THE UNIVERSITY of EDINBURGH

Text Technologies for Data Science

INFR11145

Learning to Rank

Instructor
Björn Ross

29-Nov-2023

Pre-Lecture

- Only one lecture today
- Last lecture in the course
- No lab
- After the lecture:
  Info on group project (coursework 3)

Lecture Objectives

- Learn about:
  - IR as a classification task
  - Learning to Rank approaches

Classical Models vs. ML in IR

- Classical Models:
  - Features (factors): only a few, e.g., TF, IDF, |D|, P(t|corpus) etc.
  - Structure: optimized for the a few particular features
  - Parameter & training
    - Often 1-2; not every factor has a parameter controlling its influence
    - Hand-tuning or data-based; can tune exhaustively since just 1-2 parameters
  - *tfidf* or BM25 or LMIR? PRF? What n_d, n_t?
- ML in IR
  - Features: can include up to hundreds, thousands, or even more
  - Define the basic structure of a model
  - Quite generic: such as a weighted linear combination of all features
  - Parameters & training
    - Many; control the influence of each feature and their combinations
    - Impossible to tune by hand; Must be data-driven
  - Let the ML decide what is better!

Text Classification in IR

- Text Classification:
  - Classify a document into one of two or more classes
  - Different features could be used, e.g. BOW
- Can we model IR as classification?
  - Classify document to C1: R or C2: NR
  - Challenges?
    - Training data?
    - Features? BOW?
- BOW features cannot work
  - Spam? Viagra, @ed.ac.uk
  - Sentiment? happy, sad
  - Relevant? Trump, hurricane
  - Relevance depends on the query!

From Classification to IR

- Transforming features
  - Text classification: Input (D) → output (yes/no)
  - Information Filtering: Input (D|Q) → output (yes/no)
- Feature set:
  - Independent of absolute words
  - More on relation between doc and query
  - Mostly numbers (formulas, frequencies, …)
  - As consistent as possible among different Q,D pairs
  - e.g.:
    - TFIDF, BM25
    - Query in page title? Heading?
    - Query in anchor text linking pages
    - PageRank of doc
    - Number of times page clicked for the same query

Popular Features

Column in Output	Description
1	TF(Term frequency) of body
2	TF of anchor
3	TF of title
4	TF of URL
5	TF of whole document
6	IDF(Inverse document frequency) 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	DL(Document length) of body
17	DL of anchor
18	DL of title
19	DL of URL
20	DL of whole document
21	BM25 of body
22	LMIR.ABS of body
23	LMIR.DIR of body

Column in Output	Description
24	LMIR.JM of body
25	BM25 of anchor
26	LMIR.ABS of anchor
27	LMIR.DIR of anchor
28	LMIR.JM of anchor
29	BM25 of title
30	LMIR.ABS of title
31	LMIR.DIR of title
32	LMIR.JM of title
33	BM25 of URL
34	LMIR.ABS of URL
35	LMIR.DIR of URL
36	LMIR.JM of URL
37	BM25 of whole document
38	LMIR.ABS of whole document
39	LMIR.DIR of whole document
40	LMIR.JM of whole document
41	PageRank
42	Inlink number
43	Outlink number
44	Number of slash in URL
45	Length of URL
46	Number of child page

Training Data

- Training data: {R,X}
  - X: feature representation of (D,Q) pairs
  - R = {-1,+1} … is D relevant to Q or no
- Samples:
  - Large set of (D,Q) pairs
  - Wide range of Q's (long/short, frequent/rare, …)
  - Wide range of D's for each Q (top/deep ranked, recent/old pages, …)
- Labels:
  - Manually labelled: assessors judge relevance of docs to queries (similar to standard IR)
  - Automatically labelled: click-through data

Classification or Ranking?

- Click-through data
  - User clicks can give indication of relevance
  - What about non-relevance?
  - A list of ranked results: D1 → D2 → D3
    user clicked on D3 and neglected D1 & D2
    what does it mean?
    - D3 is relevant and D1 & D2 are not relevant?
    - Relevance: D3 > D1 & D2?
- It might be better to model the problem as ranking
  - Label→ Ranking preference (e.g. gain={4,3,2,1,0})
  - Learning→ to optimize $\text{Doc}_X > \text{Doc}_Y$
    not to classify them to R/NR
  - Input: features for **set of** docs for a given query
    Objective: rank them (sort by relevance)

ML & IR: History

- Considerable interaction between these fields
  - Rocchio algorithm (60s) is a simple learning approach
  - 80s, 90s: learning ranking algorithms based on user feedback
  - 2000s: text categorization
- Limited by amount of training data
- Web query logs have generated new wave of research
  - L2R (LTR): “Learning to Rank”

What is Learning-to-Rank?

- Purpose
  - Learn a function automatically to rank results effectively
- Point-wise approach
  - Classify document to R / NR
- List-wise
  - The function is based on a ranked list of items
  - given two ranked list of the same items, which is better
- Pair-wise
  - The function is based on a pair of item
  - e.g., given two documents, predict partial ranking

Point-wise Approaches

- The function is based on features of a single object
  - e.g., regress the rel. score, classify docs into R and NR
- Very similar to classification
  - Examples of (D,Q) pairs with labels 1 or 0
- Classic retrieval models are also point-wise:
  - Calculate score(Q, D)
  - *If* score(Q,D) > θ → relevant
    else, irrelevant
- Referred to as *information filtering*
  - Standing query + new documents coming
  - Decide whether a new document is R or NR

List-based Approaches

- Given: ranked list A and ranked list B
  Task: decide which is better
- Need a loss function on a list of documents
- Challenge is scale
  - Huge number of potential lists
- Can develop tricks
  - Consider only possible re-rankings of top N retrieved by some fixed method
- Still expensive
  - No clear benefits over pairwise ones (so far)

Pair-wise Approaches

- Trying to classify
  - Which document of two should be ranked at a higher position?
- Optimize based on:
  - Margin between decision hyperplane and instances
  - Errors
  - Weighted based on some hyper-parameter C
  - Evaluation metric
- Example: SVM-rank
  - A generalization of SVM that supports ranking [Herbrichet al. 1999, 2000; Joachims et al. 2002]

SVM-rank Example

Q1
```
3 qid:1 1:1 2:1 3:0 4:0.2 5:0 # 1A
2 qid:1 1:0 2:0 3:1 4:0.1 5:1 # 1B
1 qid:1 1:0 2:1 3:0 4:0.4 5:0 # 1C
1 qid:1 1:0 2:0 3:1 4:0.3 5:0 # 1D
```
Q2
```
1 qid:2 1:0 2:0 3:1 4:0.2 5:0 # 2A
2 qid:2 1:1 2:0 3:1 4:0.4 5:0 # 2B
1 qid:2 1:0 2:0 3:1 4:0.1 5:0 # 2C
1 qid:2 1:0 2:0 3:1 4:0.2 5:0 # 2D
```
Q3
```
2 qid:3 1:0 2:0 3:1 4:0.1 5:1 # 3A
3 qid:3 1:1 2:1 3:0 4:0.3 5:0 # 3B
4 qid:3 1:1 2:0 3:0 4:0.4 5:1 # 3C
1 qid:3 1:0 2:1 3:1 4:0.5 5:0 # 3D
```

relevance (rank importance)

Set of features for (D,Q) pair

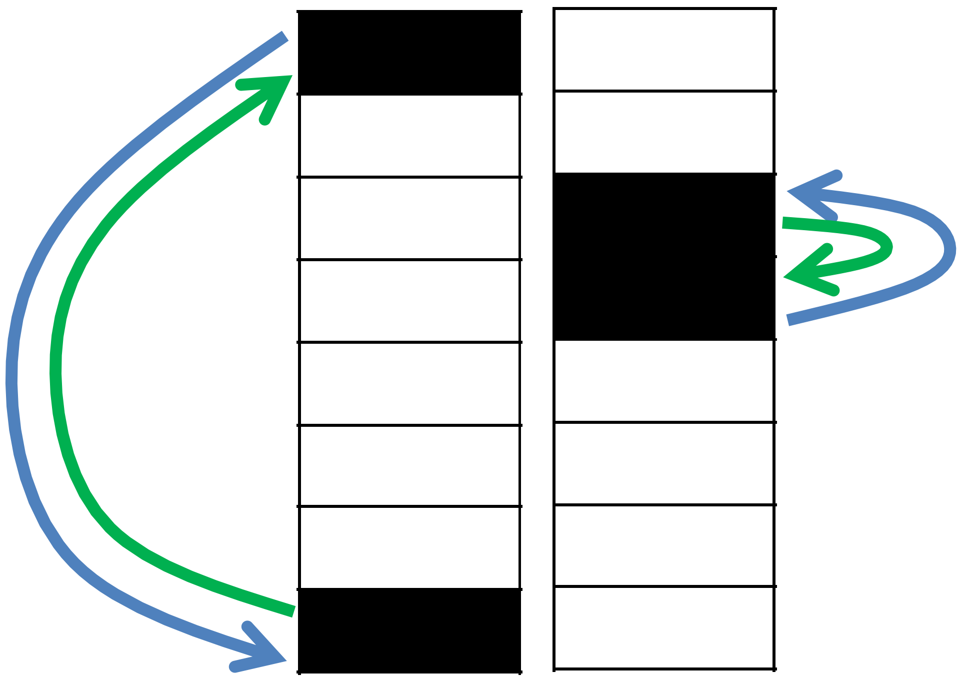
- Q3: 3C>3A, 3C>3B, 3C>3D, 3B>3A, 3B>3D, 3A>3D

Pair-wise Approaches

- The most popular approach
- Learning methods: SVM-rank, RankBoost, GBRank, Ranknet, LambdaRank, LambdaMART
- Pairwise ranking error often has better correlations with evaluation metrics than the loss/objective functions in point-wise approaches
  - Why: evaluation measures only care about rankings!
    e.g., ground-truth: rel(D1) = 3, rel(D2) = 2
    - Regression model 1: pred.rel(D1) = 2, pred.rel(D2) = 3
    - Regression model 2: pred.rel(D1) = 1, pred.rel(D2) = 0
    - Model 1 is better than model 2 by criterion of evaluation regression (the prediction error), but model 2 yields a correct ranking of docs
- Still, issues with ranking SVM e.g. it does not directly optimize an evaluation metric

Pair-wise Approaches

- LambdaMART:
  - Misordered pairs are not equally important
  - Depends on how much they contribute to the changes in the target evaluation measure



Pair-wise Approaches

- Optimizing for an evaluation metric
  - The general idea is to weight loss/objective function or gradient with pairwise changes in evaluation measure.
  - e.g., in LambdaMART: lambda gradient
- Can we optimize all measures?
  - Not necessarily
  - For some measures, pairwise changes do not only relate to the two documents themselves, but also others …
    - Position-based measures do not have the issues (pairwise change only depends on the two documents)
    - Cascade measures may have issues

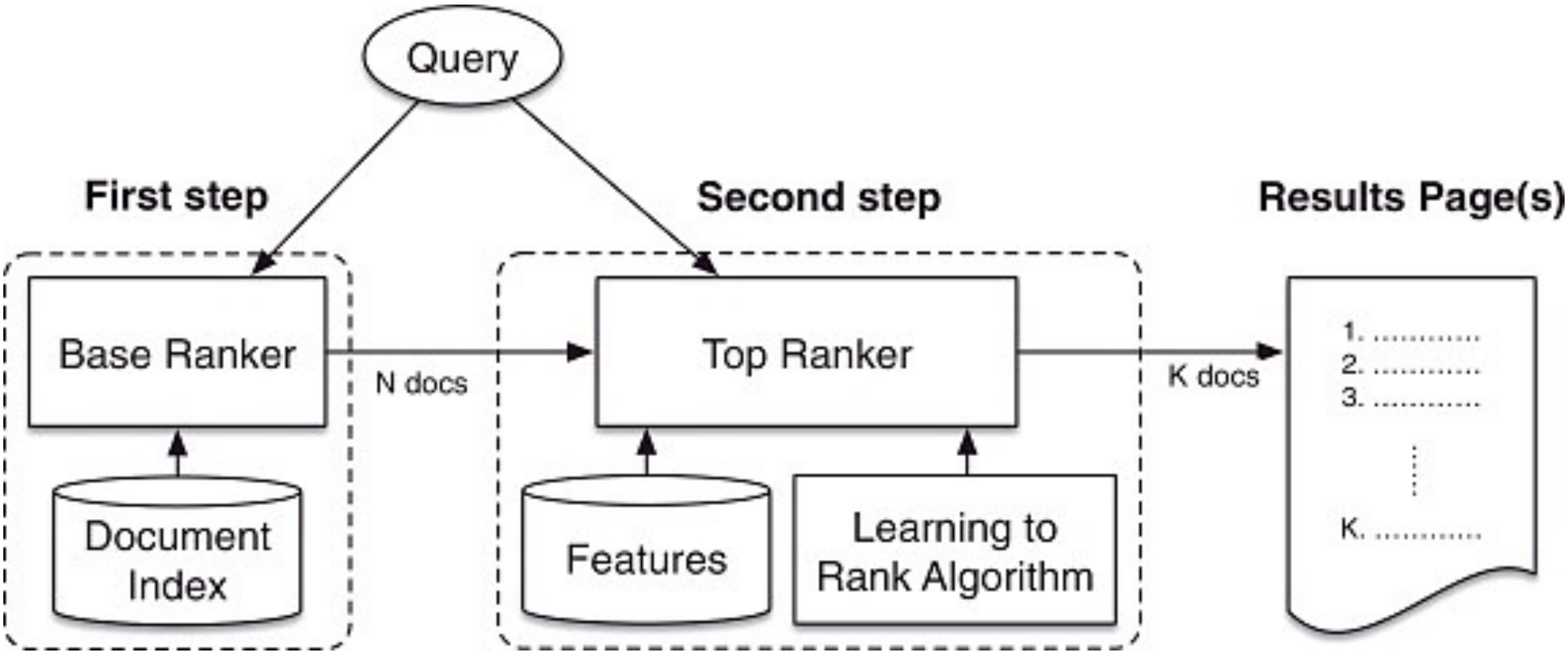
Pair-wise Approaches: Example

- Experiments
  - 1.2k queries, 45.5K documents with 1890 features
  - 800 queries for training, 400 queries for testing

	MAP	P@1	ERR	MRR	NDCG@5
ListNET	*0.2863*	*0.2074*	*0.1661*	*0.3714*	*0.2949*
LambdaMART	**0.4644**	**0.4630**	**0.2654**	**0.6105**	**0.5236**
RankNET	0.3005	0.2222	0.1873	0.3816	0.3386
RankBoost	0.4548	0.4370	0.2463	0.5829	0.4866
RankingSVM	0.3507	0.2370	0.1895	0.4154	0.3585
AdaRank	0.4321	0.4111	0.2307	0.5482	0.4421
pLogistic	0.4519	0.3926	0.2489	0.5535	0.4945
Logistic	0.4348	0.3778	0.2410	0.5526	0.4762

Honglin Wang Slides

L2R in Practice

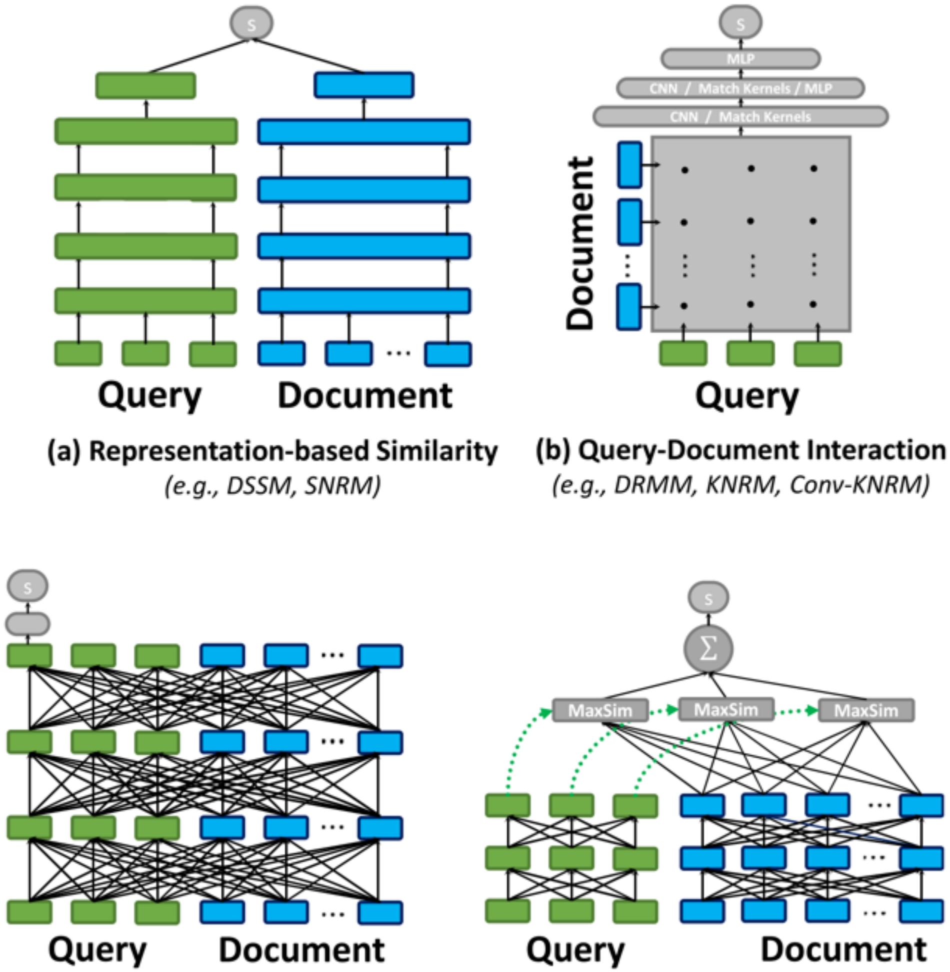


Capannini, G., et al.
Quality versus efficiency in document scoring with learning-to-rank models.
IP&M 2016.

Current work in L2R

- Deep learning models are mainly used
- No manual feature extraction is applied
- Using word-embeddings to represent queries and docs, then learn the features automatically
- Content-independent models: try to learn the pattern of relations between terms in Q and D
- Content dependent: dependent on the terms

Types of Deep LTR Models



(a) Representation-based Similarity (e.g., DSSM, SNRM)

(b) Query-Document Interaction (e.g., DRMM, KNRM, Conv-KNRM)

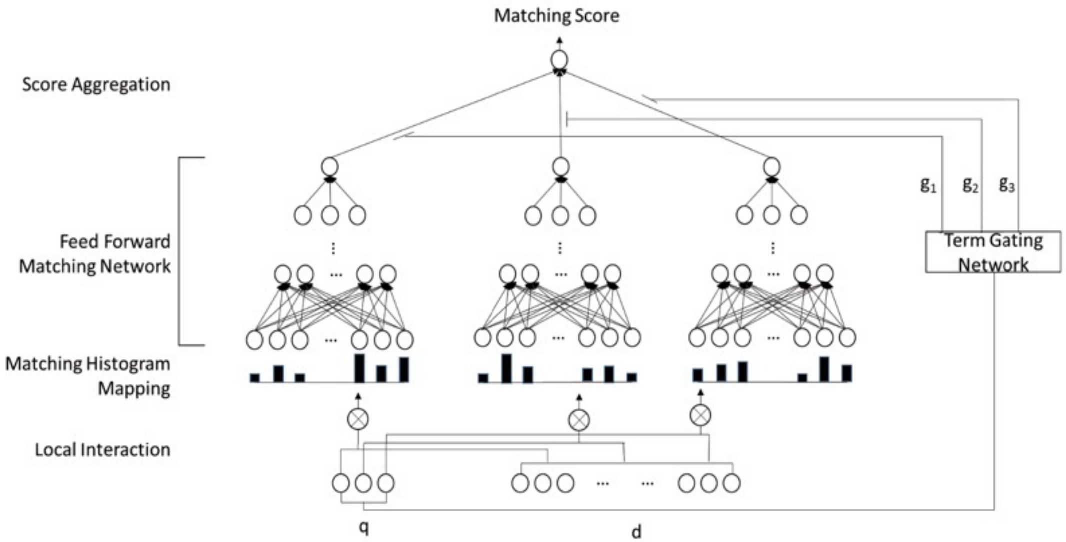
(c) All-to-all Interaction (e.g., BERT)

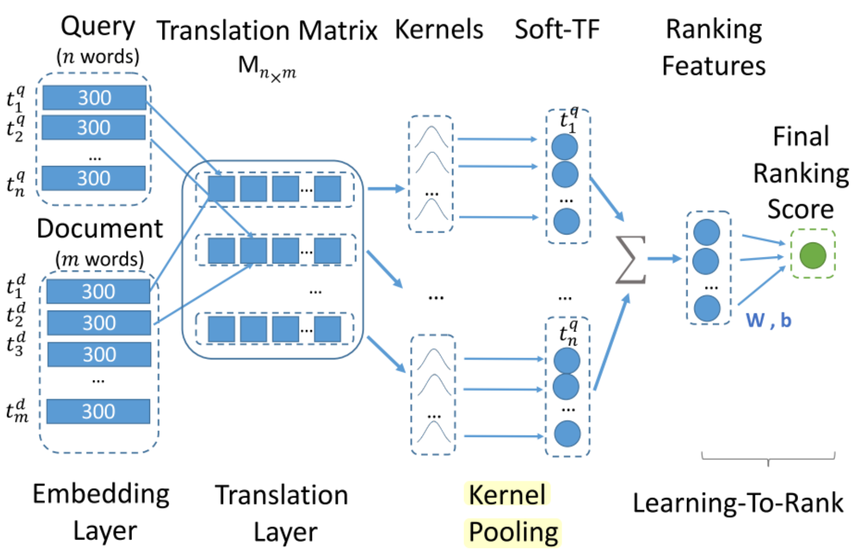
(d) Late Interaction (i.e., the proposed ColBERT)

- Early Interaction-based: Learn on the signals from a query-document interaction.
- Late Interaction (Representation) based: Learn independent representations of queries and documents and then consider the interaction between them.
- Early interaction based approaches, e.g. DRMM, are relatively independent of the content (terms themselves) – tend to generalize well.
- Late interaction based approaches, e.g. ColBERT, are usually data hungry approaches – hence likely not to generalize well on standard ad-hoc IR collections.

By: Debasis Ganguly

DRMM & KNRM





- DRMM (left) uses histograms of word pair similarities (between doc and query) terms as inputs to a feed-forward network.
- The model seeks to utilize inherent patterns in these histograms to distinguish relevance from non-relevance.
- KNRM (right) does not need to rely on histograms. Instead it applies 1D convolution.

By: Debasis Ganguly

Summary

- IR as a classification task
- Learning to rank (L2R) approaches
  - Point-wise
    - Information Filtering
  - List-wise
  - Pair-wise
    - Ranking SVM
    - LambdaMART
- Current work in L2R depends on deep learning models and word-embedding representations

Resources

- Nallapati, Ramesh.
  Discriminative models for information retrieval.
  SIGIR 2004.

- Burges, C. J. (2010).
  From ranknet to lambdarank to lambdamart: An overview.
  Learning, *11*(23-581), 81.

- SVM^{Rank}: http://svmlight.joachims.org/

- L2R test sets:
  - Microsoft's LETOR project
    http://research.microsoft.com/en-us/um/beijing/projects/letor//default.aspx
  - Microsoft L2R datasets
    http://research.microsoft.com/en-us/projects/mslr/default.aspx