1.2M unique queries (sufficient interactions for 50% of queries) and 12M interactions

On average, 30 results judged per query by human assessors on a **six point scale** (83K results judged)

	MAP
BM25F (content + link-based info)	0.184
RankNet	0.215
ReRanking	
BM25F + Click-through statistics only	0.215
BM25F + Implicit feedback	0.222
Integrated as features	
RankNet + Implicit feedback	<u>0.248</u>

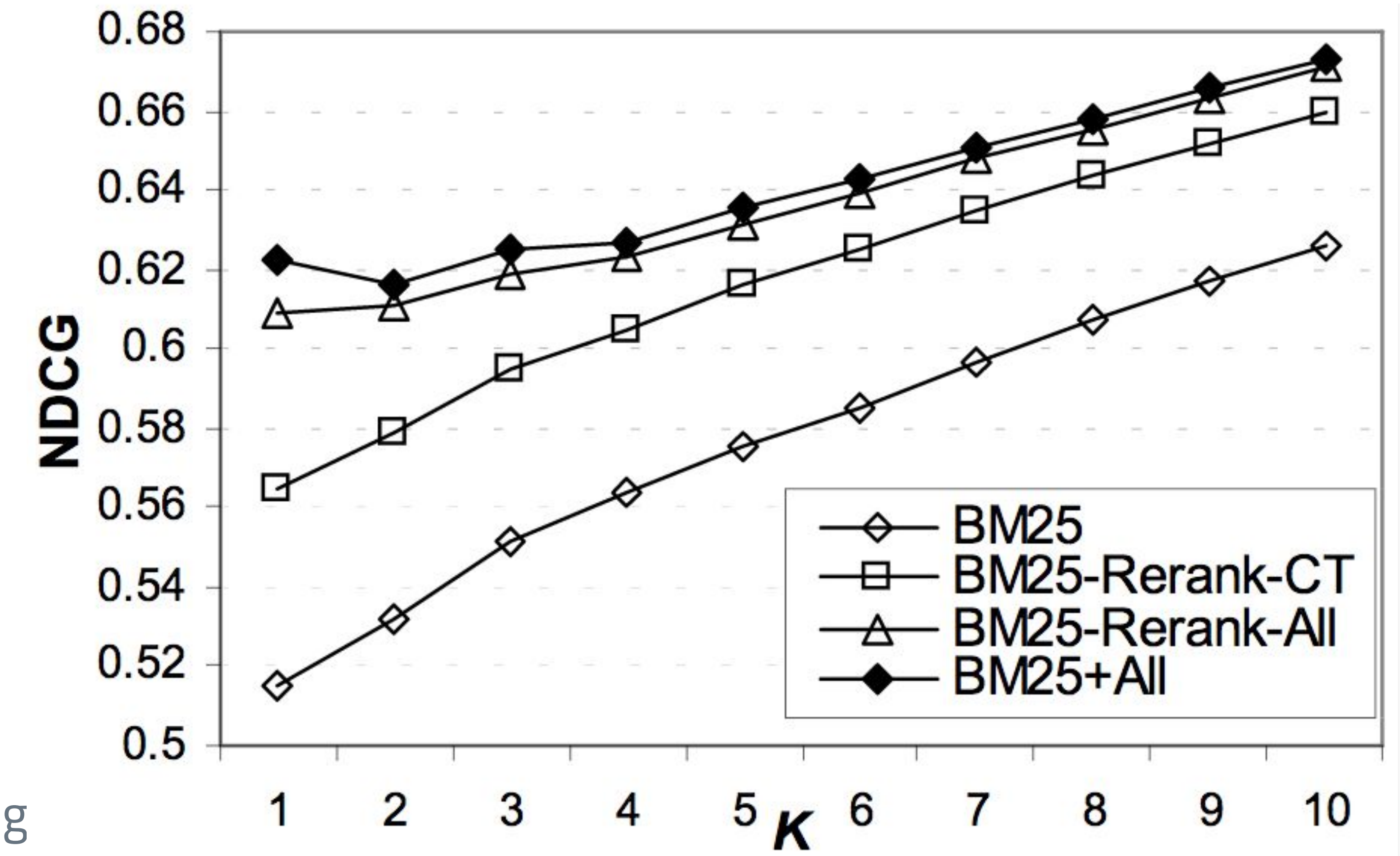
Results

`BM25F`: content-based (fields) and query-independent link-based information (PageRank, URL depth, etc.); does *not* make use of implicit/explicit feedback

`BM25F-RerankCT`: reranking based on clickthrough statistics (weight $w=1000$)

`BM25F-RerankAll`: RankNet-based reranking with all behavioural features

`BM25F+All`: RankNet-based ranking on BM25F features+IF



https://dl.acm.org/citation.cfm?id=1148177

Results ctd.

`RankNet (RN)`: hundreds of features of a Web search engine; based on explicit judgments

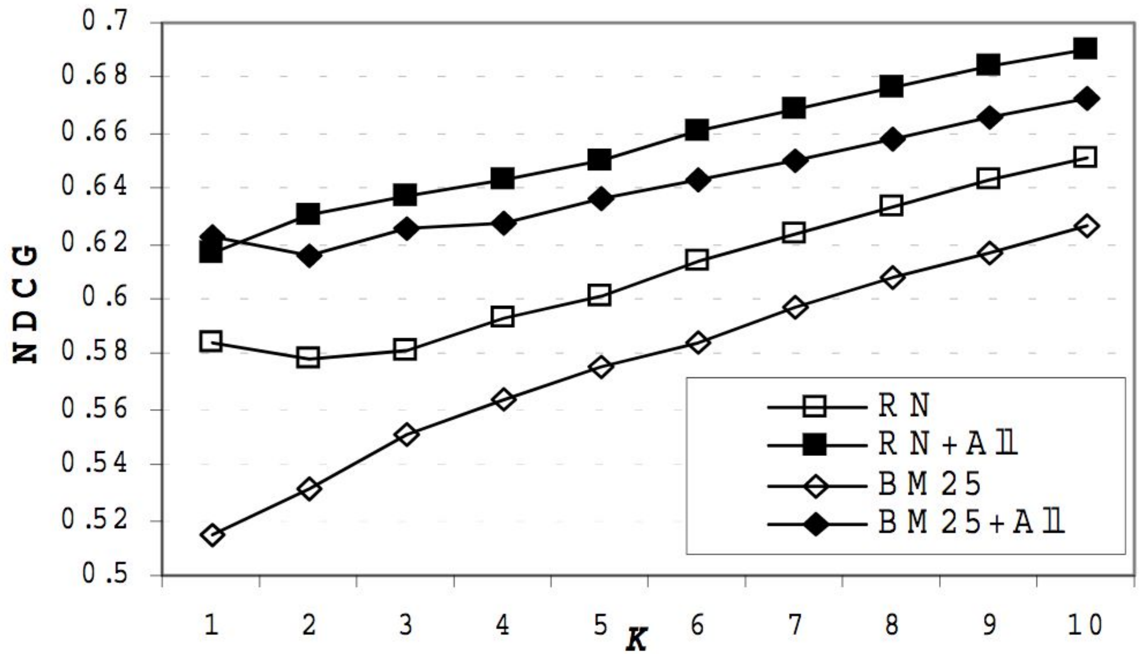
`RankNet+All`: including IF features

`BM25F`: content-based (fields) and query-independent link-based information (PageRank, URL depth, etc.)

`BM25F+All`: train RankNet over the feature set of BM25F and IF

	MAP
BM25F	0.184
BM25F-Rerank-CT	0.215
BM25F-RerankImplicit	0.218
BM25F+Implicit	**0.222**
RN	0.215
RN+All	**0.248**

IF can replace hundreds of features



https://dl.acm.org/citation.cfm?id=1148177

Implicit feedback

Evaluating the
accuracy of implicit
feedback (clicks)

https://dl.acm.org/citation.cfm?id=1229181

Generating training data from clicks

RQ: can **training examples** (qrels) be generated automatically from clickthrough data?

Advantages: cost effective, large quantities, without burdening the user (no relevance feedback), up-to-date

Disadvantages: more difficult to interpret & noisy

Controlled study!

User study investigating users' interaction with the SERP

- Relationship between click behaviour (=implicit feedback) and explicit relevance judgments
- Eye-tracking experiment provides insights into users' subconscious behaviour

https://dl.acm.org/citation.cfm?id=1229181

Generating training data from clicks

Important to know what results a user actually views

- Implicit relevance judgments need to be considered in this context (a result not viewed cannot be considered non-relevant)

Early work assumed that each click represents an endorsement of the result (i.e. a click = a positive relevance judgment)

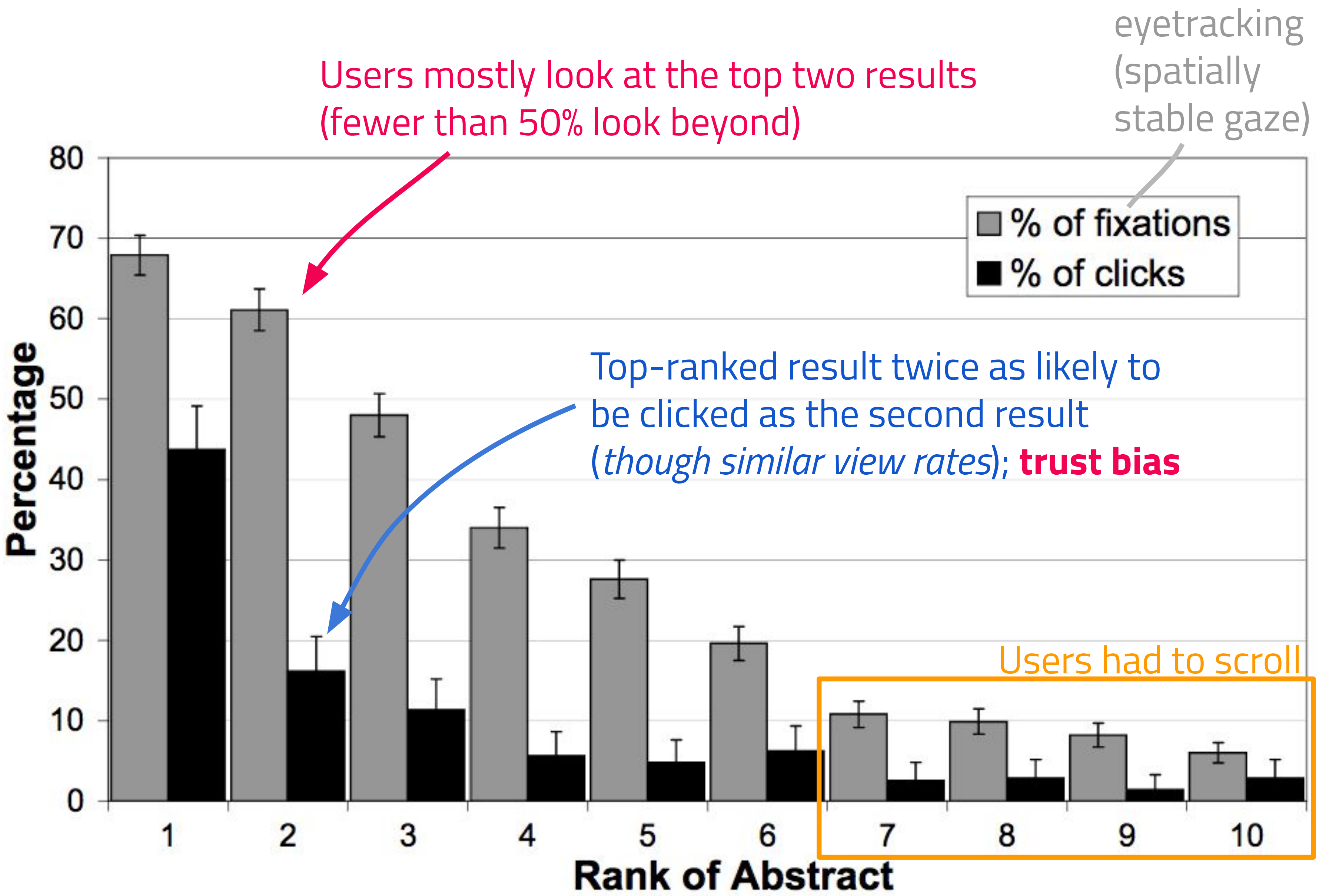
User study with 3 experimental conditions (10 topics)

- **Normal** (original SERP)
- **Swapped** (top two results swapped on the SERP)
- **Reversed** (top-10 results on the SERP in reverse order)

Explicit relevance judgments collected as control

https://dl.acm.org/citation.cfm?id=1229181

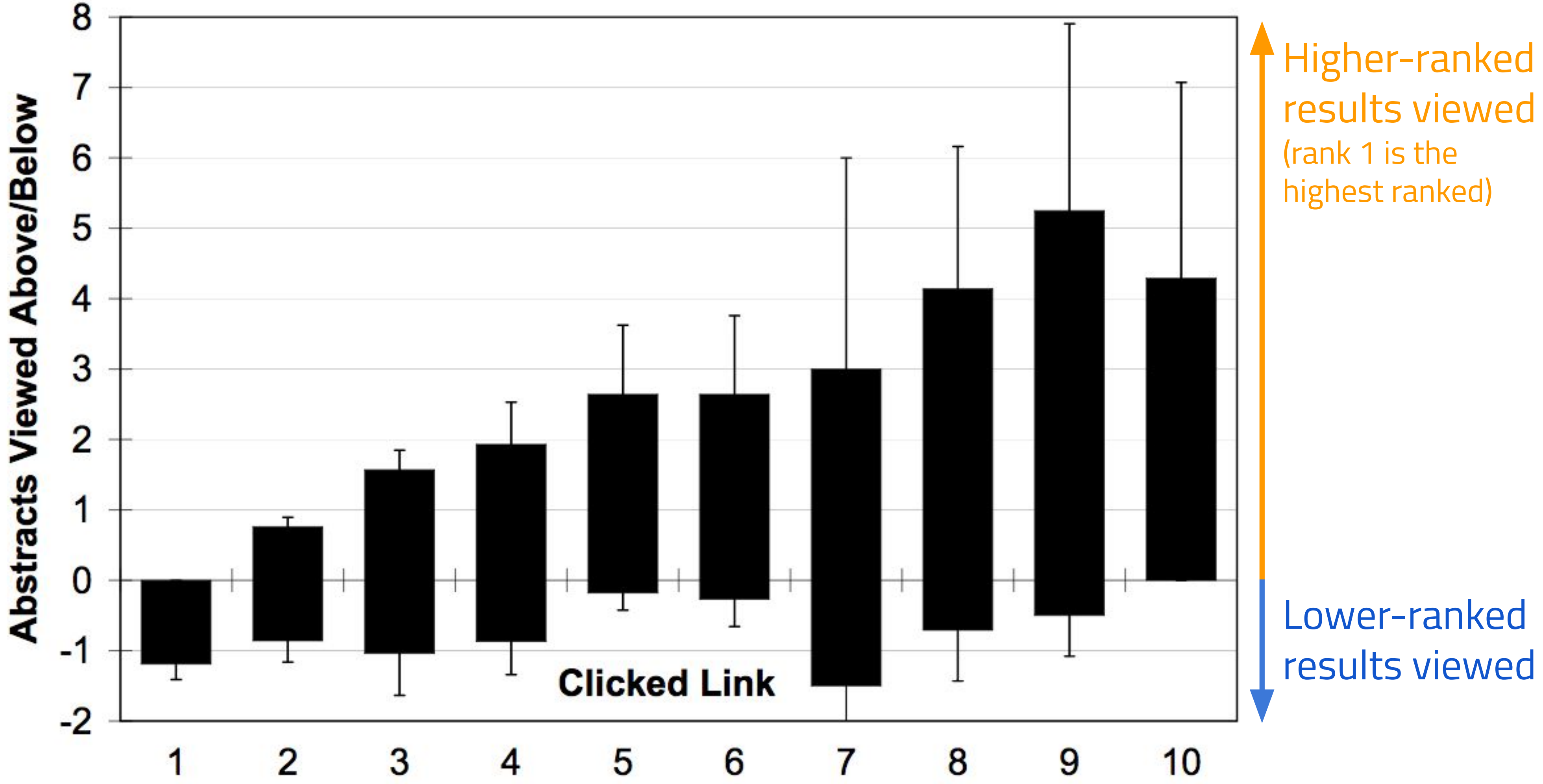
Result ranks vs. clicks & views



https://dl.acm.org/citation.cfm?id=1229181

Scanning the SERP

Which snippets do users evaluate before clicking?



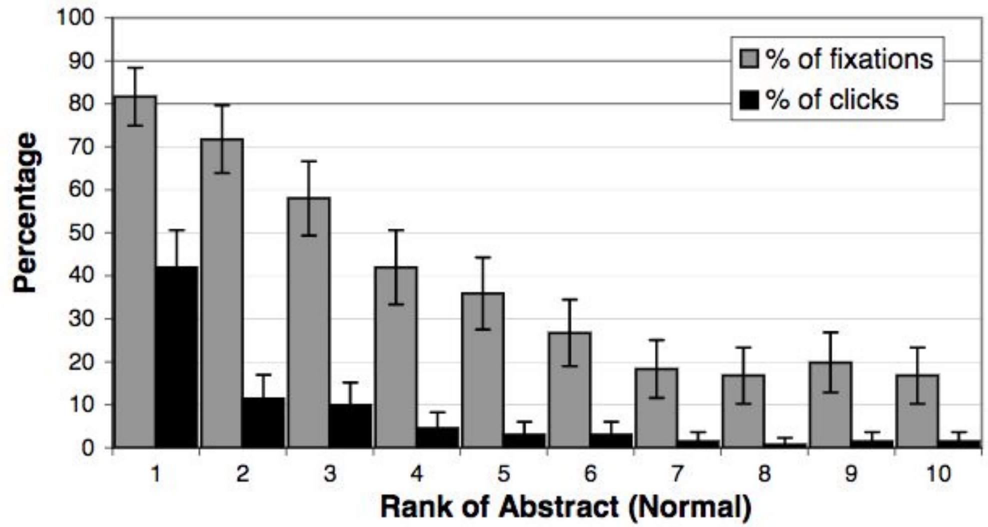
https://dl.acm.org/citation.cfm?id=1229181

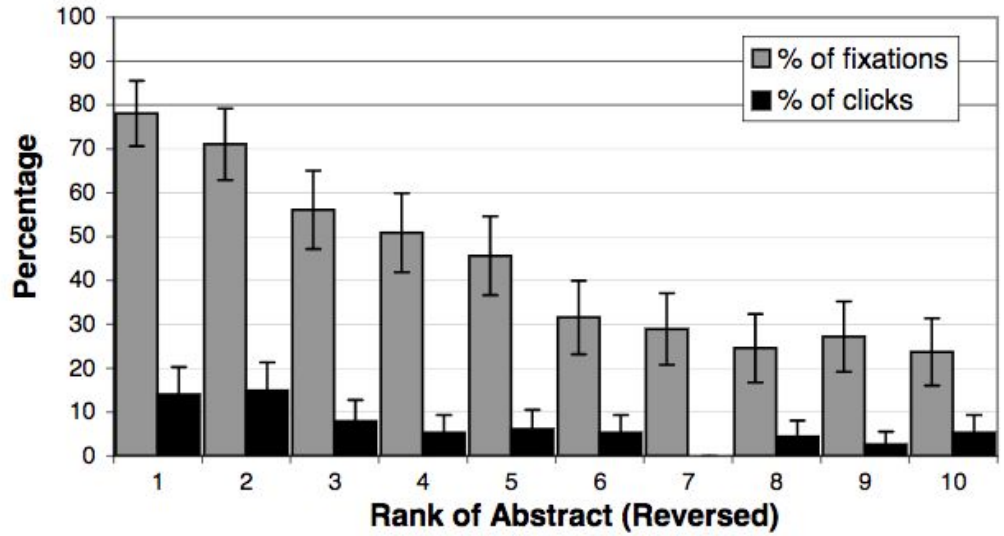
Relevance vs. user decisions

So far: clicks considered independent of relevance

Reverse condition (degraded ranking)

- Users view lower ranks more frequently
- Users are less likely to click on result 1 (av. click rank 4 vs. 2.7 in normal)
- **Quality-of-context bias**: clicks are less relevant on average compared to the normal condition (clicks dependent on overall quality of the system)



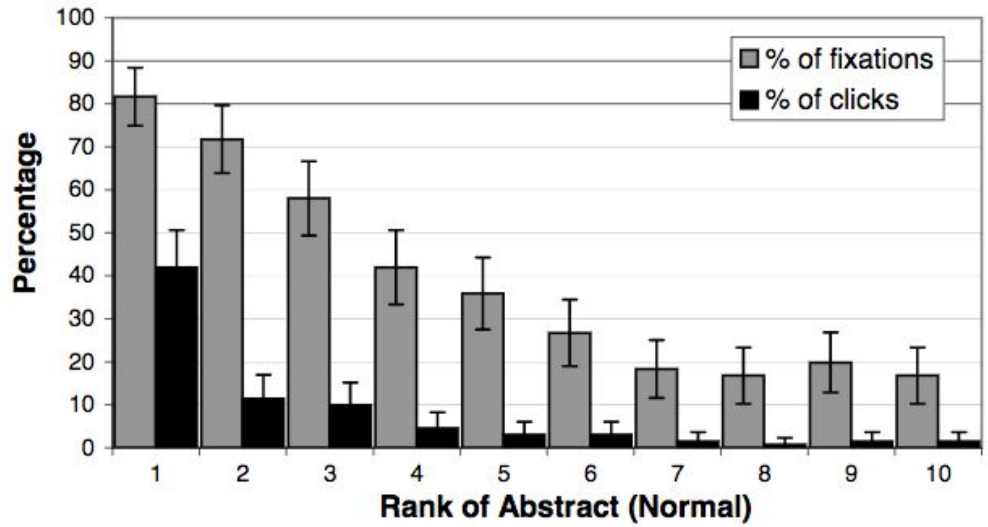


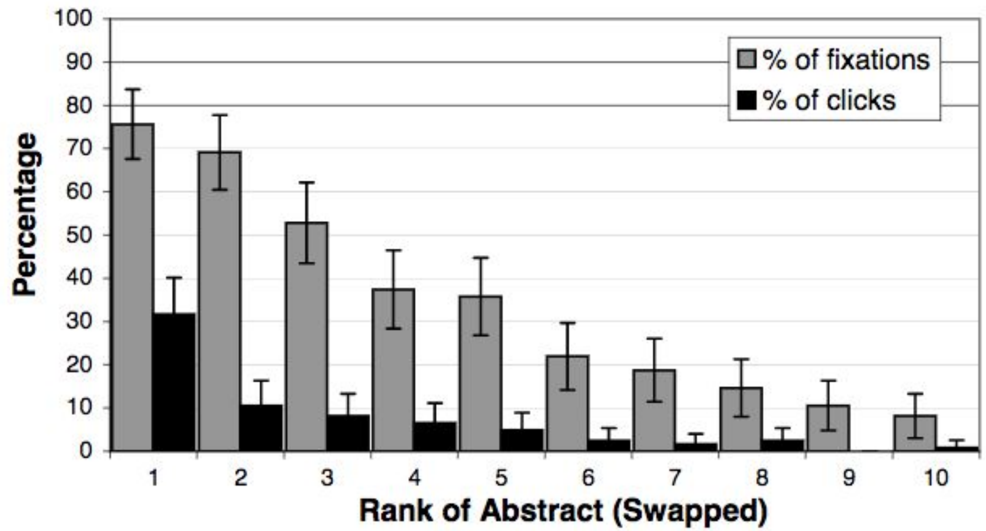
https://dl.acm.org/citation.cfm?id=1229181

Relevance vs. user decisions

Swapped condition

- Trust bias (Google must be right)
- Users *are* influenced by result order
- Decision to click influenced by result position





https://dl.acm.org/citation.cfm?id=1229181

Thus …

Interpreting clicks as <u>absolute</u> relevance judgments is likely to fail.
Accurate interpretations need to take a user's trust and the system quality into account.

Clicks can be seen as <u>preference</u> statements.
We exploit the fact that some results were not clicked.

$$l_1^* \; l_2 \; l_3^* \; l_4 \; l_5^* \; l_6 \; l_7; \quad {}^*{=}click$$

rel($l3$)>rel($l2$)   rel($l5$)>rel($l4$), rel($l5$)>rel($l2$)

https://dl.acm.org/citation.cfm?id=1229181

Extracting preference feedback

$$l_1^* \; l_2 \; l_3^* \; l_4 \; l_5^* \; l_6 \; l_7; \quad *{=}click$$

rel(*l3*)>rel(*l2*) rel(*l5*)>rel(*l4*), rel(*l5*)>rel(*l2*)

Partial rankings: a relevance based ranking should return *l3* ahead of *l2* and *l5* ahead of *l2* and *l4*.

1	$C=\{2,5,7\}$
2	$rel(l_2)>rel(l_1)$
3	$rel(l_5)>rel(l_1)$
4	$rel(l_5)>rel(l_3)$
5	$rel(l_5)>rel(l_4)$
6	$rel(l_7)>rel(l_1)$
7	$rel(l_7)>rel(l_3)$
8	$rel(l_7)>rel(l_4)$
	$rel(l_7)>rel(l_6)$

Strategy: Click > Skip-Above

Takes trust bias and quality-of-context into account

For a ranking $(l_1, l_2, \ldots)$ *and a set* C *containing the ranks of the clicked results, extract a preference example* $rel(l_i) > rel(l_j)$ *for all pairs* $1 \leq j < i$, *with* $i \in C$ *and* $j \notin C$

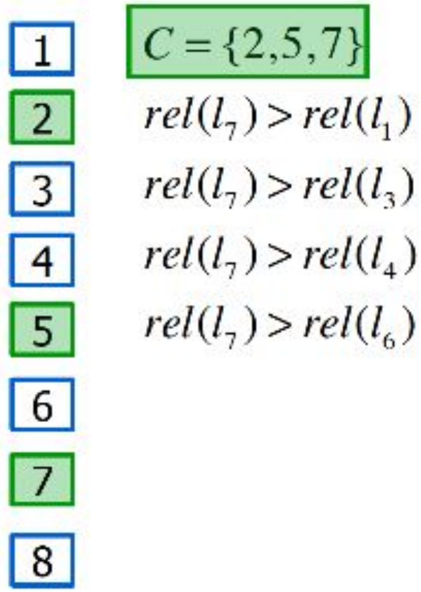
https://dl.acm.org/citation.cfm?id=1229181

Extracting preference feedback

$$l_1^* \; l_2 \; l_3^* \; l_4 \; l_5^* \; l_6 \; l_7; \quad * = click$$

rel($l3$)>rel($l2$)   rel($l5$)>rel($l4$), rel($l5$)>rel($l2$)

Partial rankings: a relevance based ranking should return $l3$ ahead of $l2$ and $l5$ ahead of $l2$ and $l4$.



Strategy: Last Click > Skip-Above

Intuition: later clicks are more informative than earlier ones

For a ranking $(l_1, l_2, \ldots)$ and a set C containing the ranks of the clicked results, let $i \in C$ be the rank of the last click. Extract a preference example $rel(l_i) > rel(l_j)$ for all pairs $1 \le j < i$, and $j \notin C$.

Other strategies exist.

https://dl.acm.org/citation.cfm?id=1229181

Accuracy of extracted feedback

Implicit feedback (clicks) ⟷ Explicit feedback (relevance judgm.)

Click > Skip Above yields 81% correct preferences

Last Click > Skip Above yields 83% correct preferences

Random baseline: 50% accuracy

Inter-annotator agreement (human assessors): 90% accuracy (upper bound)

https://dl.acm.org/citation.cfm?id=1229181

Query chains

reformulation after zero-click SERP

$$\text{oed} \Rightarrow l_1\ l_2\ l_3\ l_4\ l_5\ l_6\ l_7$$

$$\text{oxford english dictionary} \Rightarrow l'_1\ l'^{*}_2\ l'_3\ l'_4\ l'^{*}_5\ l'_6\ l'_7$$

May be relevant to "oed"

Generated preferences: comparison between the results from the same query (within-query preferences)

Too restrictive:

- Strategies only produce preferences between the top few results shown to the user
- Typically users run query chains (reformulations)

Goal: generate accurate relative preference judgments between results from **different queries** within a chain of query reformulations (same information need)

Query chains

Strategy: Click > Skip Earlier QC

For a ranking $(l_1, l_2, \ldots.)$ followed by ranking $(l'_1, l'_2, \ldots.)$ (not necessarily immediately) within the same query chain and sets C and C' containing the ranks of the clicked on results, extract a preference example $rel(l'_i) > rel(l_j)$ for all pairs $i \in C'$ and $j < \max(C)$, with $j \notin C$.

$$q_1 : l_{11}\ l_{12}\ l_{13}\ l_{14}\ l_{15}\ l_{16}\ l_{17}$$
$$q_2 : l^*_{21}\ l_{22}\ l^*_{23}\ l_{24}\ l^*_{25}\ l_{26}\ l_{27}$$
$$q_3 : l_{31}\ l^*_{32}\ l_{33}\ l_{34}\ l_{35}\ l_{36}\ l_{37}$$
$$q_4 : l^*_{41}\ l_{42}\ l_{43}\ l_{44}\ l_{45}\ l_{46}\ l_{47}$$

$$rel(l_{32}) > rel(_{22})$$
$$rel(l_{32}) > rel(l_{24})$$
$$rel(l_{41}) > rel(l_{22})$$
$$rel(l_{41}) > rel(l_{24})$$
$$rel(l_{41}) > rel(l_{31})$$

Accuracy depends on the presentation order:
85% (normal) vs. 55% (reversed)

Summary L2R

With sufficiently many features, L2R outperforms BM25 and similar non-ML baselines

It is not sufficient to deploy standard classifiers; need to be adapted towards ranking (pairwise, listwise)

Listwise vs. pairwise: the former makes most theoretical sense, the latter is empirically more robust and efficient

L2R is often neural-network based (shallow, not deep)

Best results are achieved in ensembles