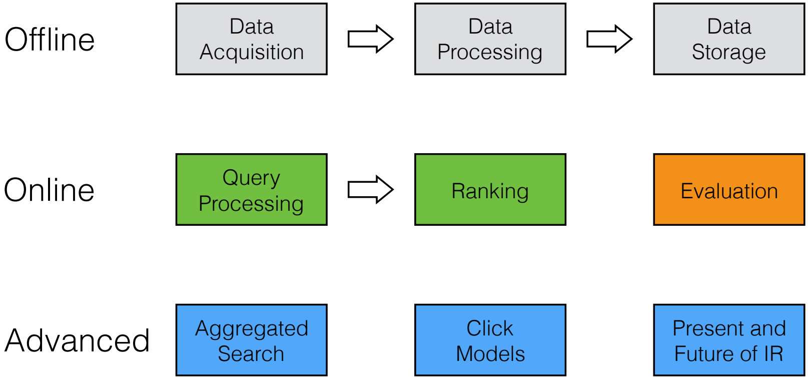
Information Retrieval

Learning to Rank

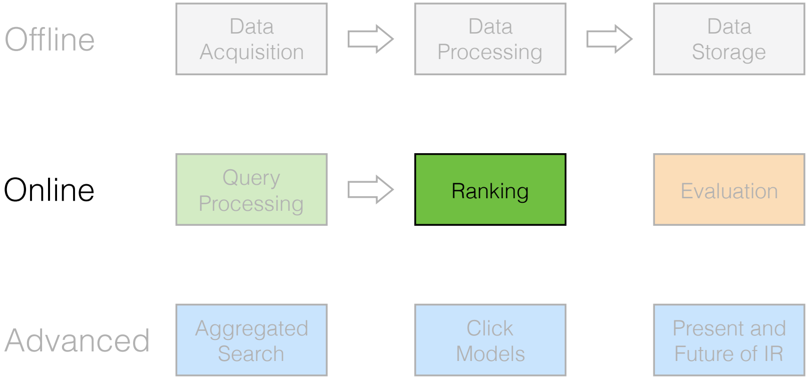
Ilya Markov
i.markov@uva.nl

University of Amsterdam

Course overview



This lecture



Outline

IR conferences

- ACM Conference on Research and Development in Information Retrieval (SIGIR)
- ACM Conference on Information Knowledge and Management (CIKM)
- ACM Conference on Web Search and Data Mining (WSDM)
- European Conference on Information Retrieval (ECIR)

IR journals

- ACM Transactions o Information Systems (TOIS)
- Information Retrieval Journal (IRJ)
- Information Processing and Management (IPM)

Surveys

- Foundations and Trends in Information Retrieval (FnTIR)
- Synthesis Lectures on Information Concepts, Retrieval, and Services by Morgan&Claypool Publishers

SIGIR 2016

- Evaluation
- Efficiency
- Retrieval models, learning-to-rank, web search
- Users, user needs, search behavior
- Novelty and diversity
- Speech, conversation systems, question answering
- Recommendation systems
- Entities and knowledge graphs

WSDM 2016

- Communities, social interaction, social networks
- Search and semantics
- Observing users, leveraging users
- Big data algorithms
- Entities and structure

Ranking methods

1. Content-based
   - Term-based
   - Semantic
2. Link-based (web search)
3. **Learning to rank**

Outline

1. Current trends in IR

2. Learning to rank
   - Machine learning
   - Features
   - LTR approaches
   - Experimental comparison
   - Summary

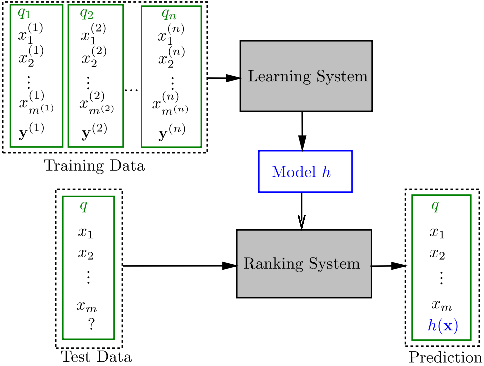
Machine learning

Traditional ML solves a prediction problem (classification or regression) on a single instance at a time.

- Input $\{\mathbf{x}_i\}_{i=1}^{n}$
- Output $\{y_i\}_{i=1}^{n}$
- Learn a model $h(\mathbf{x})$ that optimizes a loss function $L(h(\mathbf{x}), y)$
- For a new instance $\mathbf{x}_{new}$ predict the output $\overline{y} = h(\mathbf{x}_{new})$

Learning to rank

The aim of LTR is to come up with optimal ordering of items, where the relative ordering among the items is more important than the exact score that each item gets.



T.-Y. Liu, "Learning to Rank for Information Retrieval"

Outline

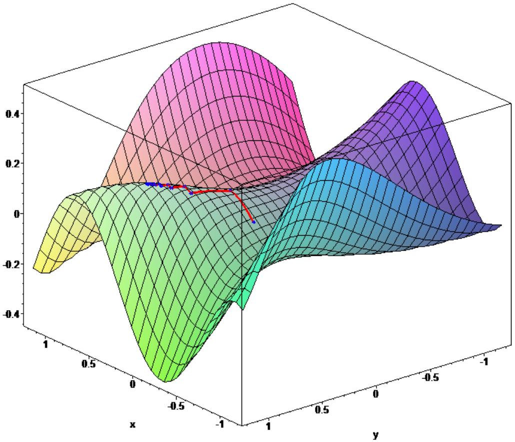
Machine learning

- Input $\{\mathbf{x}_i\}_{i=1}^n$
- Output $\{y_i\}_{i=1}^n$
- Learn a model $h(\mathbf{x})$ that optimizes a loss function $L(h(\mathbf{x}), y)$
- Examples
  - Linear model $h(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i = \sum_{k=1}^{l} w_k x_{ik}$
  - Quadratic loss function $L(h(\mathbf{x}_i), y_i) = \|h(\mathbf{x}_i) - y_i\|^2$
- How to learn the model $h(\mathbf{x})$, i.e., how to estimate its parameters?

Learning the model $h(\mathbf{x})$

- If there is a closed form solution for the parameters of $h(\mathbf{x})$
  1. Compute the derivative of the loss function L with respect to some parameter w_k
  2. Equate this derivative to zero: $\frac{\partial L}{\partial w_k} = 0$
  3. Find the optimal value of the parameter w_k
- If there is no closed form solution, use gradient descent
  1. Compute or approximate the gradient of the loss function $\nabla L = \left[\frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_l}\right]$ using the current values of the parameters
  2. Update the model parameters by taking a small step in the opposite direction of the gradient: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L$

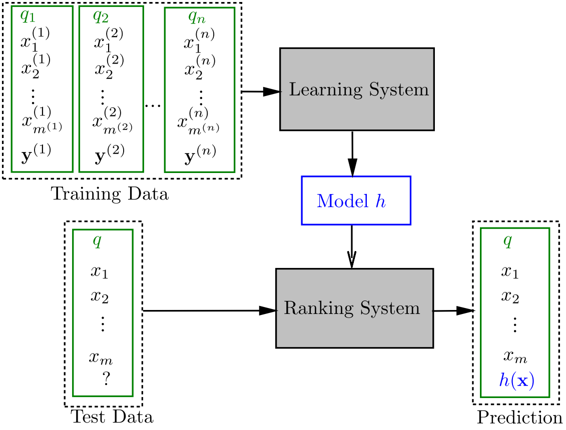
Gradient descent



Picture taken from https://en.wikipedia.org/wiki/Gradient_descent

Outline

Learning to rank



T.-Y. Liu, "Learning to Rank for Information Retrieval"

Query-document representation

- Each query-document pair $(q^{(n)}, x_i)$ is represented as a vector of features $\mathbf{x}_i^{(n)} = [x_{i1}^{(n)}, x_{i2}^{(n)}, \ldots, x_{il}^{(n)}]$
- Features
  - Content-based
  - Link-based
  - User-based

Content-based features

ID	Feature description
1	Term frequency (TF) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	Inverse document frequency (IDF) of body
7	IDF of anchor
8	IDF of title
9	IDF of URL
10	IDF of whole document
11	TF*IDF of body
12	TF*IDF of anchor
13	TF*IDF of title
14	TF*IDF of URL
15	TF*IDF of whole document
16	Document length (DL) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL
25	BM25 of whole document

T.-Y. Liu, "Learning to Rank for Information Retrieval"

Link-based features

ID	Feature description
40	LMIR.JM of whole document
41	Sitemap based term propagation
42	Sitemap based score propagation
43	Hyperlink base score propagation: weighted in-link
44	Hyperlink base score propagation: weighted out-link
45	Hyperlink base score propagation: uniform out-link
46	Hyperlink base feature propagation: weighted in-link
47	Hyperlink base feature propagation: weighted out-link
48	Hyperlink base feature propagation: uniform out-link
49	HITS authority
50	HITS hub
51	PageRank
52	HostRank
53	Topical PageRank
54	Topical HITS authority
55	Topical HITS hub
56	Inlink number
57	Outlink number
58	Number of slash in URL
59	Length of URL

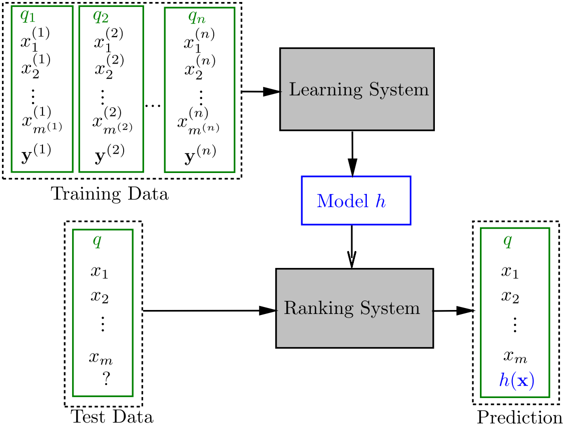
T.-Y. Liu, "Learning to Rank for Information Retrieval"

User-based features

Type of interaction	Online metric
Clicks	Click-through rate for $(q^{(n)}, x_i)$ Avg. click rank for $(q^{(n)}, x_i)$
Time	Avg. dwell time for $(q^{(n)}, x_i)$ Avg. time to first click, when this click is on x_i Avg. time to last click, when this click is on x_i
Queries	Number of reformulations before/after $q^{(n)}$ Number of times $q^{(n)}$ is abandoned

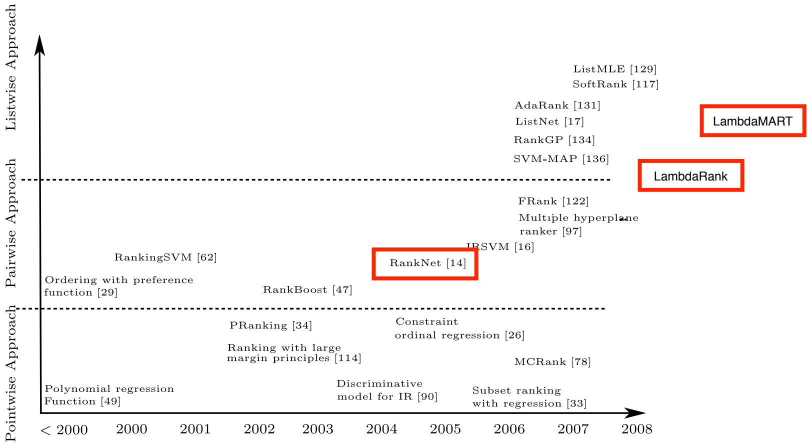
Outline

Learning to rank



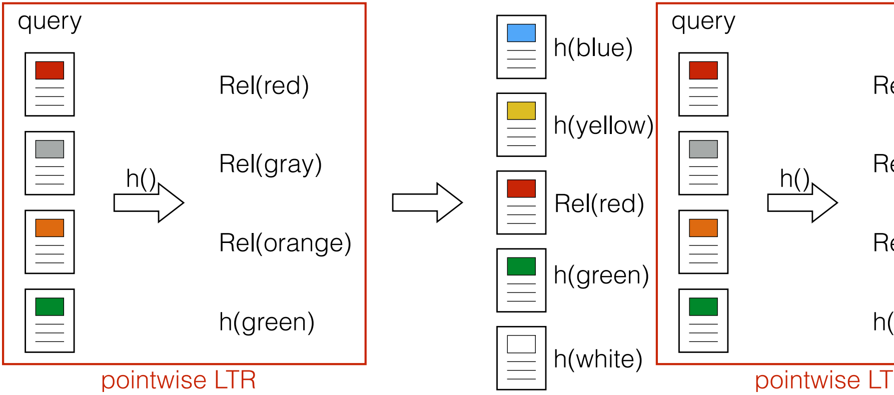
T.-Y. Liu, "Learning to Rank for Information Retrieval"

Learning to rank approaches



T.-Y. Liu, "Learning to Rank for Information Retrieval"

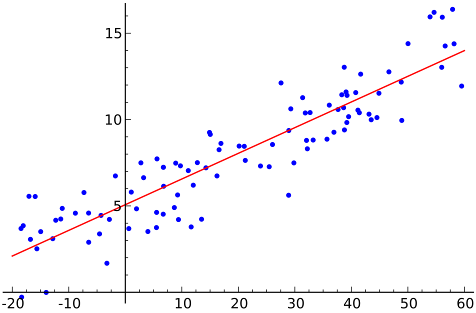
Pointwise LTR



Pointwise LTR

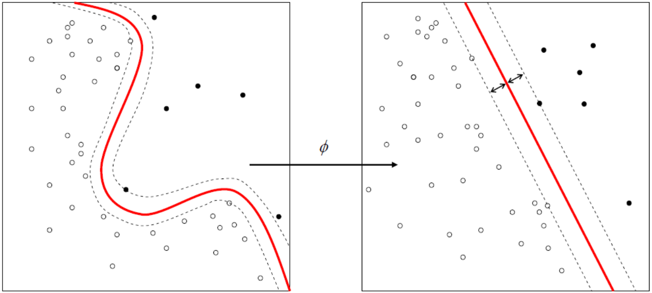
- Reduces to traditional ML
- Input: query-document feature vectors
  $\mathbf{x}_i^{(n)} = [x_{i1}^{(n)}, x_{i2}^{(n)}, \dots, x_{il}^{(n)}]$
- Output: relevance labels y_i
- Objective: learn a model $h(\mathbf{x})$ that correctly predicts labels y

Regression



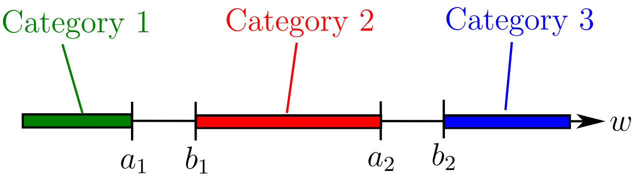
Picture taken from https://en.wikipedia.org/wiki/Regression_analysis

Classification



Picture taken from https://en.wikipedia.org/wiki/Statistical_classification

Ordinal regression

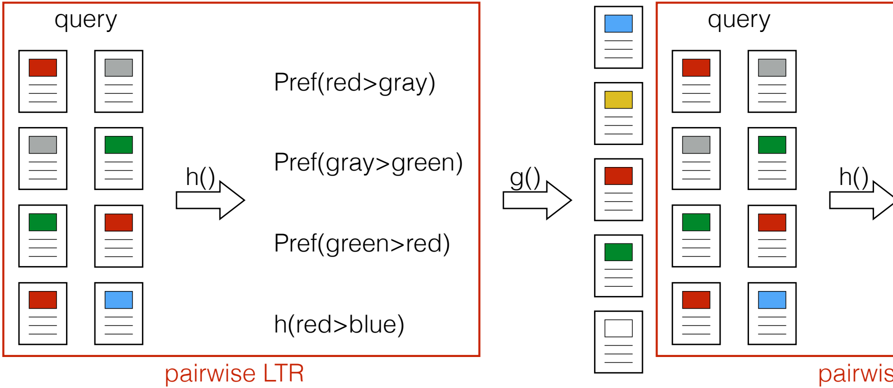


T.-Y. Liu, "Learning to Rank for Information Retrieval"

Pointwise LTR

- Pros
  - + Intuitive interpretation of relevance
  - + Clear, how to get relevance judgements
- Cons
  - – Has a different optimization objective compared to IR (e.g., finding a correct class)

Pairwise LTR



RankNet

- *Pointwise* scoring function $f(\mathbf{x}_i)$ with parameters $\{w_k\}_{k=1}^{l}$
- *Pairwise* ground-truth $\overline{P}_{ij} = \mathcal{I}(\mathbf{x}_i > \mathbf{x}_j)$
- Probability of $\mathbf{x}_i > \mathbf{x}_j$ is modeled using logistic regression

$$P_{ij} = P(\mathbf{x}_i > \mathbf{x}_j) = \frac{1}{1 + e^{-\sigma(f_i - f_j)}}$$

- *Pairwise* loss function (cross entropy)

$$\begin{aligned} C &= -\overline{P}_{ij} \log P_{ij} - (1 - \overline{P}_{ij}) \log(1 - P_{ij}) \\ &= (1 - \overline{P}_{ij})\sigma(f_i - f_j) + \log(1 + e^{-\sigma(f_i - f_j)}) \end{aligned}$$

- RankNet optimizes the total number of pairwise errors

RankNet (cont'd)

- Optimize the cost C

$$\frac{\partial C}{\partial f_i} = \sigma \left[(1 - \overline{P}_{ij}) - \frac{1}{1 + e^{\sigma(f_i - f_j)}} \right] = -\frac{\partial C}{\partial f_j}$$

- Update parameter w_k of the function $f(\mathbf{x}_i)$

$$w_k \leftarrow w_k - \eta \left(\frac{\partial C}{\partial f_i} \frac{\partial f_i}{\partial w_k} + \frac{\partial C}{\partial f_j} \frac{\partial f_j}{\partial w_k} \right)$$

Speeding up RankNet training

- Define λ_{ij} as

$$\lambda_{ij} = \frac{\partial C}{\partial f_i} = \sigma \left[(1 - \overline{P}_{ij}) - \frac{1}{1 + e^{\sigma(f_i - f_j)}} \right]$$

- Let $\mathcal{I}$ denote the set of pairs of indices $\{i, j\}$, for which x_i should be ranked differently from x_j for a given query q

$$\mathcal{I} = \{i, j \mid \mathbf{x}_i > \mathbf{x}_j\}$$

- Sum all contributions to update parameter w_k

$$\begin{aligned}
\delta w_k &= -\eta \sum_{\{i,j\} \in \mathcal{I}} \left(\lambda_{ij} \frac{\partial f_i}{\partial w_k} - \lambda_{ij} \frac{\partial f_j}{\partial w_k} \right) \\
&= -\eta \sum_i \left(\sum_{j:\{i,j\} \in \mathcal{I}} \lambda_{ij} \frac{\partial f_i}{\partial w_k} - \sum_{j:\{j,i\} \in \mathcal{I}} \lambda_{ij} \frac{\partial f_i}{\partial w_k} \right) \\
&= -\eta \sum_i \lambda_i \frac{\partial f_i}{\partial w_k}
\end{aligned}$$

Interpreting λ's

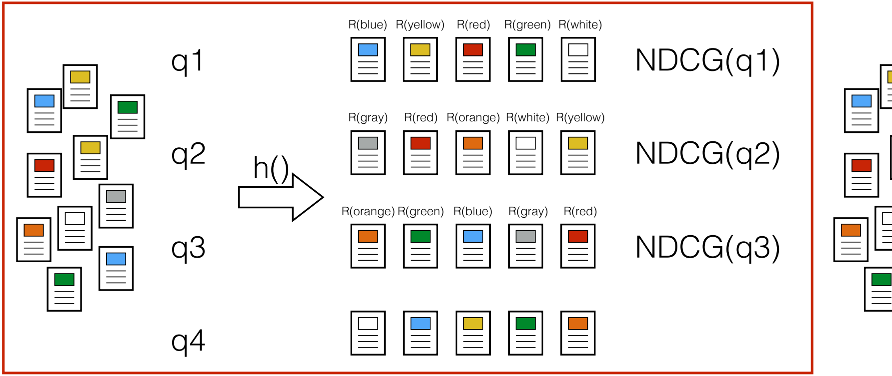
$$\lambda_i = \sum_{j:\{i,j\}\in\mathcal{I}} \lambda_{ij} - \sum_{j:\{j,i\}\in\mathcal{I}} \lambda_{ij}$$

- λ_i is a sum of "forces" applied to document x_i shown for query q
- All documents x_j, that should be ranked below x_i, push it up with the force λ_{ij}
- All documents x_j, that should be ranked above x_i, push it down with the force λ_{ij}

Pairwise LTR

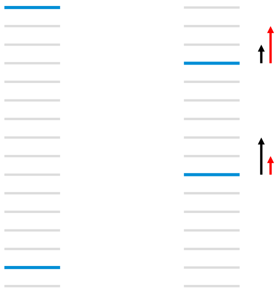
- Pros
  - + Easy to get preference judgements
  - + Comes closer to optimizing the ranking
- Cons
  - – Still does not optimize the whole ranking
  - – Higher computational complexity compared to pointwise LTR

Listwise LTR



listwise LTR

From RankNet to LambdaRank



- The black arrows denote the RankNet gradients (which increase with the number of pairwise errors)
- RankNet cost decreases from 13 on the left to 11 on the right
- The actual ranking gets worse
- The red arrows is what we would actually like to see

C. Burges, "From RankNet to LambdaRank to LambdaMART: An Overview"

LambdaRank

- λ_{ij} in RankNet

$$\lambda_{ij} = \sigma \left[(1 - \overline{P}_{ij}) - \frac{1}{1 + e^{\sigma(f_i - f_j)}} \right]$$

- λ_{ij} in LambdaRank

$$\lambda_{ij} = \frac{-\sigma}{1 + e^{\sigma(f_i - f_j)}} |\Delta_{NDCG}|$$

- $|\Delta_{NDCG}| = NDCG(\text{orig. ranking}) - NDCG(x_i \text{ and } x_j \text{ are swapped})$
- LambdaRank directly uses the ranking to compute gradients (i.e., λ_{ij}'s) instead of computing and optimizing a cost function

LambdaRank (cont'd)

Proceed similarly to RankNet:

- Sum all λ_{ij}'s for document x_i and query q

$$\lambda_i = \sum_{j:\{i,j\} \in I} \lambda_{ij} - \sum_{j:\{j,i\} \in I} \lambda_{ij}$$

- Update parameter w_k of the function $f(\mathbf{x}_i)$

$$w_k \leftarrow w_k - \eta \sum_i \lambda_i \frac{\partial f_i}{\partial w_k}$$

LambdaMART

- Multiple Additive Regression Trees (MART)
- MART does not need a cost function but gradients
- Adopts gradients (λ_{ij}'s) from LambdaRank
- Hence the name: Lambda + MART

Listwise LTR

- Pros
  + Directly optimizes the whole ranking
- Cons
  – Needs many judgements
  – High computational complexity

Outline

LEarning TO Rank datasets (LETOR)

- Query-document pairs – precomputed feature vectors
- Relevance judgements

http://research.microsoft.com/en-us/um/beijing/projects/letor/

Historical LTR datasets

- TREC 2003, 2004 Web IR track
- "Gov" corpus with $1,053,110$ pages
- Tasks
  - TD – topic distillation
  - HP – homepage finding
  - NP – named page finding

Task	TREC2003	TREC2004
Topic distillation	50	75
Homepage finding	150	75
Named page finding	150	75

Figure: Number of queries

T.-Y. Liu, "Learning to Rank for Information Retrieval"

Results on NP2003

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.447	0.614	0.665	0.447	0.220	0.081	0.564
RankSVM	0.580	0.765	0.800	0.580	0.271	0.092	0.696
RankBoost	0.600	0.764	0.807	0.600	0.269	0.094	0.707
FRank	0.540	0.726	0.776	0.540	0.253	0.090	0.664
ListNet	0.567	0.758	0.801	0.567	0.267	0.092	0.690
AdaRank	0.580	0.729	0.764	0.580	0.251	0.086	0.678
SVM^{map}	0.560	0.767	0.798	0.560	0.269	0.089	0.687

T.-Y. Liu, "Learning to Rank for Information Retrieval"

Results on HP2004

Algorithm	N@1	N@3	N@10	P@1	P@3	P@10	MAP
Regression	0.387	0.575	0.646	0.387	0.213	0.08	0.526
RankSVM	0.573	0.715	0.768	0.573	0.267	0.096	0.668
RankBoost	0.507	0.699	0.743	0.507	0.253	0.092	0.625
FRank	0.600	0.729	0.761	0.600	0.262	0.089	0.682
ListNet	0.600	0.721	0.784	0.600	0.271	0.098	0.690
AdaRank	0.613	0.816	0.832	0.613	0.298	0.094	0.722
SVM^{map}	0.627	0.754	0.806	0.627	0.280	0.096	0.718

T.-Y. Liu, "Learning to Rank for Information Retrieval"

Experimental comparison

- Listwise ranking algorithms perform very well on most datasets
- ListNet seems to be better than the others
- Pairwise ranking algorithms obtain good ranking accuracy on some (although not all) datasets
- Linear regression performs worse than the pairwise and listwise ranking algorithms

T.-Y. Liu, "Learning to Rank for Information Retrieval"

Outline

2 Learning to rank
- Machine learning
- Features
- LTR approaches
- Experimental comparison
- Summary

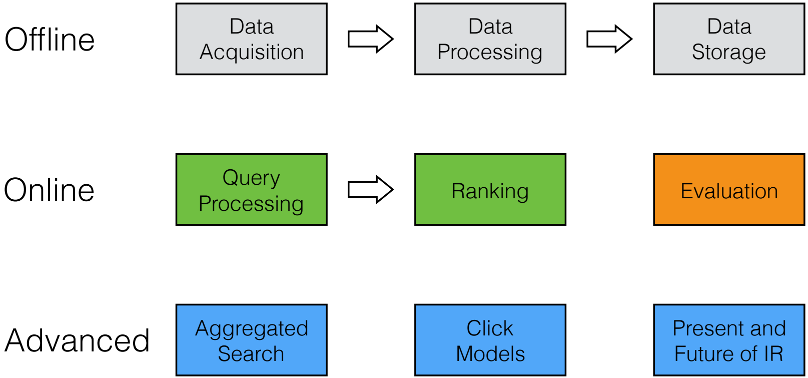
Learning to rank summary

- Features
  - Content-based
  - Link-based
  - User-based
- Approaches
  - Pointwise (regression, classification, ordinal regression)
  - Pairwise (RankNet)
  - Listwise (LambdaRank, LambdaMART)

Materials

- Tie-Yan Liu
  Learning to Rank for Information Retrieval
  Foundations and Trends in Information Retrieval, 2009
- Christopher J.C. Burges
  From RankNet to LambdaRank to LambdaMART: An Overview
  Microsoft Research Technical Report, 2010

Course overview



See you tomorrow!

Offline	Data Acquisition	⇨	Data Processing	⇨	Data Storage
Online	Query Processing	⇨	Ranking		Evaluation
Advanced	Aggregated Search		Click Models		Present and Future of IR