Learning to Rank for Information Retrieval

Challenges of Conventional Models

- Manual parameter tuning is usually difficult, especially when there are many parameters and the evaluation measures are non-smooth.
- Manual parameter tuning sometimes leads to over-fitting.
- It is non-trivial to combine the large number of models proposed in the literature to obtain an even more effective model.

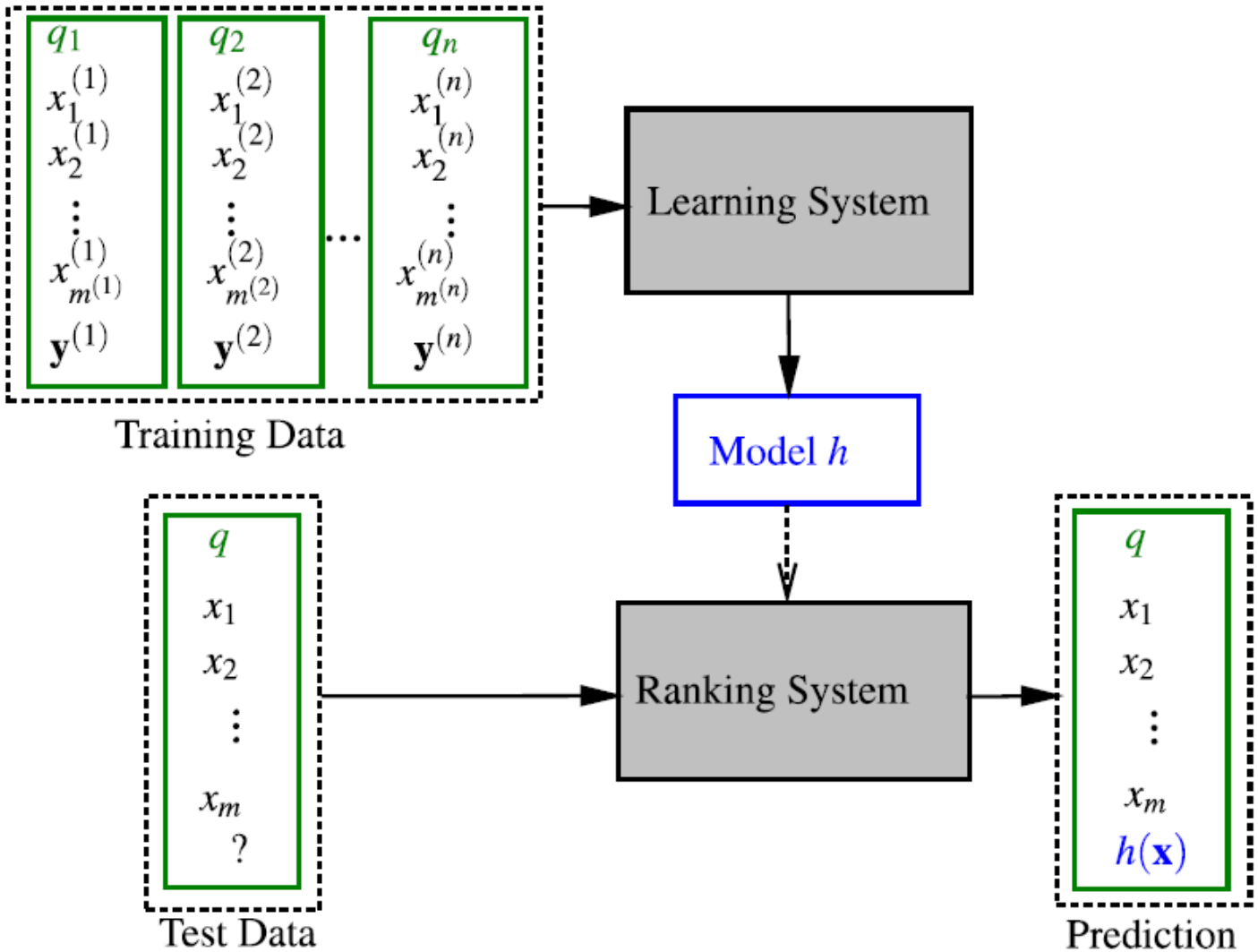
Machine Learning Can Help

- Machine learning is an effective tool
  - To automatically tune parameters.
  - To combine multiple evidences.
  - To avoid over-fitting (by means of regularization, etc.)

Learning to Rank Definition

- “Learning to Rank”
  - In general, those methods that use machine learning technologies to solve the problem of ranking can be named as “learning to rank” methods.
- Learning to rank for Information Retrieval (IR) is a task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance.
- Learning to rank methods are methods that learn how to combine pre-defined features for ranking by discriminative learning.

Learning-to-Rank Framework



Training Data Creation

- LTR is a supervised learning task, and needs high quality training data
- Two common ways to create training data:
  - Human labeling
  - Derivation from click through data

Feature Construction

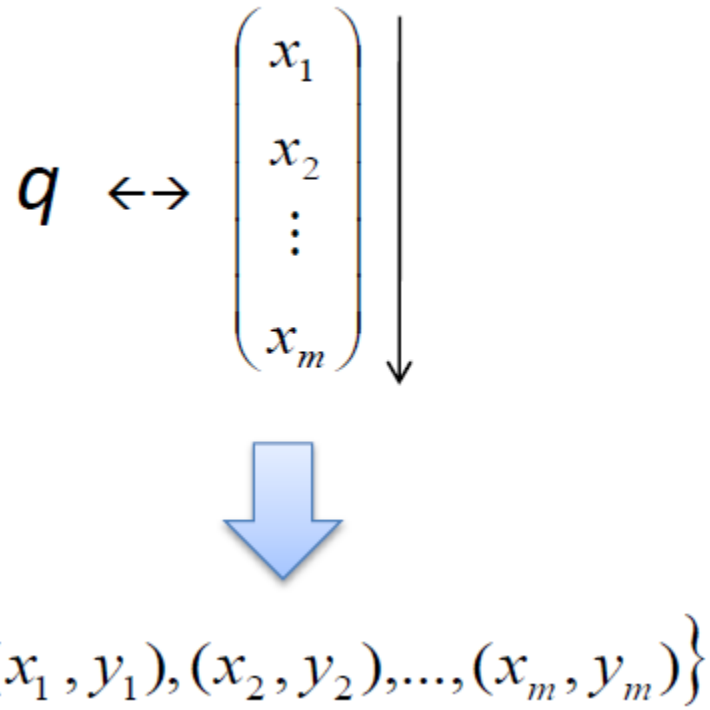
- Query-dependent document features (dynamic features):
  - Depend both on the content of the document and the query.
  - For example, TF-IDF score, BM25 score.
- Query-independent document features (static features):
  - Depend only on the document, but not on the query.
  - For example, PageRank, document length.
- Query level features (query features):
  - Depend only on the query.
  - The values of features for each query are the same for all documents
  - For example, the number of words in a query.

Approaches to Learning-to-Rank

- The pointwise approach
  - Input space: feature vectors of single documents
  - Output space: relevance degree of each document
- The pairwise approach
  - Input space: pairs of documents, each represented as feature vectors
  - Output space: pairwise preference between each pair of documents
- The listwise approach
  - Input space: entire group of documents associated with each query
  - Output space: relevance degrees of all the documents associated with a query

The Pointwise Approach

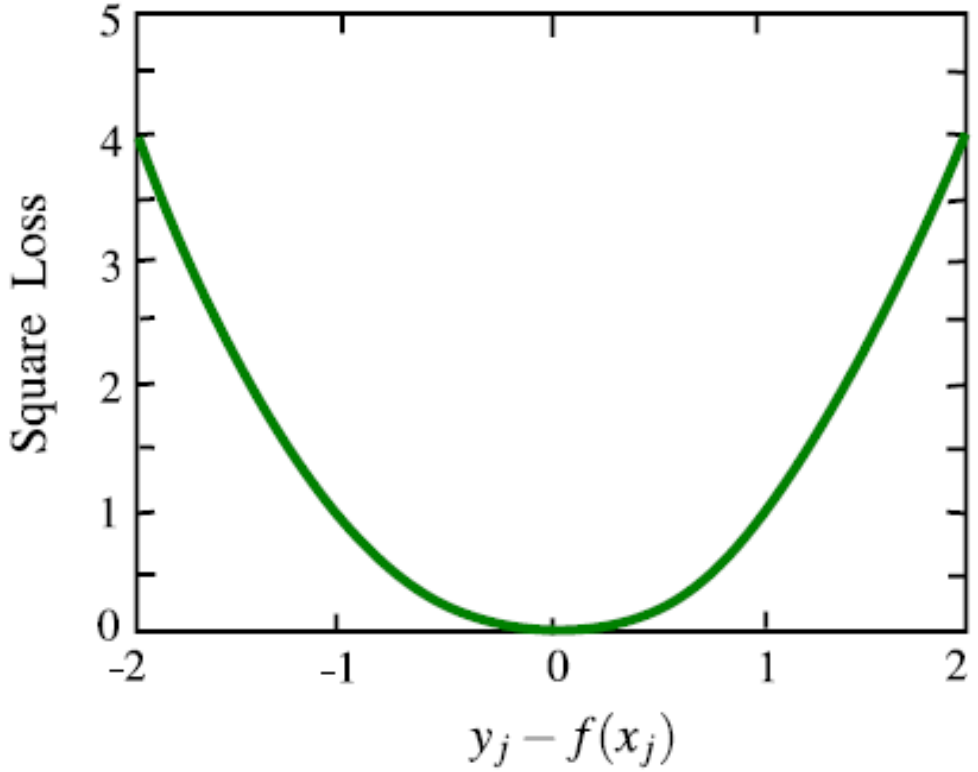
- Use existing learning methods to solve the problem of ranking
- Try to predict the exact relevance degree of each document
- Three subcategories:
  - Regression based algorithms
  - Classification based algorithms
  - Ordinal regression based algorithms



Subset Ranking with Regression

- Regard relevance degree as real number, and use regression to learn the ranking function.

$$L(f; x_j, y_j) = \left(f(x_j) - y_j\right)^2$$



SVM-Based Method

- Regards all the relevant documents (i.e., $y_j = +1$) as positive examples and all the irrelevant documents (i.e., $y_j = -1$) as negative examples
- Uses SVM to perform binary classification:

$$\min \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \sum_{j=1}^{m^{(i)}} \xi_j^{(i)}$$

$$\text{s.t.} \quad w^T x_j^{(i)} \leq -1 + \xi_j^{(i)}, \quad \text{if } y_j^{(i)} = 0.$$

$$w^T x_j^{(i)} \geq 1 - \xi_j^{(i)}, \quad \text{if } y_j^{(i)} = 1.$$

$$\xi_j^{(i)} \geq 0, \quad j = 1, \ldots, m^{(i)}, i = 1, \ldots, n,$$

Each training document is an instance of learning

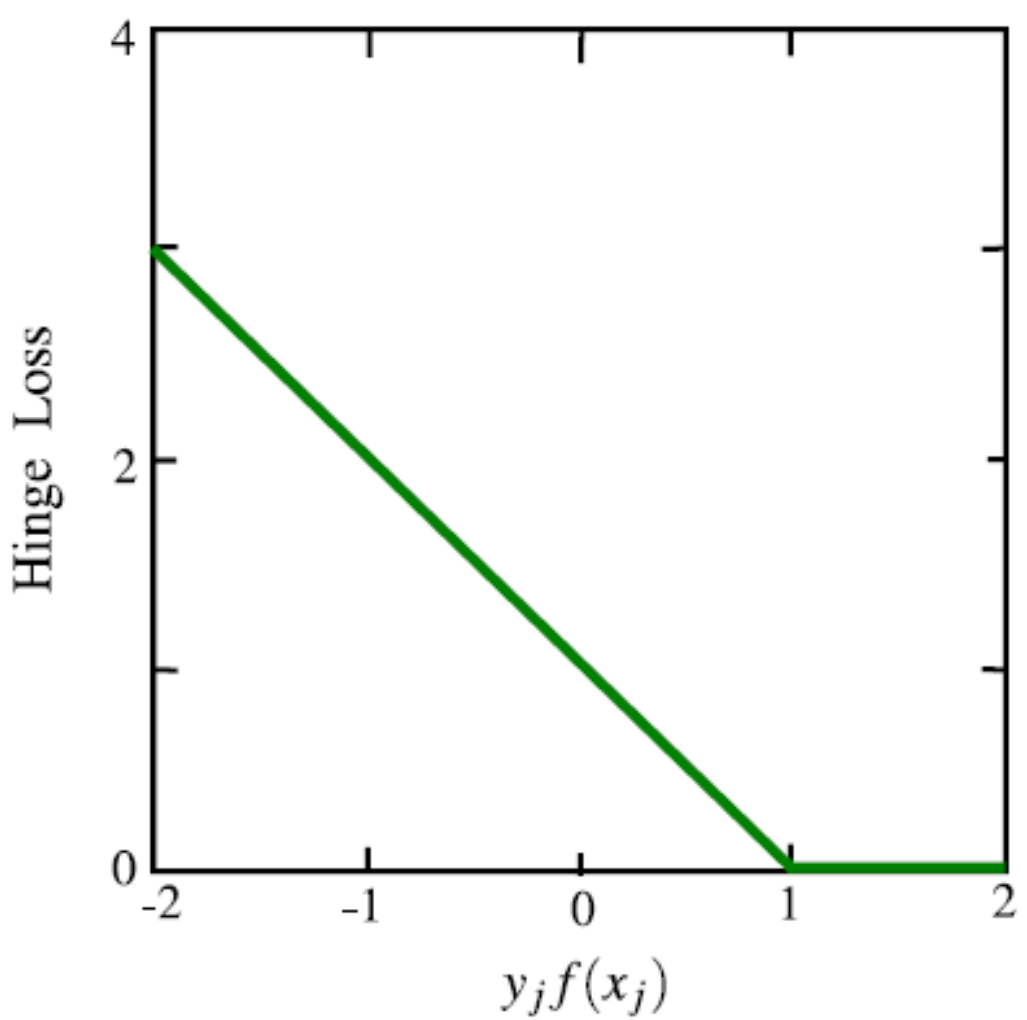
Use SVM to perform binary classification on these instances, to learn model parameter w

$$f(w, x) = w^T x$$

Use w for testing

SVM-Based Method

- $L(f; x_j, y_j) = \left(1 - y_j f(x_j)\right)_+$

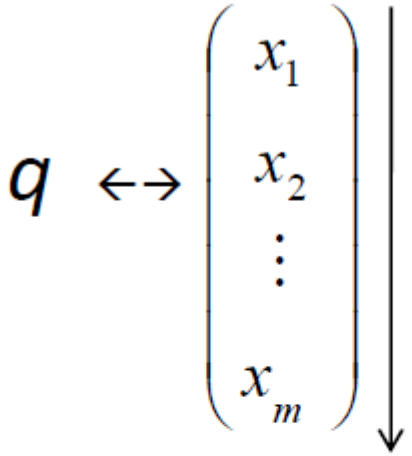


Problems with the Pointwise Approach

- Properties of IR evaluation measures have not been well considered.
  - The fact is ignored that some documents are associated with the same query and some are not.
    - When the number of associated documents varies largely for different queries, the overall loss function will be dominated by those queries with a large number of documents.
- The position of documents in the ranked list is invisible to the loss functions.
  - The pointwise loss function may unconsciously emphasize too much those unimportant documents (which are ranked low in the final results).

The Pairwise Approach

- Cares about the relative order between two documents
- The ranking problem is reduced to classification on document pairs
- Goal: minimize the number of miss-classified document pairs



$$\{(x_1, x_2, +1), (x_2, x_1, -1), \ldots, (x_2, x_m, +1), (x_m, x_2, -1)\}$$

Ranking SVM

- Applies the SVM technology to perform pairwise classification
- Given n training queries $\{q_i\}_{i=1}^{n}$, their associated document pairs $(x_u^{(i)}, x_v^{(i)})$, and the corresponding ground truth label $y_{u,v}^{(i)}$, the mathematical formulation of Ranking SVM is:

$$\min \frac{1}{2}\|w\|^2 + \lambda \sum_{i=1}^{n} \sum_{u,v:y_{u,v}^{(i)}=1} \xi_{u,v}^{(i)}$$

$$\text{s.t.} \quad w^T \left(x_u^{(i)} - x_v^{(i)}\right) \geq 1 - \xi_{u,v}^{(i)}, \text{ if } y_{u,v}^{(i)} = 1,$$

$$\xi_{u,v}^{(i)} \geq 0, \quad i = 1, \ldots, n.$$

$x_u - x_v$ as positive instance of learning

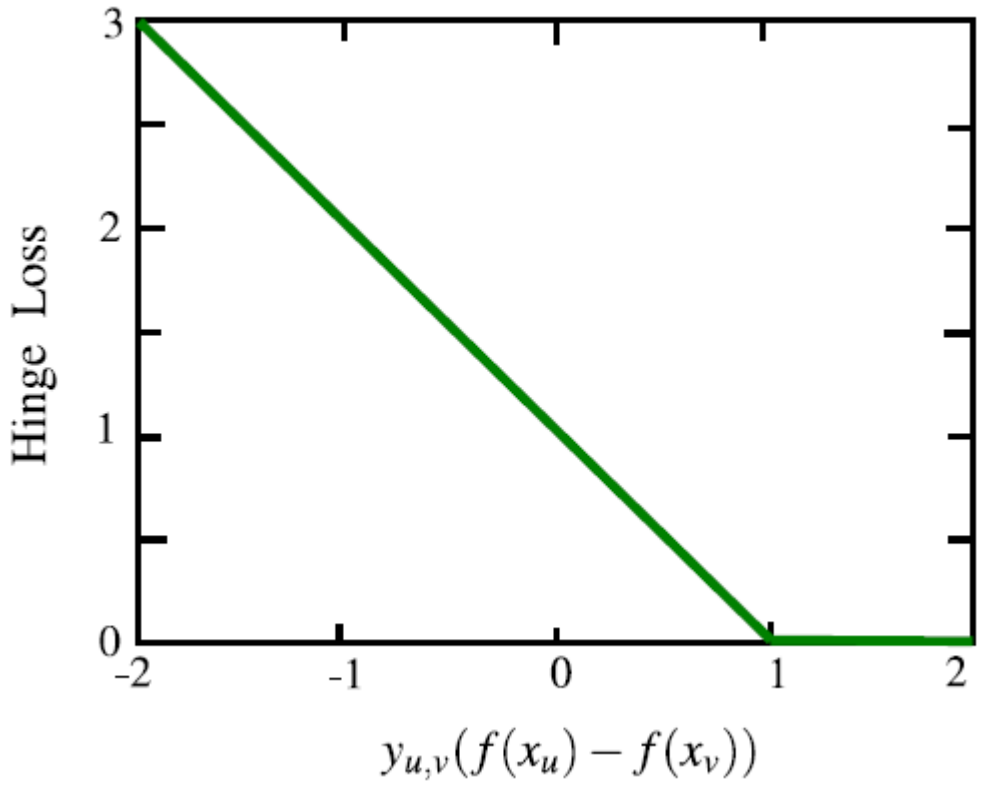
Use SVM to perform binary classification on these instances, to learn model parameter w

$$f(x) = w^T x$$

Use w for testing

Ranking SVM

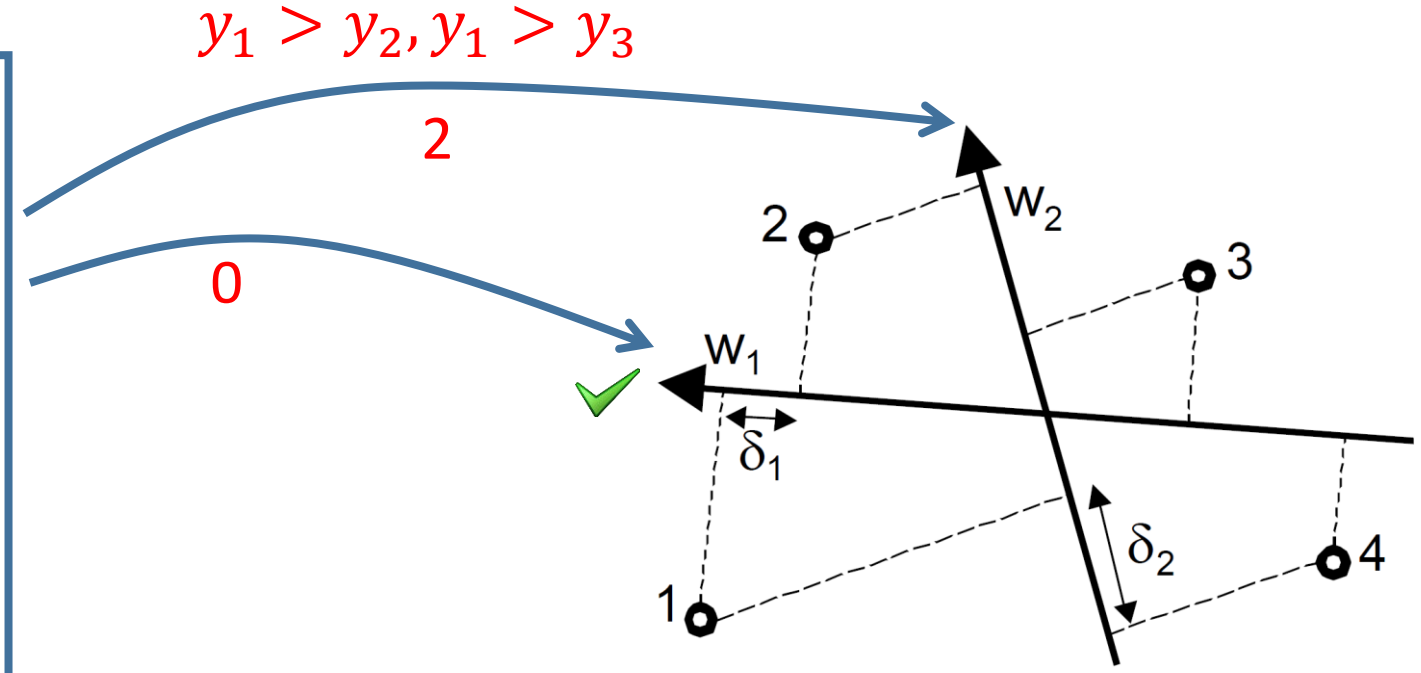
- $L(f; x_u, x_v, y_{u,v}) = \left(1 - y_{u,v}(f(x_u) - f(x_v))\right)_+$



Ranking SVM

- Minimizing the number of misordered pairs

$y_1 > y_2$
$y_1 > y_3$
$y_1 > y_4$
$y_2 > y_3$
$y_2 > y_4$
$y_3 > y_4$



Problems with the Pairwise Approach

- When we are given the relevance judgment in terms of multiple ordered categories, converting it to pairwise preference will lead to the missing of the information about the finer granularity in the relevance judgment
- The distribution of document pair number is more skewed than the distribution of document number, with respect to different queries.
- The pairwise approach is more sensitive to noisy label than the pointwise approach.
- Most of the pairwise ranking algorithms have not considered the position in the final ranking results.

The Listwise Approach

- Measure-specific loss
  - Optimize the IR evaluation measures directly
- Non measure-specific loss
  - Optimize a loss function defined on all documents associated with a query, according to the unique properties of ranking

Measure-specific Listwise Ranking

- Also known as “Direct Optimization of IR Measures”.
- It is natural to directly optimize what is used to evaluate the ranking results.
- However, it is non-trivial.
  - Evaluation measures such as NDCG are non-continuous and non-differentiable since they depend on the rank positions.
  - It is challenging to optimize such objective functions, since most optimization techniques in the literature were developed to handle continuous and differentiable cases.

Tackle the Challenges

- Approximate the objective
  - Soften (approximate) the evaluation measure so as to make it smooth and differentiable (SoftRank)
- Bound the objective
  - Optimize a smooth and differentiable upper bound of the evaluation measure (SVM-MAP)
- Optimize the non-smooth objective directly
  - Use IR measure to update the distribution in Boosting (AdaRank)
  - Use genetic programming (RankGP)

Non Measure-specific Listwise Ranking

- Defining listwise loss functions based on the understanding on the unique properties of ranking for IR.
- Representative Algorithms
  - ListNet
  - ListMLE
  - BoltzRank

Discussions

- Advantages
  - Take all the documents associated with the same query as the learning instance
  - Rank position is visible to the loss function
- Problems
  - Complexity issue.
  - The use of the position information is insufficient.

Benchmark Datasets

	Queries	Doc.	Rel.	Feat.	Year
Letor 3.0 – Gov	575	568 k	2	64	2008
Letor 3.0 – Ohsume	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

GOV Features

ID	Feature description
1	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in body
2	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in anchor
3	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in title
4	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in URL
5	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in the whole document
6	$\sum_{t_i \in q} IDF(t_i)$ in body
7	$\sum_{t_i \in q} IDF(t_i)$ in anchor
8	$\sum_{t_i \in q} IDF(t_i)$ in title
9	$\sum_{t_i \in q} IDF(t_i)$ in URL
10	$\sum_{t_i \in q} IDF(t_i)$ in the whole document
11	$\sum_{t_i \in q \cap d} TF(t_i, d) \cdot IDF(t_i)$ in body
12	$\sum_{t_i \in q \cap d} TF(t_i, d) \cdot IDF(t_i)$ in anchor
13	$\sum_{t_i \in q \cap d} TF(t_i, d) \cdot IDF(t_i)$ in title
14	$\sum_{t_i \in q \cap d} TF(t_i, d) \cdot IDF(t_i)$ in URL
15	$\sum_{t_i \in q \cap d} TF(t_i, d) \cdot IDF(t_i)$ in the whole document
16	$LEN(d)$ of body
17	$LEN(d)$ of anchor
18	$LEN(d)$ of title
19	$LEN(d)$ of URL
20	$LEN(d)$ of the whole document
21	BM25 of body
22	BM25 of anchor
23	BM25 of title
24	BM25 of URL

GOV Features

25	BM25 of the whole document
26	LMIR.ABS of body
27	LMIR.ABS of anchor
28	LMIR.ABS of title
29	LMIR.ABS of URL
30	LMIR.ABS of the whole document
31	LMIR.DIR of body
32	LMIR.DIR of anchor
33	LMIR.DIR of title
34	LMIR.DIR of URL
35	LMIR.DIR of the whole document
36	LMIR.JM of body
37	LMIR.JM of anchor
38	LMIR.JM of title
39	LMIR.JM of URL
40	LMIR.JM of the whole document
41	Sitemap based term propagation
42	Sitemap based score propagation
43	Hyperlink based score propagation: weighted in-link
44	Hyperlink based score propagation: weighted out-link
45	Hyperlink based score propagation: uniform out-link
46	Hyperlink based feature propagation: weighted in-link
47	Hyperlink based feature propagation: weighted out-link
48	Hyperlink based 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
60	Number of child page
61	BM25 of extracted title
62	LMIR.ABS of extracted title
63	LMIR.DIR of extracted title
64	LMIR.JM of extracted title

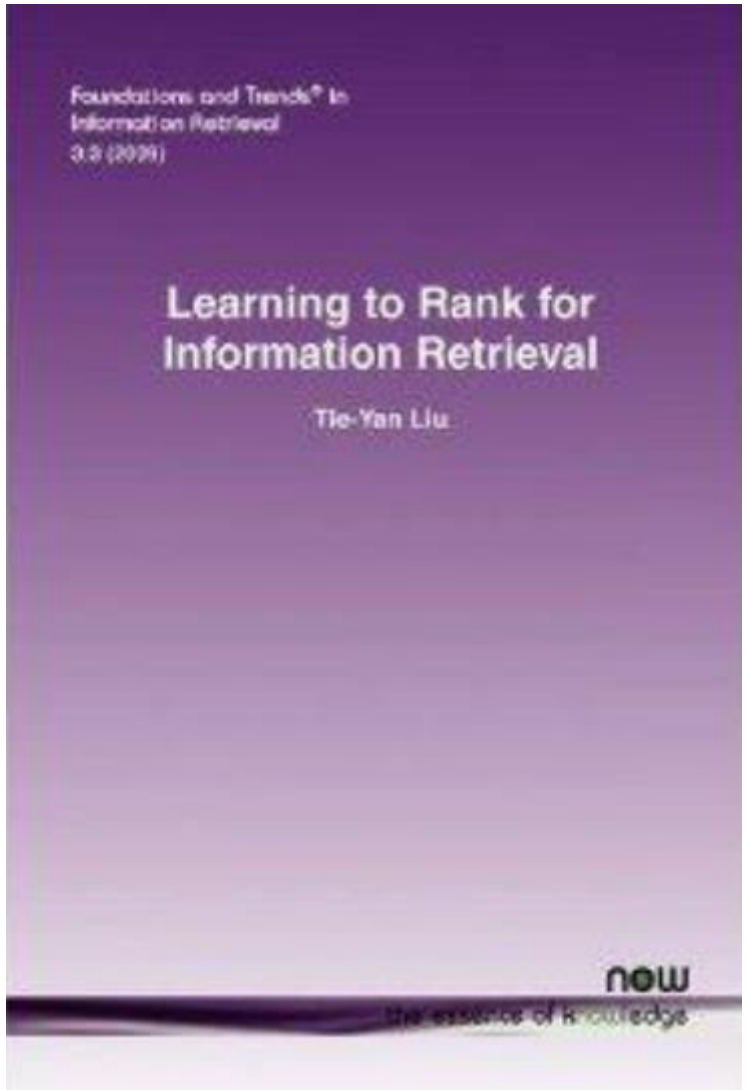
LTR Libraries

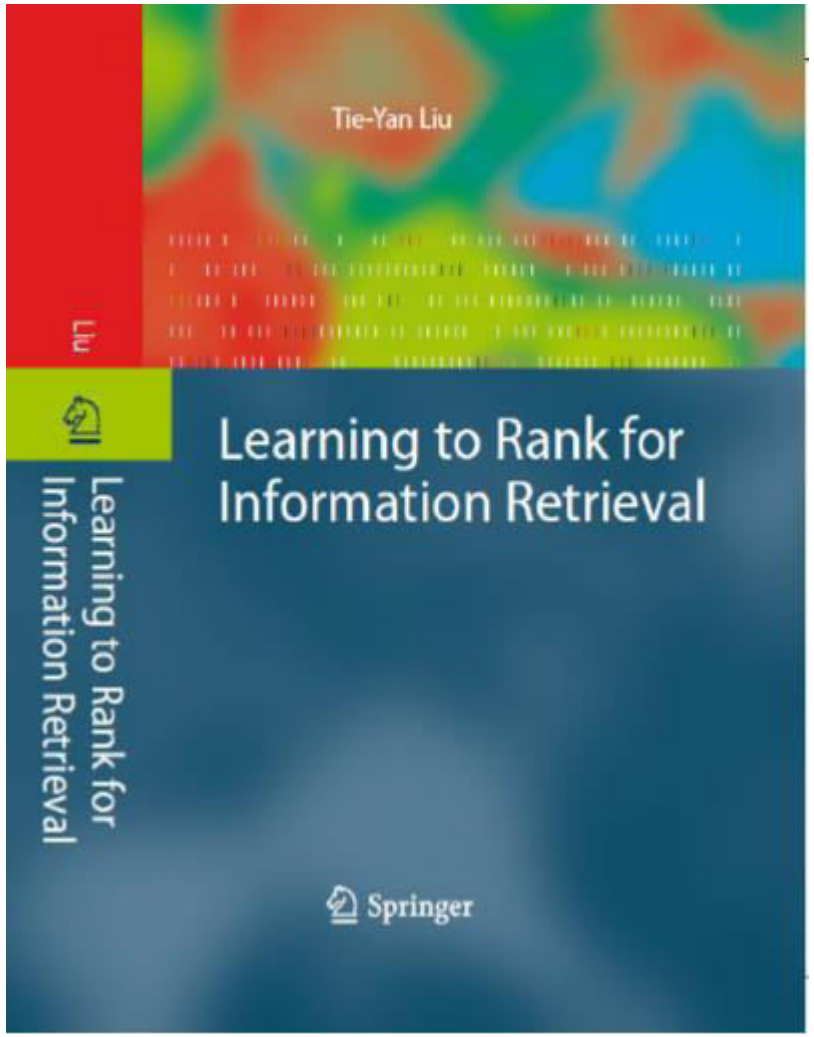
- RankLib (Java) – from Lemur project (UMASS, CMU)
  - Coordinate Ascent, Random Forest (pointwise), MART, RankNet, RankBoost (pairwise), LambdaMART (pair/listwise), AdaRank and ListNet (listwise)
- SVMRank (C++) – from Cornell, provides SVMRank (pairwise)
- XGBoost (Python/C++) – LambdaRank (pairwise)
- PyLTR (Python) – LambdaMart (pairwise)
- Michael Alcorn (Python) – RankNet and LambdaMART (pairwise)

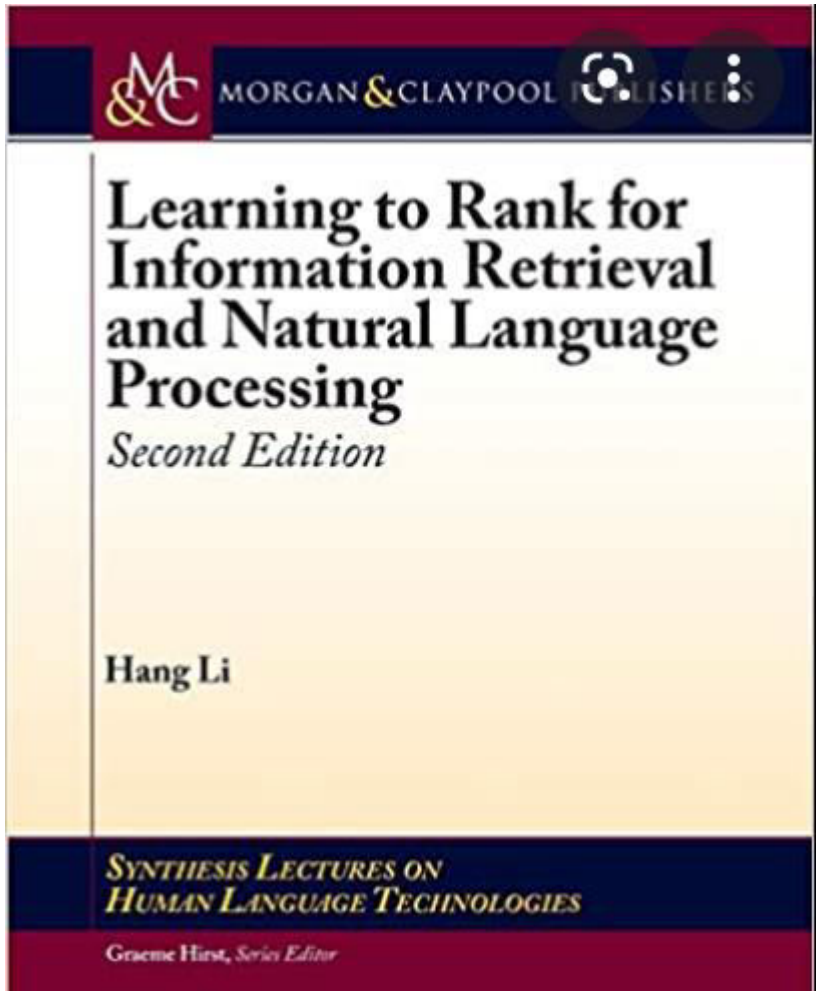
Applications of Learning to Rank

- Web search
- Collaborative filtering
- Definition search
- Keyphrase extraction
- Query dependent summarization

References


Foundations and Trends® in Information Retrieval
3:3 (2009)
Learning to Rank for Information Retrieval
Tie-Yan Liu
now


Tie-Yan Liu
Liu
Learning to Rank for Information Retrieval
Learning to Rank for Information Retrieval
Springer


MORGAN & CLAYPOOL PUBLISHERS
Learning to Rank for Information Retrieval and Natural Language Processing
Second Edition
Hang Li
SYNTHESIS LECTURES ON HUMAN LANGUAGE TECHNOLOGIES
Graeme Hirst, Series Editor

Questions?