CCF ADL 2011
Beijing
Aug. 27, 2011

Learning to Rank

Hang Li
Microsoft Research Asia

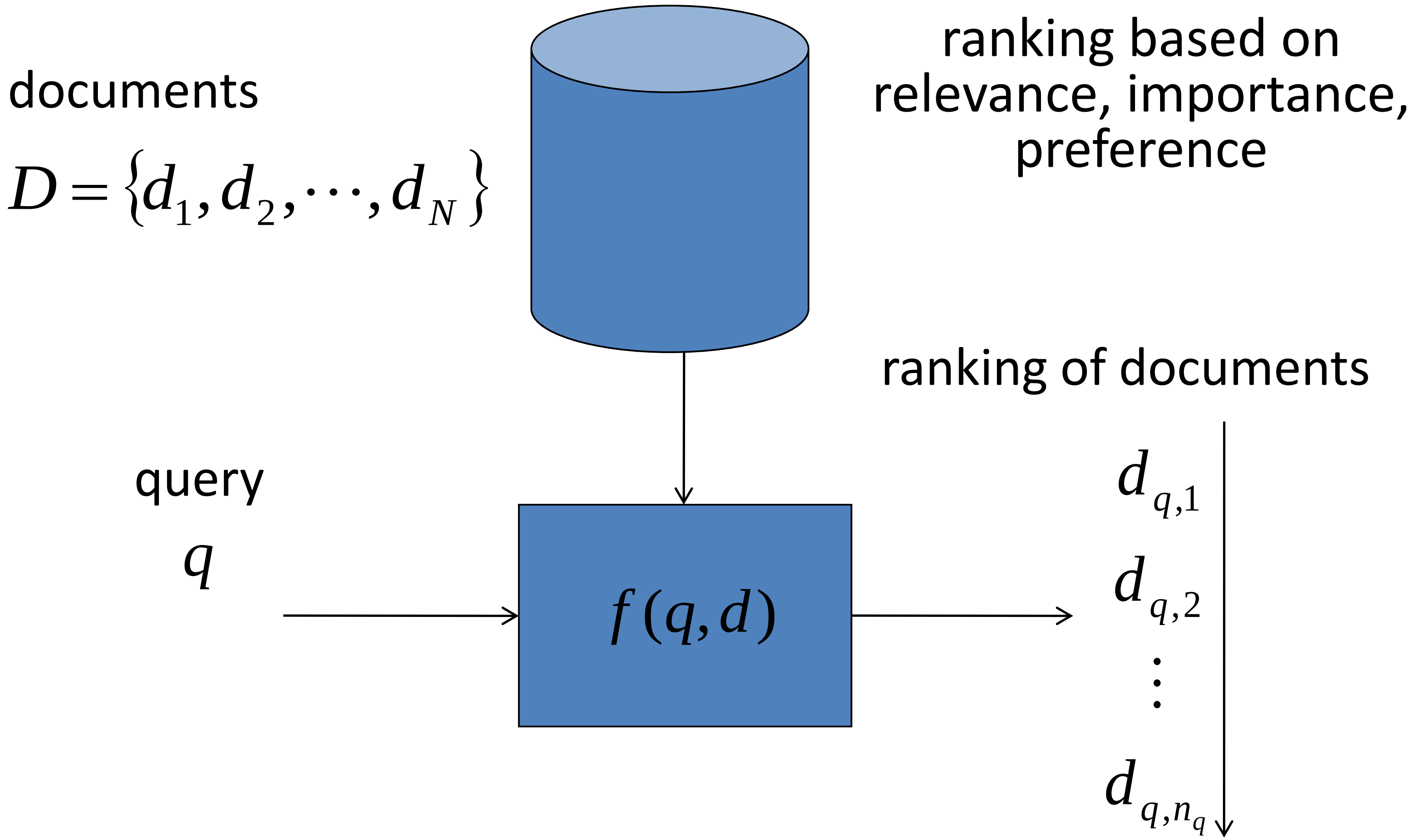
Outline of Tutorial

1. Learning to Rank
2. Learning for Ranking Creation
3. Learning for Ranking Aggregation
4. Methods of Learning to Rank
5. Applications of Learning to Rank
6. Theory of Learning to Rank
7. Ongoing and Future Work

1. Learning to Rank

Ranking Plays Key Role in Many Applications

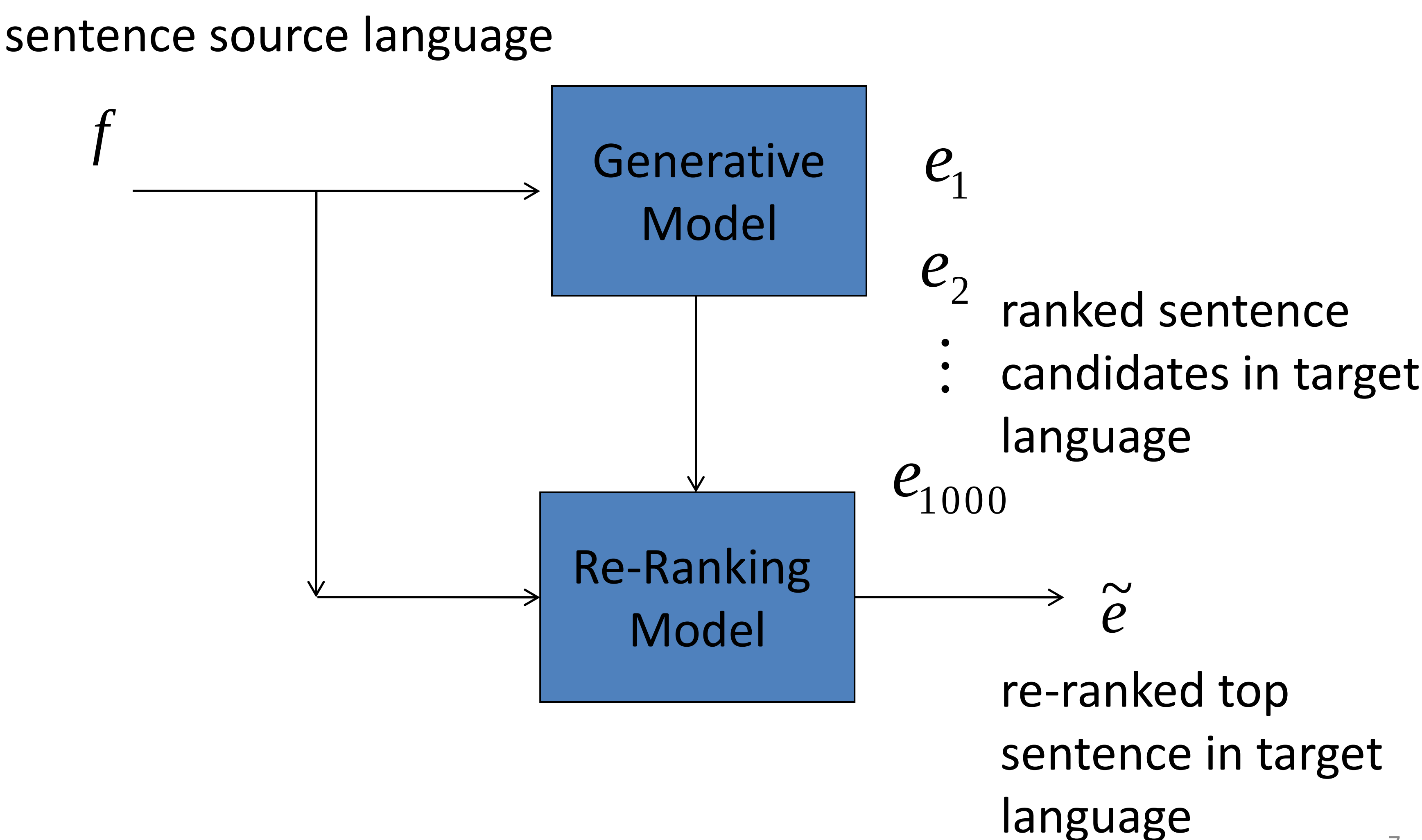
Ranking Problem:
Example = Document Search



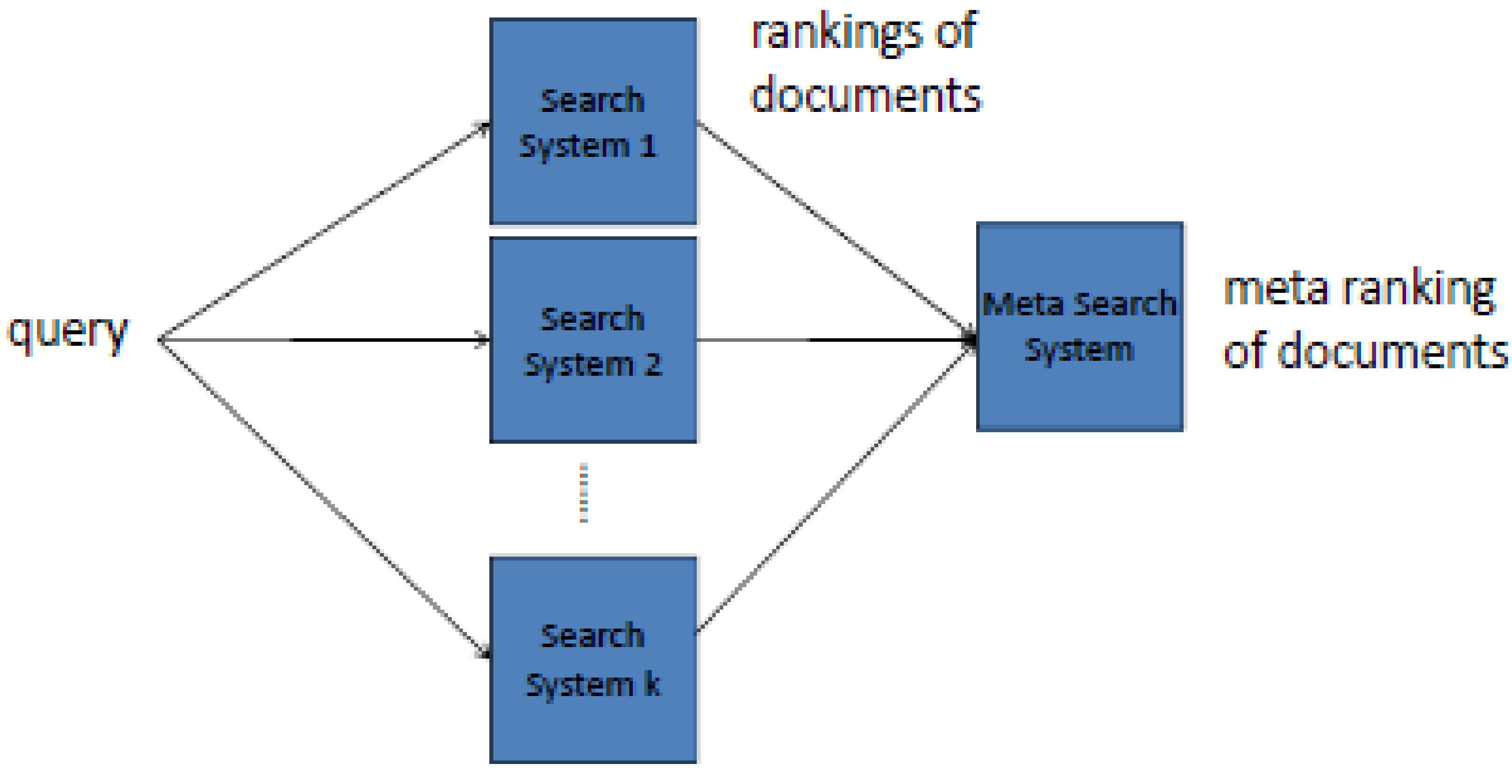
Ranking Problem
Example = Recommender System

	Item1	**Item2**	**Item3**	**...**	
User1	5	4			
User2	1		2		2
...		?	?	?	
UserM	4	3			

Ranking Problem
Example = Machine Translation



Ranking Problem
Example = Meta Search



Learning to Rank

- Definition 1 (in broad sense)

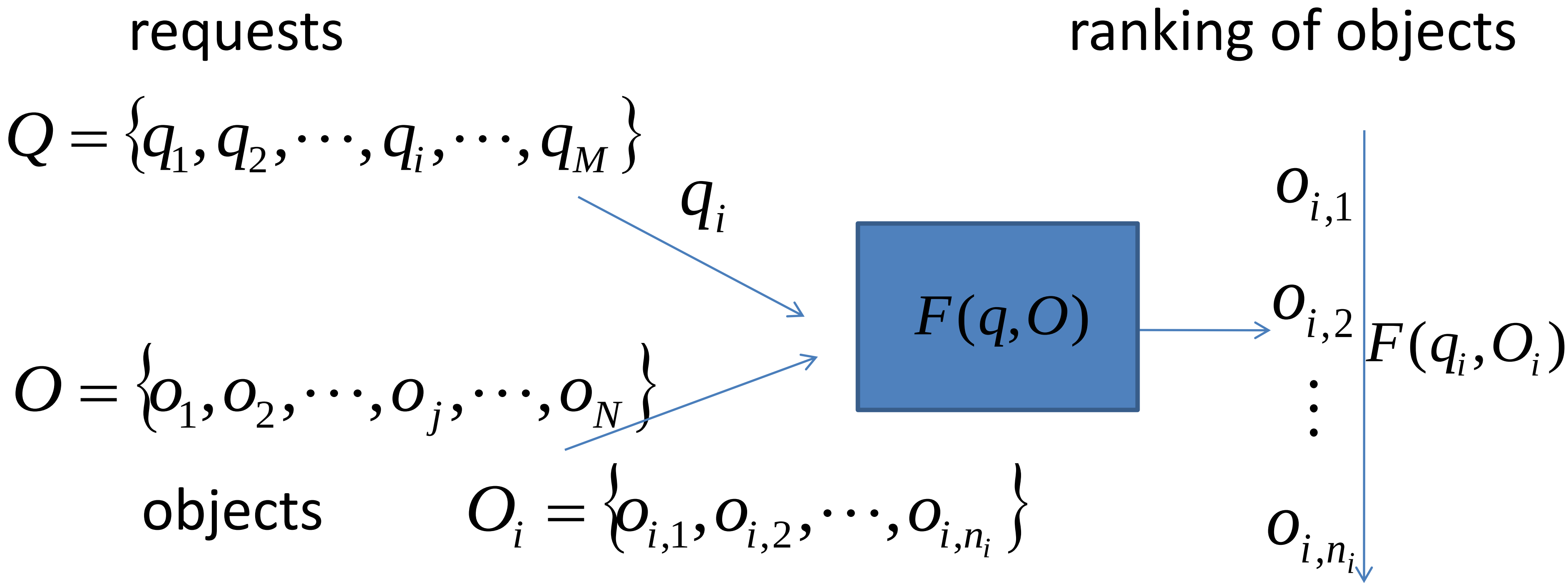
  Learning to rank = any machine learning technology for ranking problem
- Definition 2 (in narrow sense)

  Learning to rank = machine learning technology for ranking creation and ranking aggregation
- This tutorial takes Definition 2

Taxonomy of Problems in Learning to Rank

- Learing to Rank
  - Ranking Creation
    - Supervised (e.g., Ranking SVM)
    - Unsupervised (e.g., BM25)
  - Ranking Aggregation
    - Supervised (e.g., CRank)
    - Unsupervised (e.g., Borda Count)

Ranking Creation
(with Global Ranking Model)



Ranking Creation
(with Local Ranking Model)

requests

$$S = \{q_1, q_2, \cdots, q_i, \cdots, q_M\}$$

q_i

$$f(q, o)$$

ranking of objects

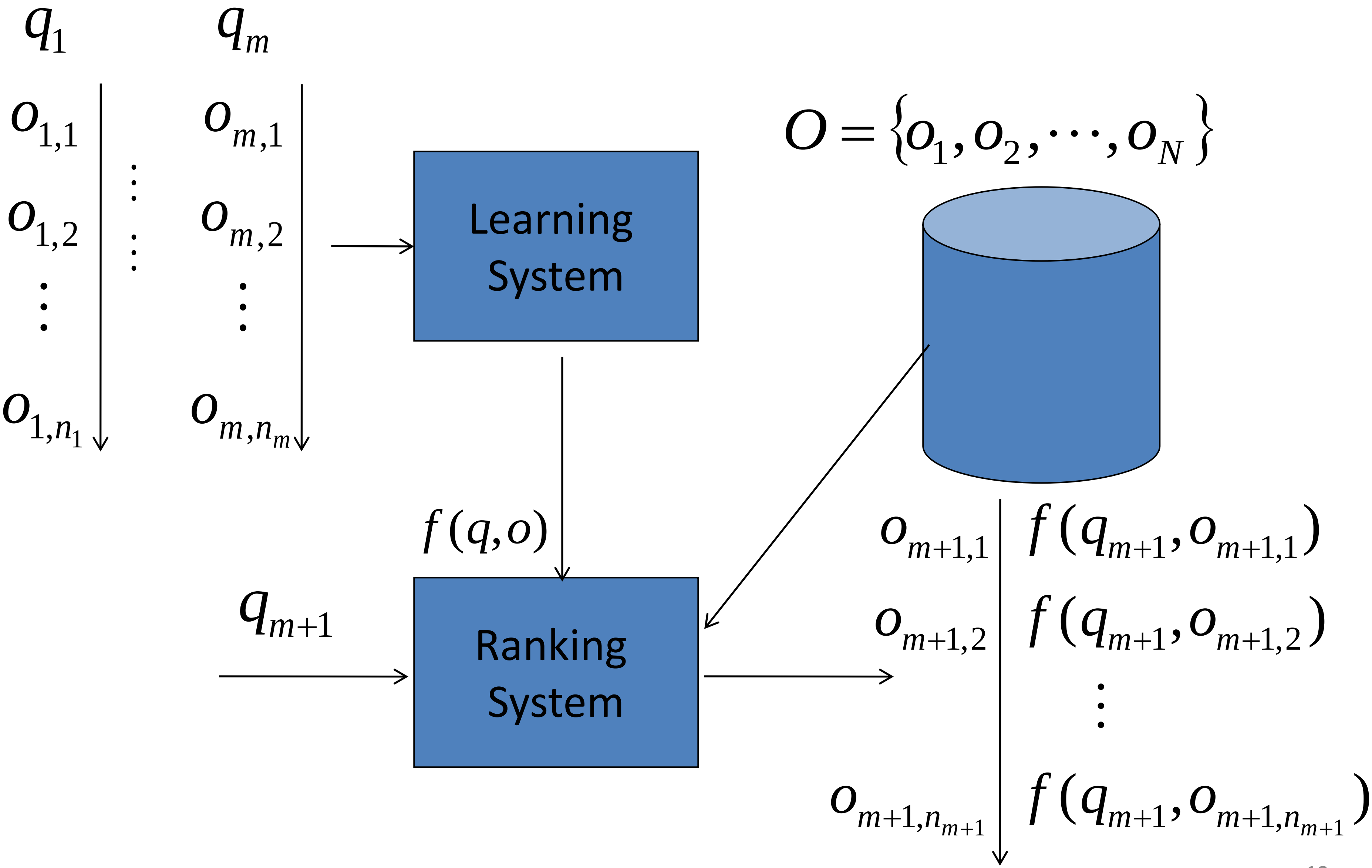
$$\begin{array}{l|l} o_{i,1} & f(q_i, o_{i,1}) \\ o_{i,2} & f(q_i, o_{i,2}) \\ & \vdots \\ o_{i,n_i} & f(q_i, o_{i,n_i}) \end{array}$$

$$O = \{o_1, o_2, \cdots, o_j, \cdots, o_N\}$$

objects

$$O_i = \{o_{i,1}, o_{i,2}, \cdots, o_{i,n_i}\}$$

Learning for Ranking Creation



Model in Ranking Creation

- Global ranking model

$$S_O = F(q, O)$$

$$\pi = \mathrm{sort}_{S_O}(O),$$

- Local ranking model

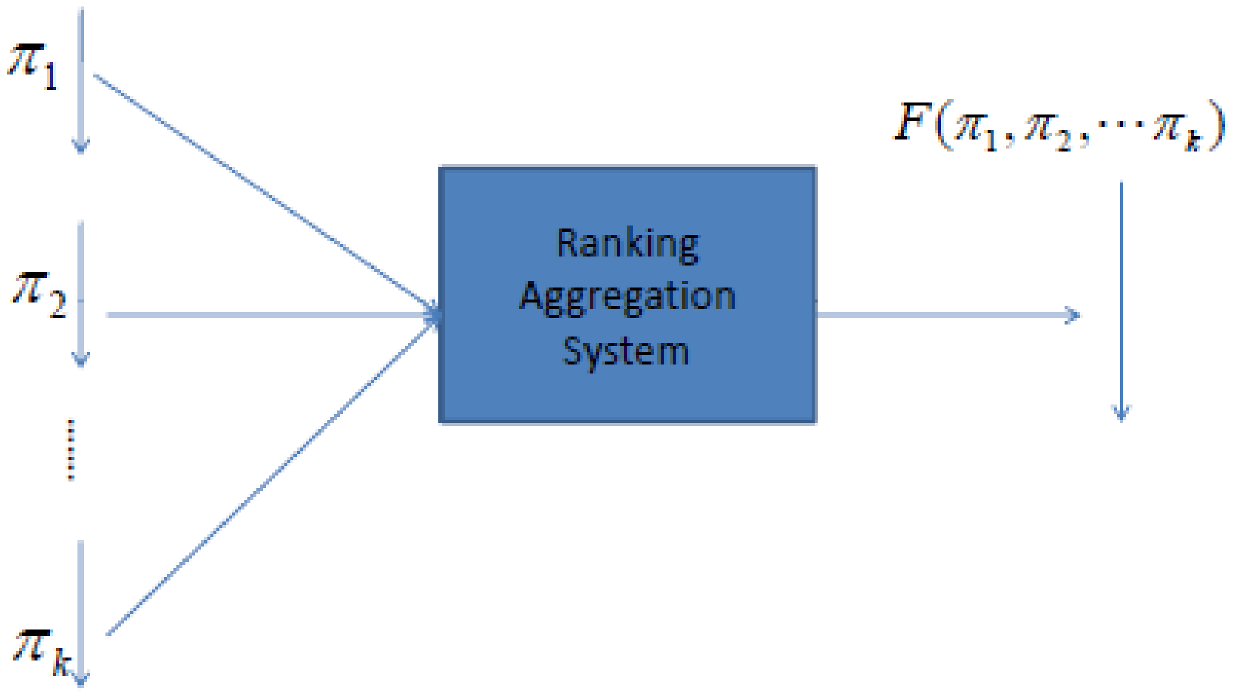
$$s_o = f(q, o)$$

$$\pi = \mathrm{sort}_{s_o, o \in O}(O).$$

Learning for Ranking Creation

- Creating a ranking list of offerings based on request and offerings
- Feature-based
- Usually local ranking model
- Usually supervised learning

Ranking Aggregation



Model in Ranking Aggregation

- Global ranking

$$S_O = F(q, \Sigma)$$
$$\pi = sort_{S_O}(O)$$

Learning for Ranking Aggregation

- Aggregating a ranking list from multiple ranking lists of offerings
- Ranking based
- Usually global ranking model
- Both supervised and unsupervised learning

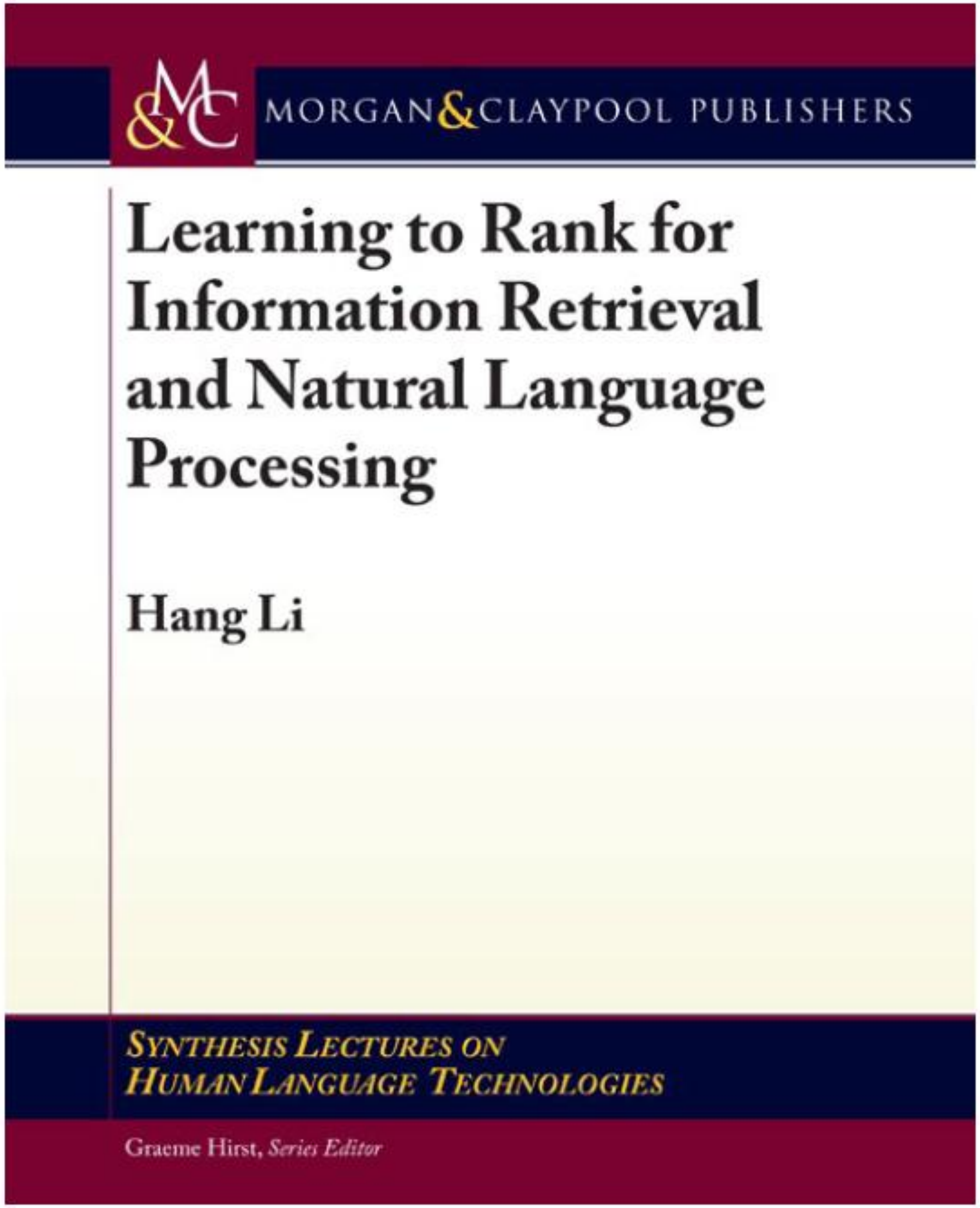
Technologies on Learning to Rank

- Methods
  - Pointwise Methods
  - Pairwise Methods
  - Listwise Methods
- Theory
  - Generalization
  - Consistency
- Applications
  - Search
  - Collaborative Filtering
  - Key Phrase Extraction

Recent Trends on Learning to Rank

- Successfully applied to web search
- Over 100 publications at SIGIR, ICML, NIPS, etc
- One book on Learning to rank for information retrieval
- 2 sessions at SIGIR every year
- 3 SIGIR workshops
- Special issue at Information Retrieval Journal
- Yahoo Learning to rank challenge
- LETOR benchmark dataset

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


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

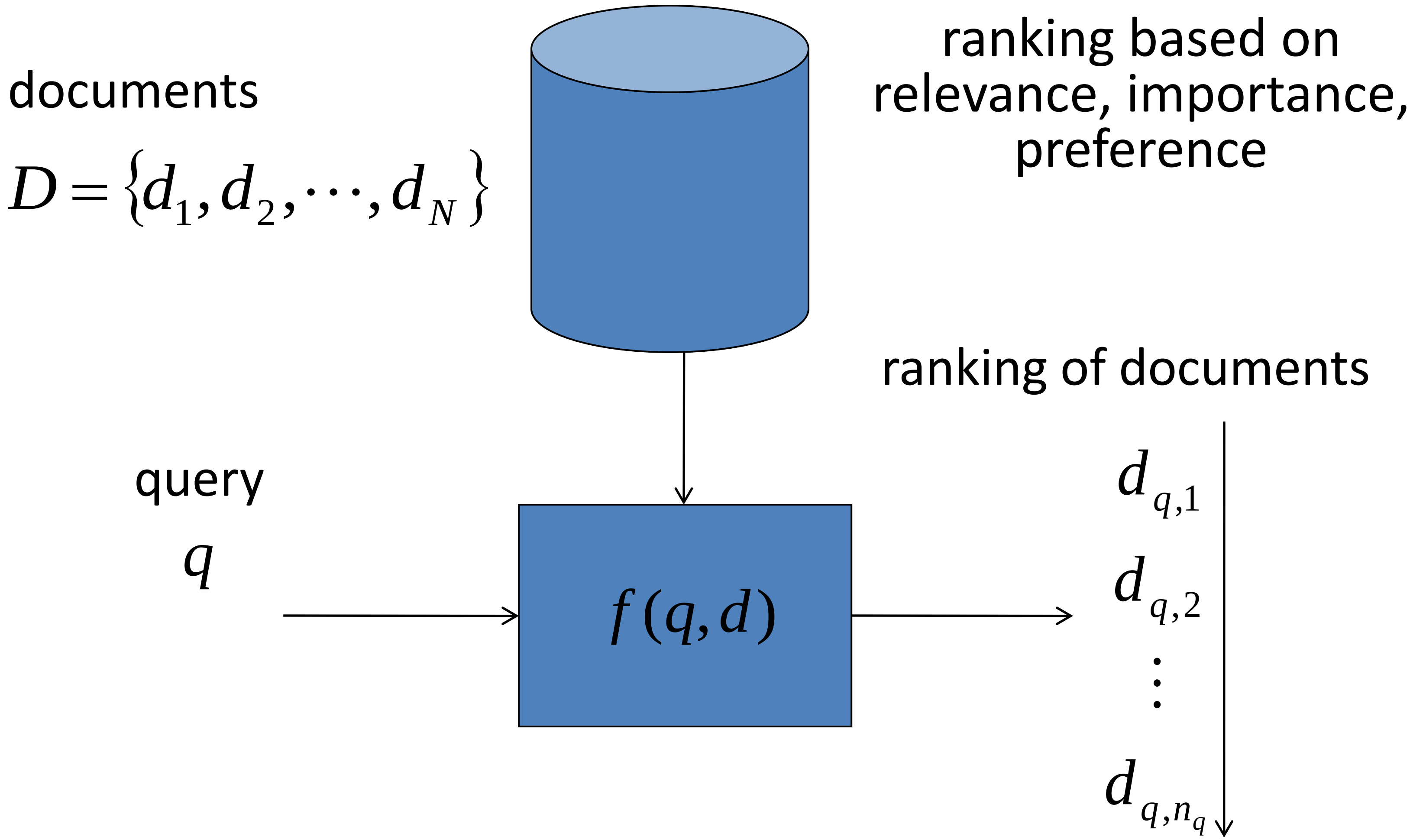
Scope of This Tutorial

- Overview of Learning to rank technologies
- Focusing on Learning to rank methods
- Touching theoretical issues
- Showing future directions
- Knowledge necessary for this tutorial: Machine Learning

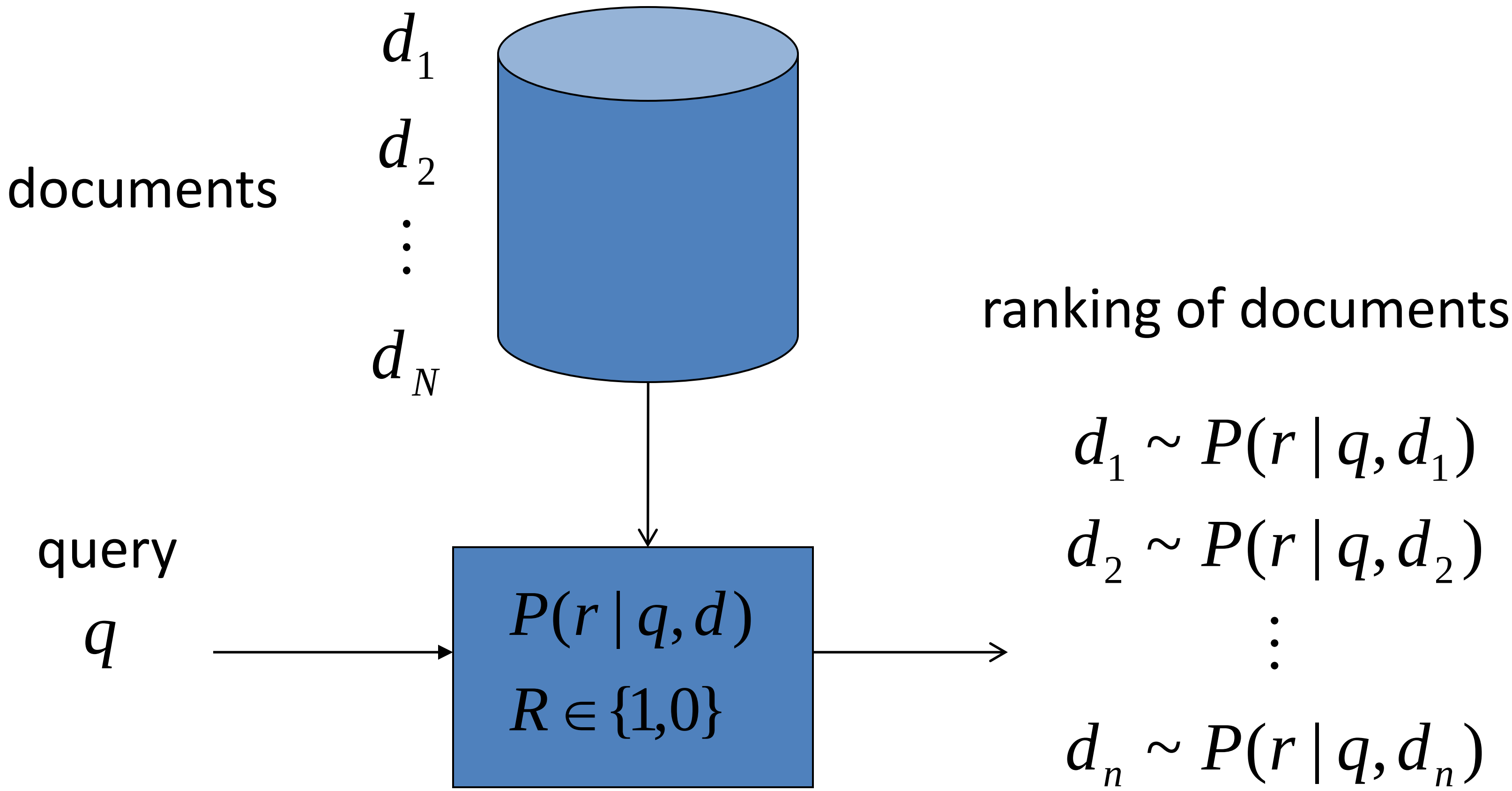
2. Learning for Ranking Creation

2.1 Document Retrieval as Example

Ranking Problem: Example = Document Search



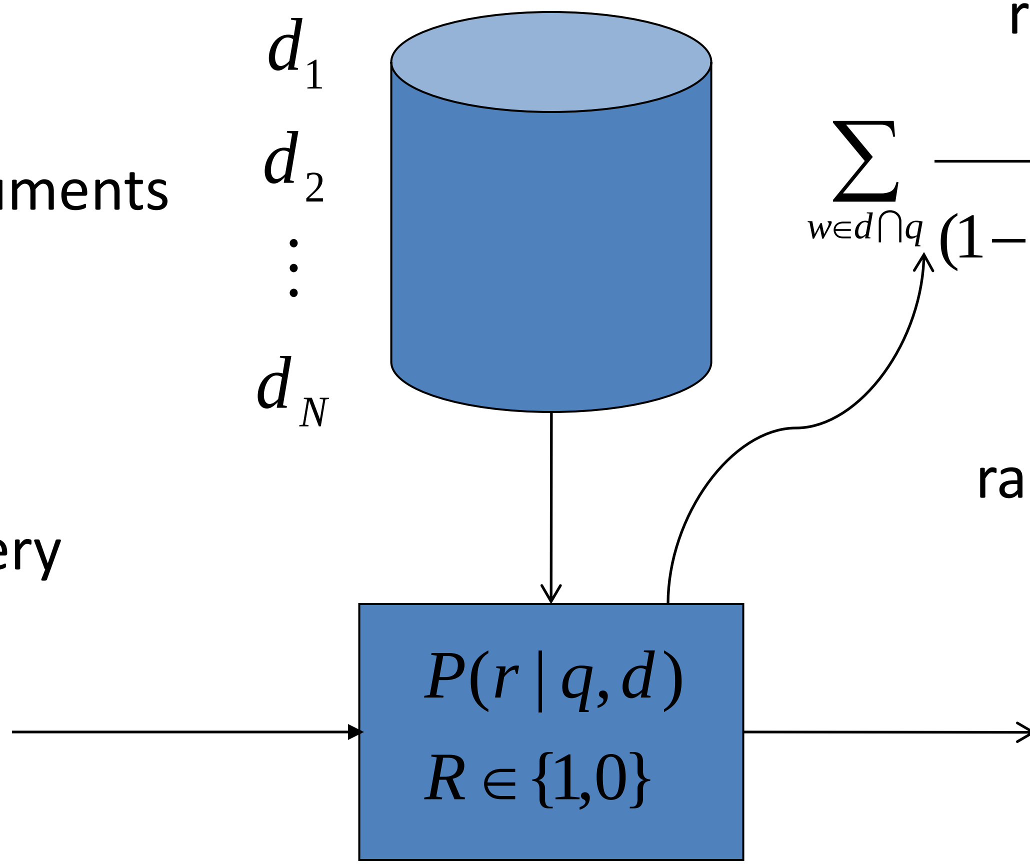
Traditional Approach = Probabilistic Model



BM25
[Robertson & Walker 94]

documents

query
q



ranking function

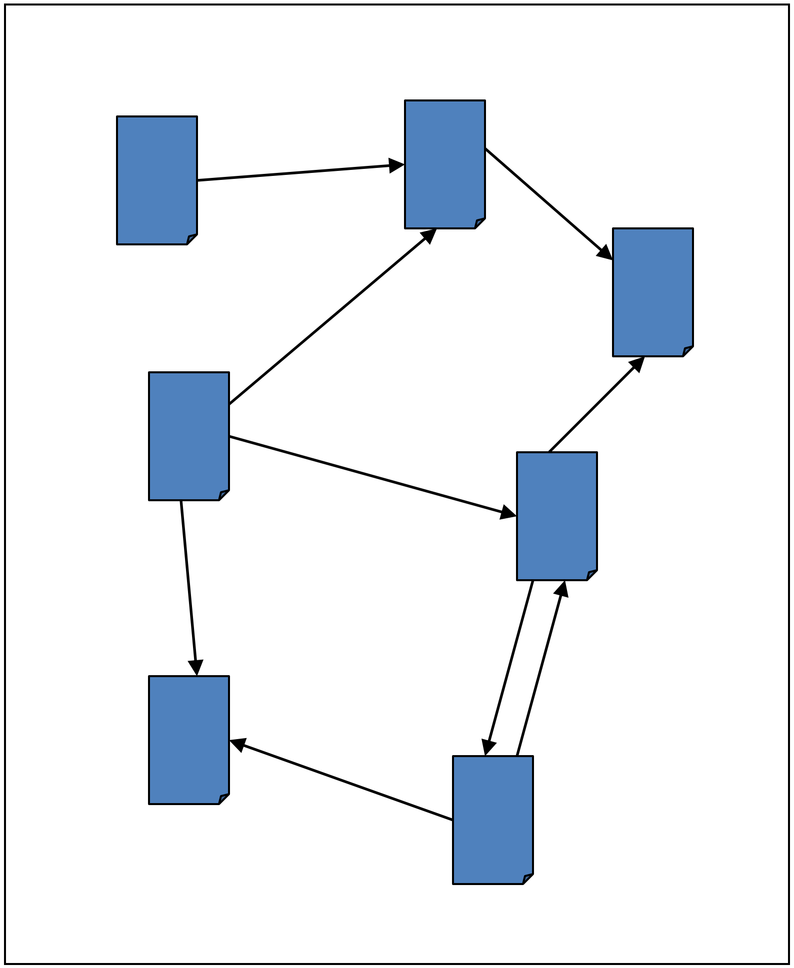
$$\sum_{w \in d \cap q} \frac{(k+1)tf(w)}{(1-b)k + b\dfrac{dl}{avgdl} + tf(w)}$$

ranking of documents

$$d_1 \sim P(r \mid q, d_1)$$
$$d_2 \sim P(r \mid q, d_2)$$
$$\vdots$$
$$d_n \sim P(r \mid q, d_n)$$

PageRank

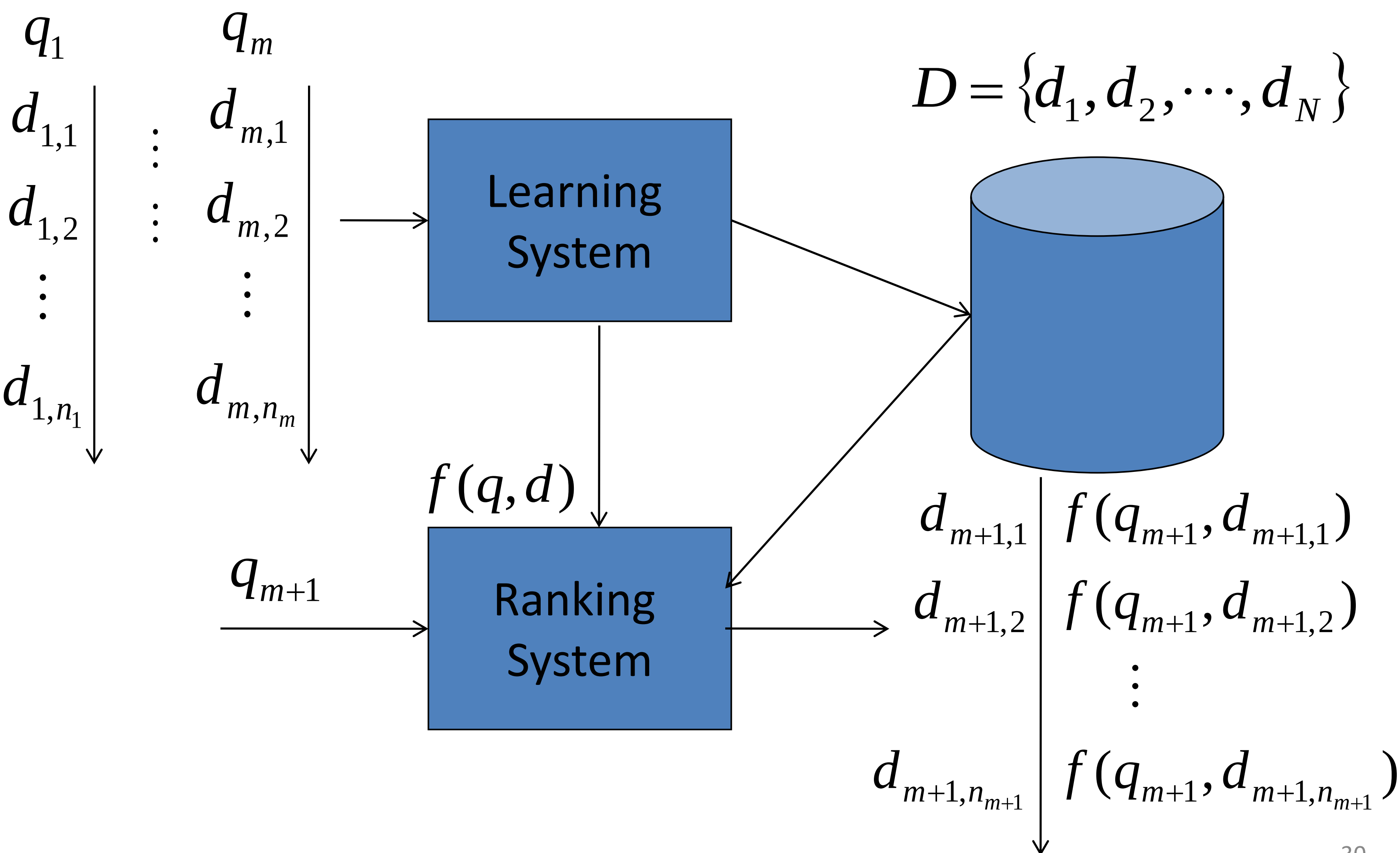
[Page et al, 1999]



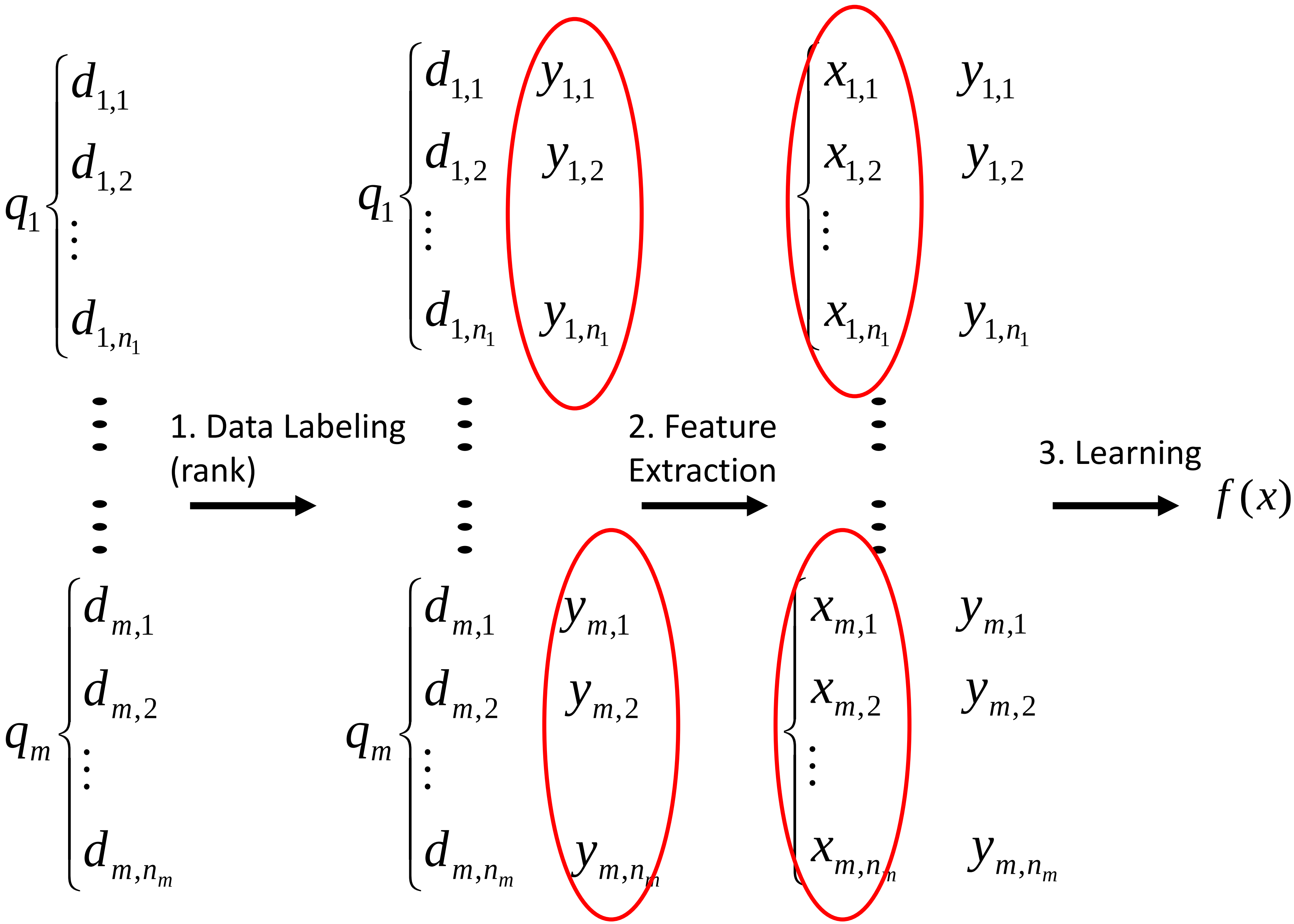
$$P(d_i) = \alpha \sum_{d_j \in M(d_i)} \frac{P(d_j)}{L(d_j)} + (1-\alpha)\frac{1}{n}$$

2.1 Learning Task

New Approach = Learning to Rank



Training Process



Testing Process

$$q_{m+1} \begin{cases} d_{m+1,1} \\ d_{m+1,2} \\ \vdots \\ d_{m+1,n_{m+1}} \end{cases}$$

1. Data Labeling (rank)

$$q_{m+1} \begin{cases} d_{m+1,1} & y_{m+1,1} \\ d_{m+1,2} & y_{m+1,2} \\ \vdots & \\ d_{m+1,n_{m+1}} & y_{m+1,n_{m+1}} \end{cases}$$

2. Feature Extraction

$$\begin{cases} x_{m+1,1} & y_{m+1,1} \\ x_{m+1,2} & y_{m+1,2} \\ \vdots & \\ x_{m+1,n_{m+1}} & y_{m+1,n_{m+1}} \end{cases}$$

3. Ranking with $f(x)$

$$\begin{cases} x_{m+1,1} & f(x_{m+1,1}) & y_{m+1,1} \\ x_{m+1,2} & f(x_{m+1,2}) & y_{m+1,2} \\ \vdots & & \\ x_{m+1,n_{m+1}} & f(x_{m+1,n_{m+1}}) & y_{m+1,n_{m+1}} \end{cases}$$

4. Evaluation

Evaluation Result

Notes

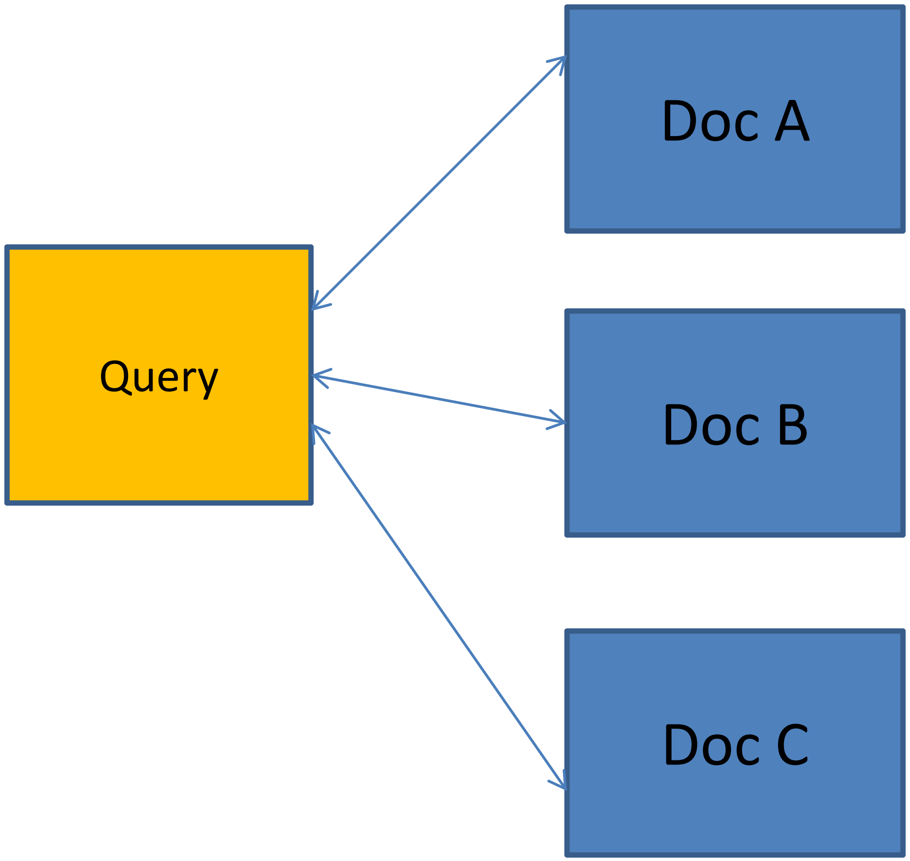
- Features are functions of query and document
- Query and associated documents form a group
- Groups are i.i.d. data
- Feature vectors within group are not i.i.d. data
- Ranking model is function of features
- Several data labeling methods (here labeling of grade)

Issues in Learning to Rank

- Data Labeling
- Feature Extraction
- Evaluation Measure
- Learning Method (Model, Loss Function, Algorithm)

Data Labeling Problem

- E.g., relevance of documents w.r.t. query

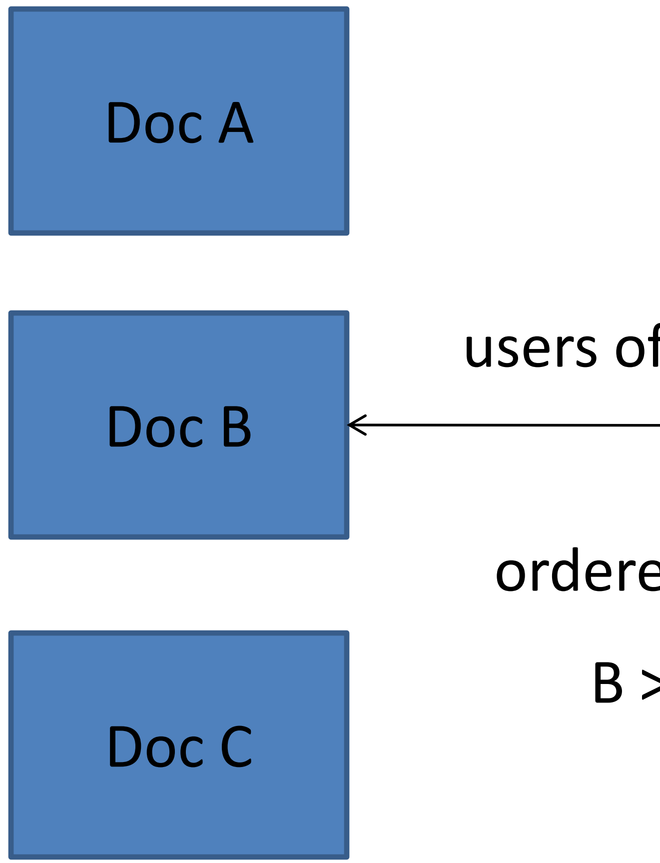


Data Labeling Methods

- Labeling of Grades
  - Multiple levels (e.g., relevant, partially relevant, irrelevant)
  - Widely used in IR
- Labeling of Ordered Pairs
  - Ordered pairs between documents (e.g. A>B, B>C)
  - Implicit relevance judgment: derived from click-through data
- Creation of List
  - List (or permutation) of documents is given
  - Ideal but difficult to implement

Implicit Relevance Judgment

ranking of documents at search system

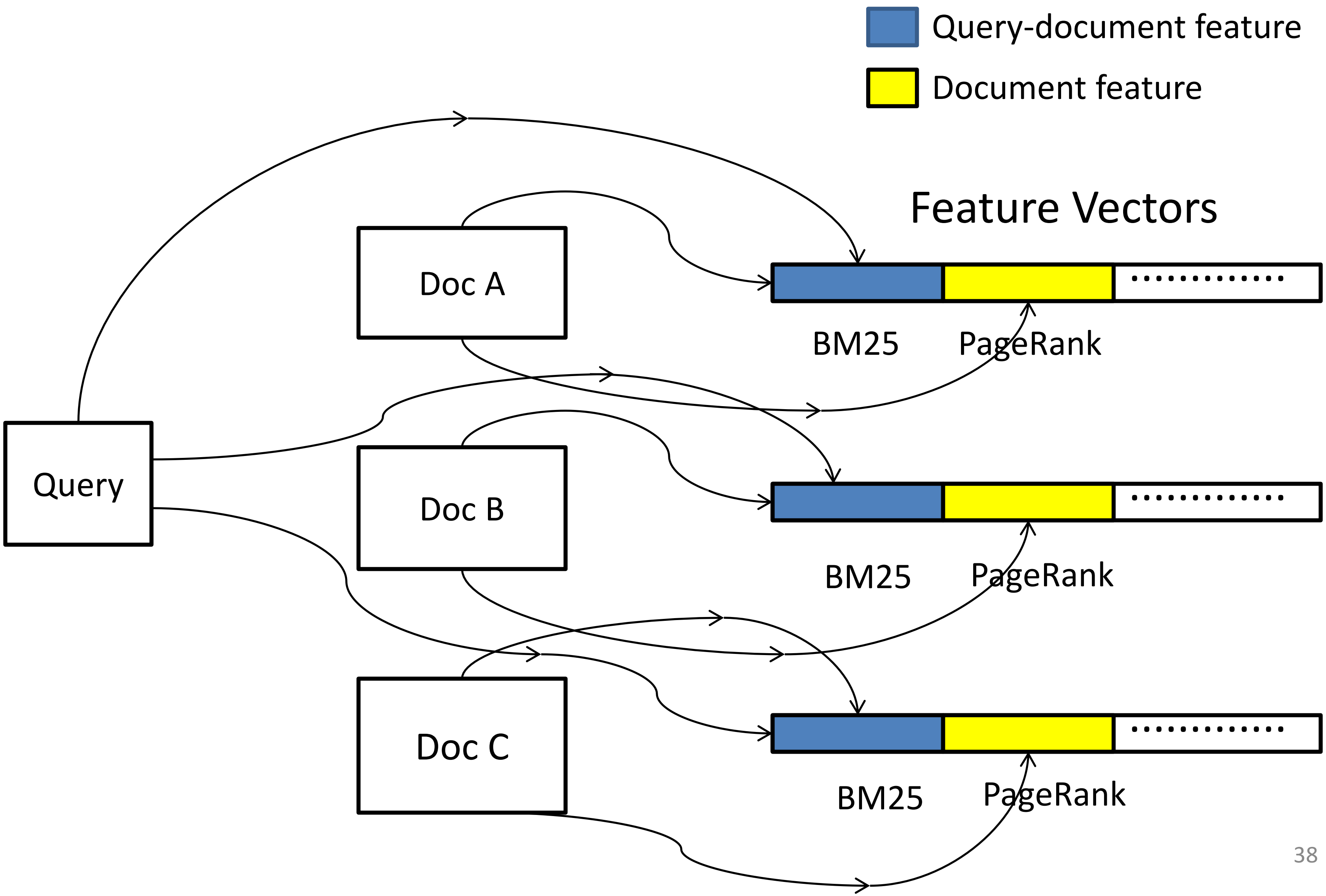


users often clicked on Doc B

ordered pair

B > A

Feature Extraction



Example Features

Table 2.3: Example Features of Learning to Rank for Web Search

Feature	Type	Explanation	Reference
Number of occurrences	Matching	number of times query exactly occurs in title, anchor, URL, extracted title, associated query, and body	
BM25	Matching	BM25 scores on title, anchor, URL, extracted title, associated query, and body	[90]
N-gram BM25	Matching	BM25 scores of n-grams on title, anchor, URL, extracted title, associated query, and body	[109]
Edit Distance	Matching	edit distance scores between query and title, anchor, URL, extracted title, associated query, and span in body (minimum length of text segment including all query words [94])	Our unpublished work
Number of in-links	Document	number of in-links to the page	
PageRank	Document	importance score of page calculated on web link graph	[78]
Number of clicks	Document	number of clicks on the page in search log	
BrowseRank	Document	importance score of page calculated on user browsing graph	[72]
Spam score	Document	likelihood of spam page	[45]
Page quality score	Document	likelihood of low quality page	[10]

Evaluation Measures

- Important to rank top results correctly
- Measures
  - NDCG (Normalized Discounted Cumulative Gain)
  - MAP (Mean Average Precision)
  - MRR (Mean Reciprocal Rank)
  - WTA (Winners Take All)
  - Kendall's Tau

NDCG

- Evaluating ranking using labeled grades
- NDCG at position *j*

$$\frac{1}{n_j}\sum_{i=1}^{j}(2^{r(i)}-1)/\log(1+i)$$

NDCG (cont')

- Example: perfect ranking
  - (3, 3, 2, 2, 1, 1, 1) grade *r=3,2,1*
  - (7, 7, 3, 3, 1, 1, 1) gain $2^{r(j)}-1$
  - (1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33) position discount $1/\log(1+j)$
  - (7, 11.41, 12.91, …) DCG

    $$\sum_{i=1}^{j}(2^{r(i)}-1)/\log(1+i)$$

  - (1/7, 1/11.41, 1/12.91, …) normalizing factor n_j
  - (1, 1,1,1,1,1,1) NDCG for perfect ranking

NDCG (cont')

- Example: imperfect ranking
  - (2, 3, 2, 3, 1, 1, 1)
  - (3, 7, 3, 7, 1, 1, 1) Gain
  - (1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33) Position discount
  - (3, 7.41, 8.91, …) DCG
  - (1/7, 1/11.41, 1/12.91, …) normalizing factor
  - (0.43, 0.65, 0.69, ….) NDCG

- Imperfect ranking decreases NDCG

MAP

- Evaluating ranking using two grades
- AP

$$AP = \frac{\sum_{j=1}^{n_i} P(j) \cdot y_{i,j}}{\sum_{j=1}^{n_i} y_{i,j}},$$

$$P(j) = \frac{\sum_{k:\pi_i(k) \leq \pi_i(j)} y_{i,k}}{\pi_i(j)},$$

MAP (cont')

- Example: perfect ranking
  - (1,0,1,1,0,0,0) grade *r=0,1*
  - (1, -, 0.67, 0.75, -, -, -) *P(j)* precision at position *j*
  - 0.81 *AP* average precision

Relations with Other Learning Tasks

- No need to predict category
  vs Classification
- No need to predict value of $f(q,d)$
  vs Regression
- Relative ranking order is more important
  vs Ordinal regression
- *Learning to rank can be approximated by classification, regression, ordinal regression*

Ordinal Regression (Ordinal Classification)

- Categories are ordered
  - 5, 4, 3, 2, 1
  - e.g., rating restaurants
- Prediction
  - Map to ordered categories

2.3 Learning Approaches

Three Major Approaches

- Pointwise approach
- Pairwise approach
- Listwise approach

- SVM based
- Boosting based
- Neural Network based
- Others

Categorization of Learning to rank Methods

Table 2.6: Categorization of Learning to Rank Methods

	SVM	Boosting	Neural Net	Others
Pointwise	OC SVM [92]	McRank [67]		Prank [30] Subset Ranking [29]
Pairwise	Ranking SVM [48] IR SVM [13]	RankBoost [37] GBRank [115] LambdaMART [102]	RankNet [11] Frank [97] LambdaRank [12]	
Listwise	SVM MAP [111] PermuRank [110]	AdaRank [108]	ListNet [14] ListMLE [104]	SoftRank [95] AppRank [81]

Pointwise Approach

- Transforming ranking to regression, classification, or ordinal classification
- Query-document group structure is ignored

Pointwise Approach

Table 2.7: Characteristics of Pointwise Approach

Pointwise Approach (Classification)		
	Learning	**Ranking**
Input	feature vector x	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^{n}$
Output	category $y = \text{classifier}(f(x))$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^{n})$
Model	$\text{classifier}(f(x))$	ranking model $f(x)$
Loss	classification loss	ranking loss
Pointwise Approach (Regression)		
	Learning	**Ranking**
Input	feature vector x	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^{n}$
Output	real number $y = f(x)$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^{n})$
Model	regression model $f(x)$	ranking model $f(x)$
Loss	regression loss	ranking loss

Pointwise Approach

Pointwise Approach (Ordinal Classification)		
	Learning	**Ranking**
Input	feature vector	feature vectors
	x	$\mathbf{x} = \{x_i\}_{i=1}^{n}$
Output	ordered category	ranking list
	$y = \text{threshold}(f(x))$	$\text{sort}(\{f(x_i)\}_{i=1}^{n})$
Model	$\text{threshold}(f(x))$	ranking model $f(x)$
Loss	ordinal classification loss	ranking loss

Pairwise Approach

- Transforming ranking to pairwise classification
- Query-document group structure is ignored

Pairwise Approach

Table 2.8: Characteristics of Pairwise Approach

	Learning	Ranking
Pairwise Approach (Classification)		
	Learning	**Ranking**
Input	feature vectors $x^{(1)}, x^{(2)}$	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	pairwise classification $\text{classifier}(f(x^{(1)}) - f(x^{(2)}))$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	$\text{classifier}(f(x))$	ranking model $f(x)$
Loss	pairwise classification loss	ranking loss
Pairwise Approach (Regression)		
	Learning	**Ranking**
Input	feature vectors $x^{(1)}, x^{(2)}$	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^n$
Output	pairwise regression $f(x^{(1)}) - f(x^{(2)})$	ranking list $\text{sort}(\{f(x_i)\}_{i=1}^n)$
Model	regression model $f(x)$	ranking model $f(x)$
Loss	pairwise regression loss	ranking loss

Listwise Approach

- List as instance
- Query-document group structure is used
- Straightforwardly represents learning to rank problem

Listwise Approach

Table 2.9: Characteristics of Listwise Approach

Listwise Approach		
	Learning	**Ranking**
Input	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^{n}$	feature vectors $\mathbf{x} = \{x_i\}_{i=1}^{n}$
Output	ranking list $\mathrm{sort}(\{f(x_i)\}_{i=1}^{n})$	ranking list $\mathrm{sort}(\{f(x_i)\}_{i=1}^{n})$
Model	ranking model $f(x)$	ranking model $f(x)$
Loss	listwise loss function	ranking loss

Learning to rank Methods

- Pointwise Approach
  - Subset Ranking [Cossock and Zhang, 2006]: Regression
  - McRank [Li et al 2007]: Multi-Class Classification Using Boosting Tree
  - PRank [Crammer and Singer 2002]: Ordinal Classification Using Perceptron
  - OC SVM [Shashua & Levin 2002]: Ordinal Classification Using SVM

Learning to rank Methods

- Pairwise Approach
  - Ranking SVM: Pairwise Classification Using SVM
  - RankBoost [Freund et al 2003]: Pairwise Classification Using Boosting
  - RankNet [Burges et al 2005]: Pairwise Classification Using Neural Net
  - Frank [Tsai et al 2007]: Pairwise Classification Using Fidelity Loss and Neural Net
  - GBRank [Zheng et al 2007]: Pairwise Regression Using Boosting Tree
  - IR SVM [Cao et al 2006]: Cost-sensitive Pairwise Classification Using SVM
  - LambdaRank [Burges et al 2007]: Using Implicit Loss Function
  - LambdaMART [Wu et al 2010]: Using Implicit Loss Function

Learning to rank Methods

- Listwise Approach
  - ListNet [Cao et al 2007]: Probabilistic Ranking Model
  - ListMLE [Xia et al 2008]: Probabilistic Ranking Model
  - AdaRank [Xu and Li 2007]: Direct Optimization of Evaluation Measure
  - SVM Map [Yue et al 2007]: Direct Optimization of Evaluation Measure (Using Structure SVM)
  - PermuRank [Xu et al 2008]: Direct Optimization of Evaluation Measure
  - Soft Rank [Taylor et al 2008]: Approximation of Evaluation Measure
  - AppRank [Qin et al 2010]: Approximation of Evaluation Measure

LETOR Data Set

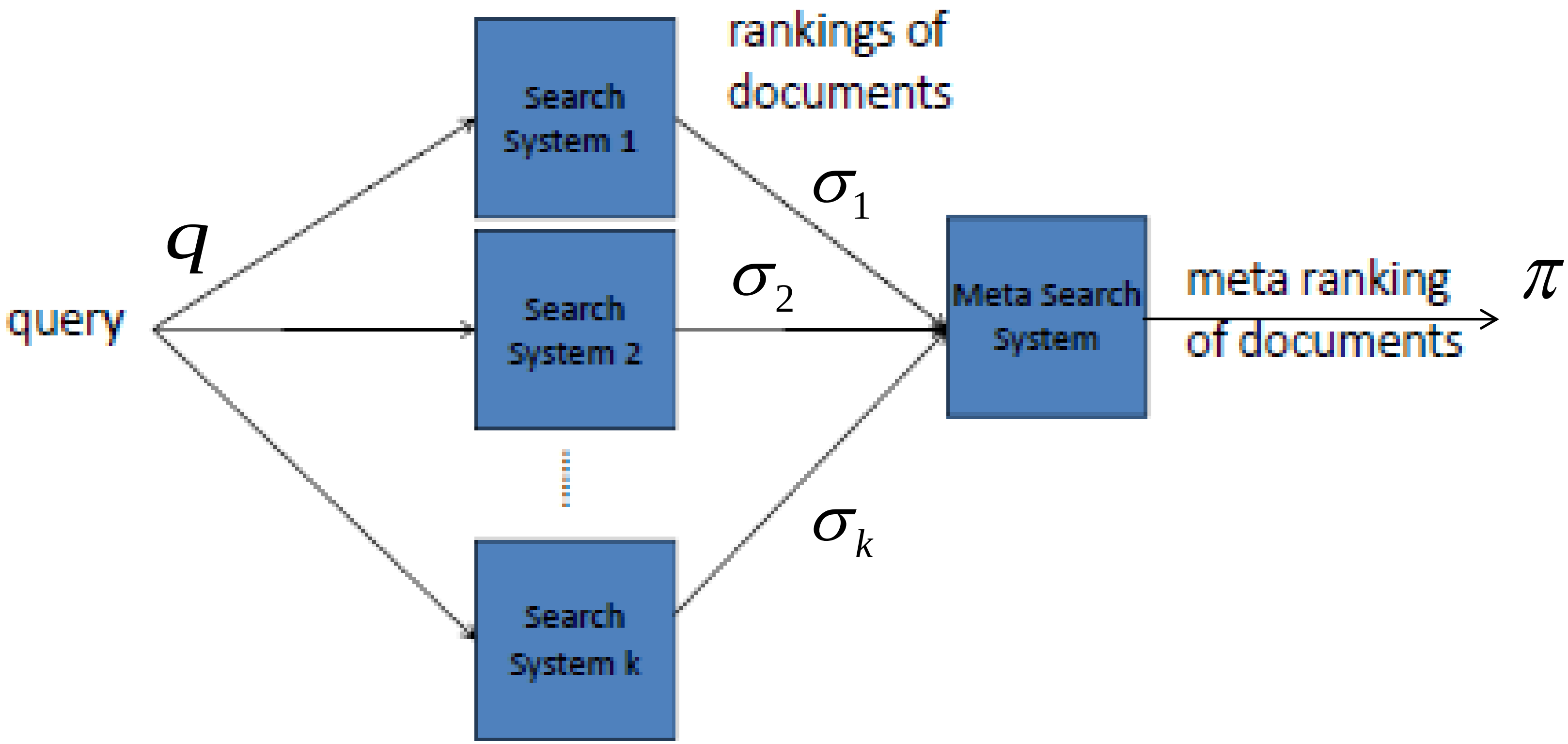
- Available at
  - http://research.microsoft.com/~letor/
- Data Corpora: TREC, OHSUMED
- Training/Validation/Test split
- Standard IR Features

Evaluation Results

- Pairwise approach and listwise approach perform better than pointwise approach
- LabmdaMART performs best in Yahoo Learning to rank Challenge
- No significant difference among pairwise and listwise methods

3. Learning for Ranking Aggregation

Ranking Problem
Example = Meta Search



Training Process

$$q_1 \begin{cases} \sigma_{1,1} \\ \sigma_{1,2} \\ \vdots \\ \sigma_{1,k} \end{cases}$$

$$\vdots$$

$$q_m \begin{cases} \sigma_{m,1} \\ \sigma_{m,2} \\ \vdots \\ \sigma_{m,k} \end{cases}$$

1. Data Labeling (ranking) $\longrightarrow$

$$q_1 \begin{cases} \sigma_{1,1} \\ \sigma_{1,2} \\ \vdots \\ \sigma_{1,k} \end{cases} \quad \pi_1$$

$$\vdots$$

$$q_m \begin{cases} \sigma_{m,1} \\ \sigma_{m,2} \\ \vdots \\ \sigma_{m,k} \end{cases} \quad \pi_m$$

2. Learning $\longrightarrow$

$$F(q, \Sigma)$$

$$F(\Sigma)$$

Testing Process

$$q_{m+1}\begin{cases}\sigma_{m+1,1}\\ \sigma_{m+1,2}\\ \vdots\\ \sigma_{m+1,k}\end{cases}$$

1. Data Labeling (ranking) $\longrightarrow$

$$q_{m+1}\begin{cases}\sigma_{m+1,1}\\ \sigma_{m+1,2}\\ \vdots\\ \sigma_{m+1,k}\end{cases} \quad \pi_{m+1}$$

2. Sort with $F(q,\Sigma)$ obtaining $\bar{\pi}_{m+1}$ $\longrightarrow$

$$q_{m+1} : \pi_{m+1}, \bar{\pi}_{m+1}$$

3. Evaluation $\longrightarrow$

Evaluation Result

Learning Methods

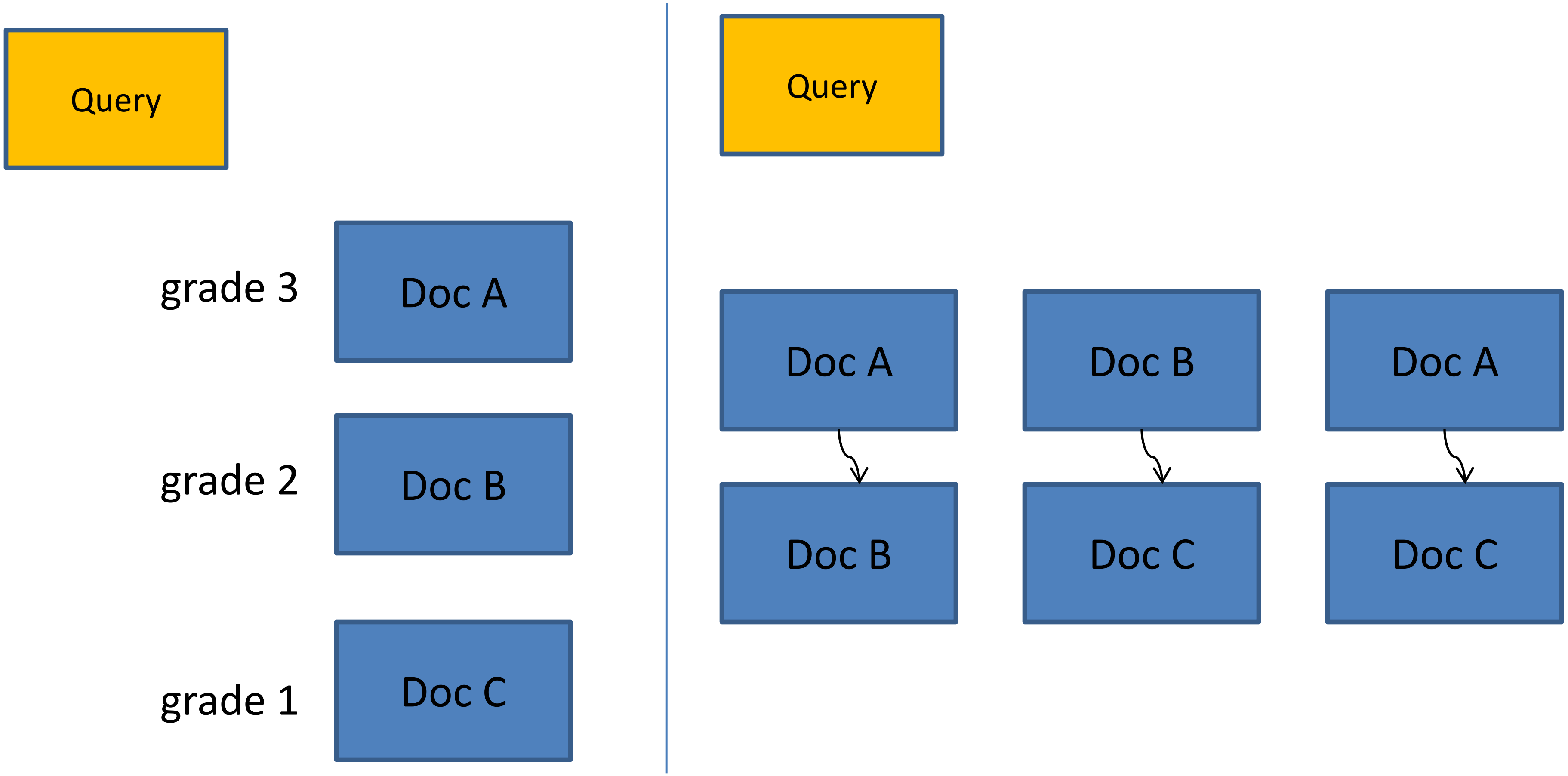
- Unsupervised Learning
  - Borda Count [Aslam & Montague 2001]
  - Markov Chain [Dwork et al 2001]
- Supervised learning
  - CRanking [Lebanon & Lafferty 2002]

4. Learning to rank Methods

Ranking SVM

Pairwise Classification

- Converting document list to document pairs



Transforming Ranking to Pairwise Classification

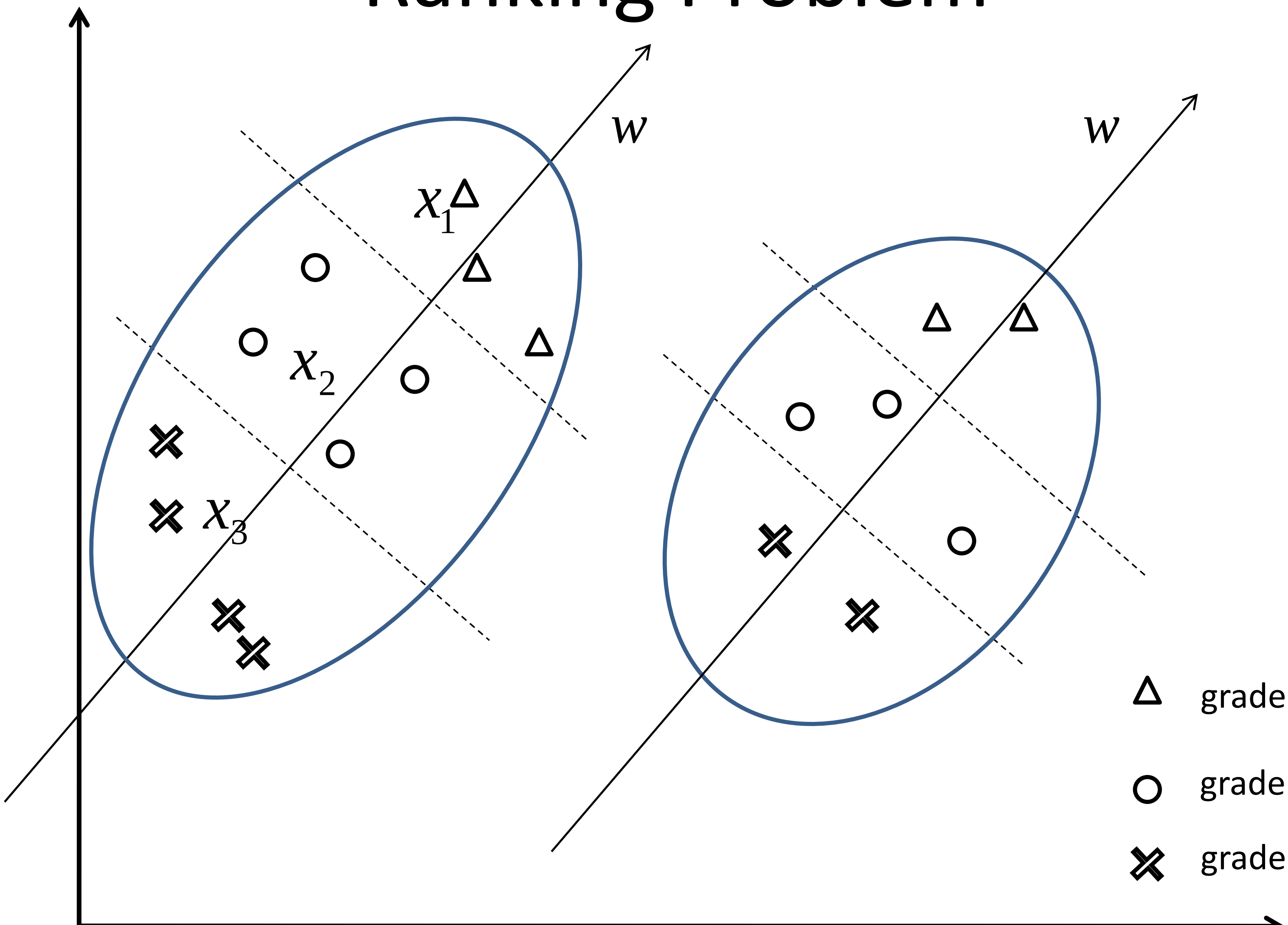
- Input space: *X*
- Ranking function $f : X \rightarrow R$
- Ranking: $x_i \succ x_j \iff f(x_i; w) > f(x_j; w)$
- Linear ranking function: $f(x; w) = \langle w, x \rangle$

$$\langle w, x_i - x_j \rangle > 0 \iff f(x_i; w) > f(x_j; w)$$

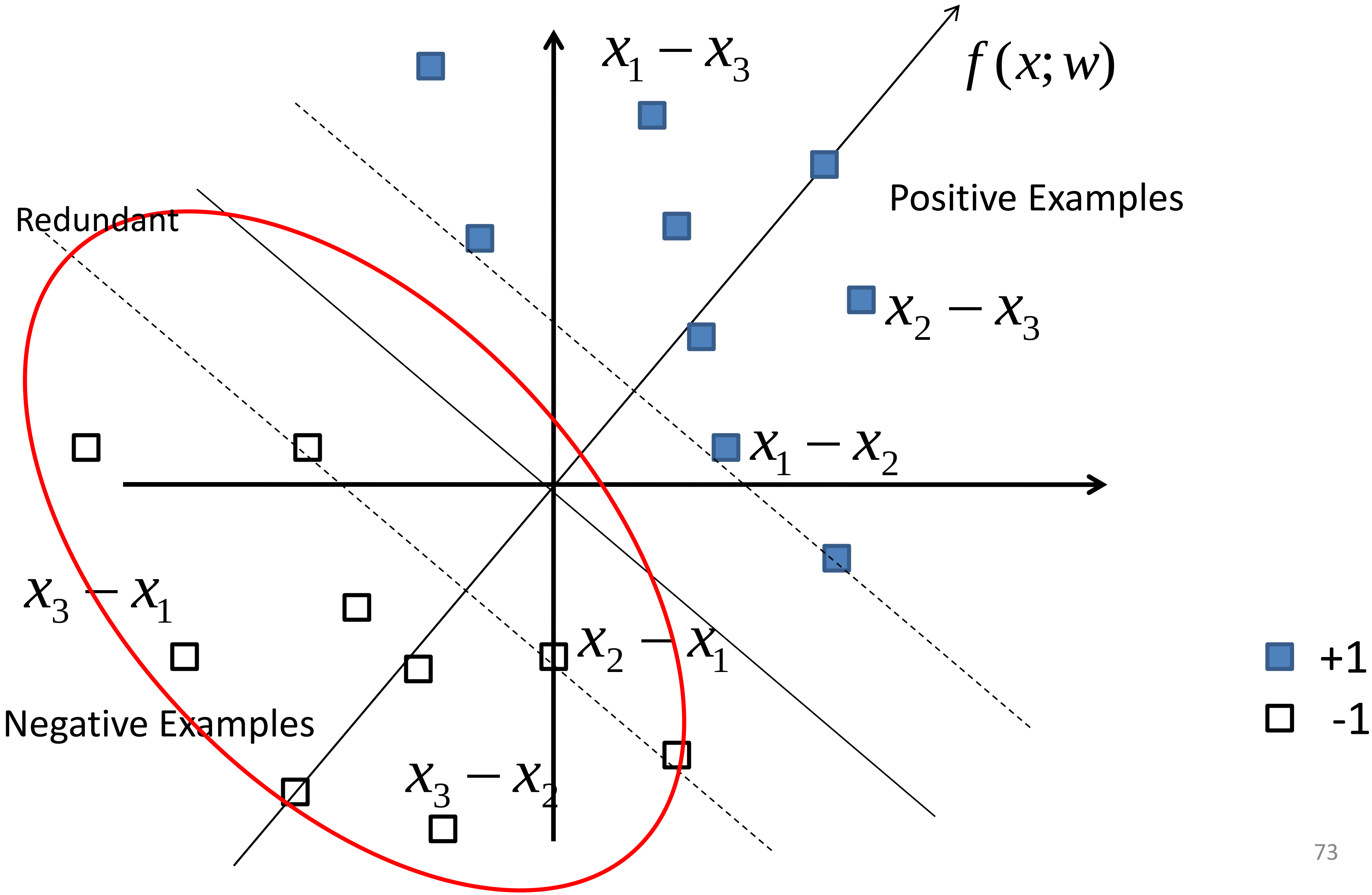
- Transforming to pairwise classification:

$$(x_i - x_j, z), \;\; y = \begin{cases} +1 & x_i \succ x_j \\ -1 & x_j \succ x_i \end{cases}$$

Ranking Problem



Transformed Pairwise Classification Problem



Ranking SVM

- Pairwise classification on differences of feature vectors
- Corresponding positive and negative examples
- Negative examples are redundant and can be discarded
- Hyper plane passes the origin
- Soft margin and kernel can be used
- *Ranking SVM =* pairwise classification SVM

Learning of Ranking SVM

$$\min_{w,\xi} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \xi_i$$

$$y_i \left\langle w, x_i^{(1)} - x_i^{(2)} \right\rangle \geq 1 - \xi_i \quad i = 1, \cdots, N$$

$$\xi_i \geq 0$$

$$\Updownarrow$$

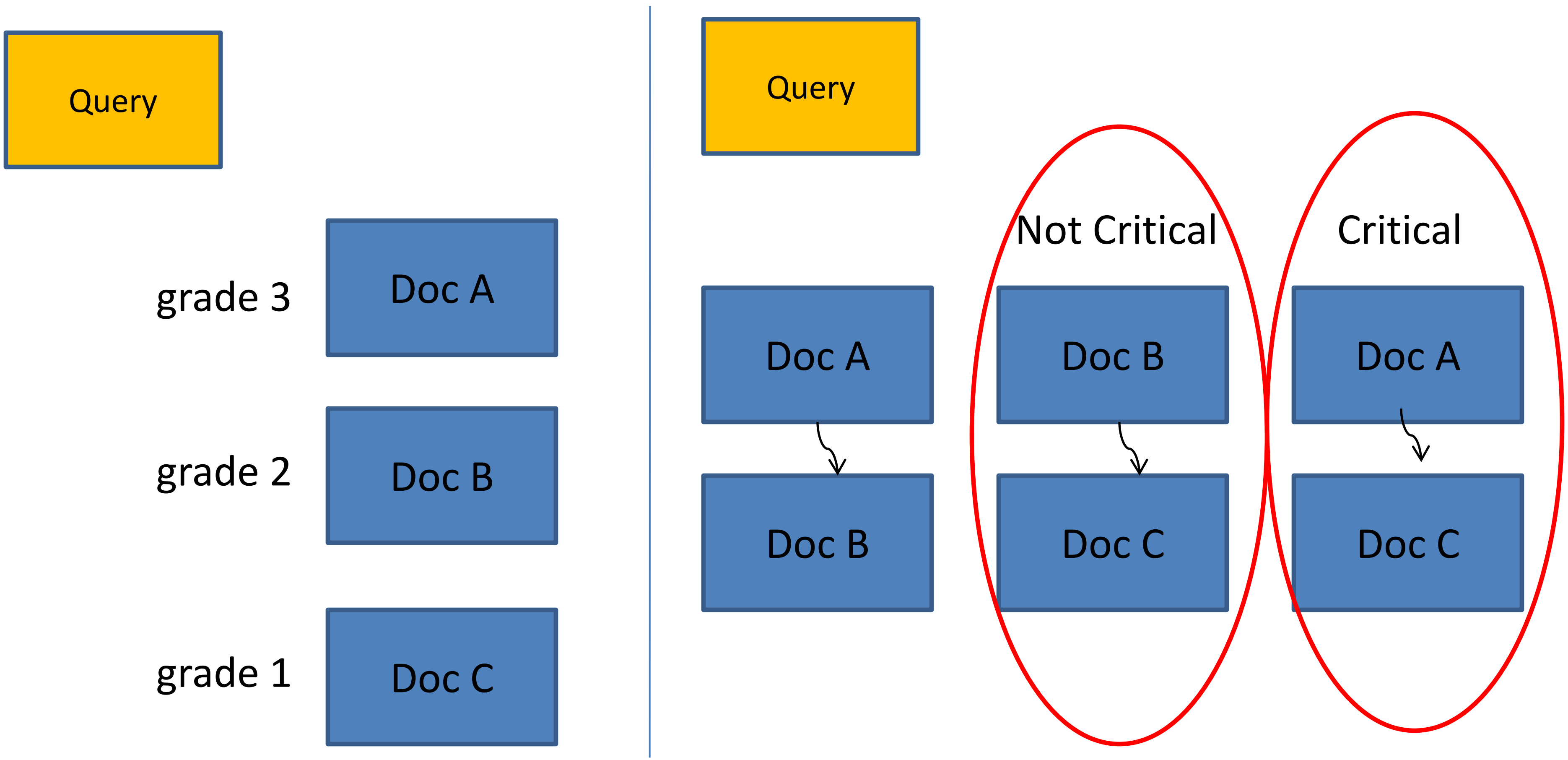
$$\min_{w} \sum_{i=1}^{l} \left[1 - y_i \left\langle w, x_i^{(1)} - x_i^{(2)} \right\rangle \right]_+ + \lambda \| w \|^2$$

$$[s]_+ = \max(0, s) \quad \lambda = \frac{1}{2C}$$

IR SVM

Cost-sensitive Pairwise Classification

- Converting to document pairs



Problems with Ranking SVM

- Not sufficient emphasis on correct ranking on top

  grades: 3, 2, 1

  ranking 1: 2 3 2 1 1 1 1

  ranking 2: 3 2 1 2 1 1 1

  ranking 2 should be better than ranking 1

  Ranking SVM views them as the same
- Numbers of pairs vary according to queries

  q1: 3 2 2 1 1 1 1

  q2: 3 3 2 2 2 1 1 1 1 1

  number of pairs for q1 : 2*(2-2) + 4*(3-1) + 8*(2-1) = 14

  number of pairs for q2: 6*(3-2) + 10*(3-1) + 15*(2-1) = 31

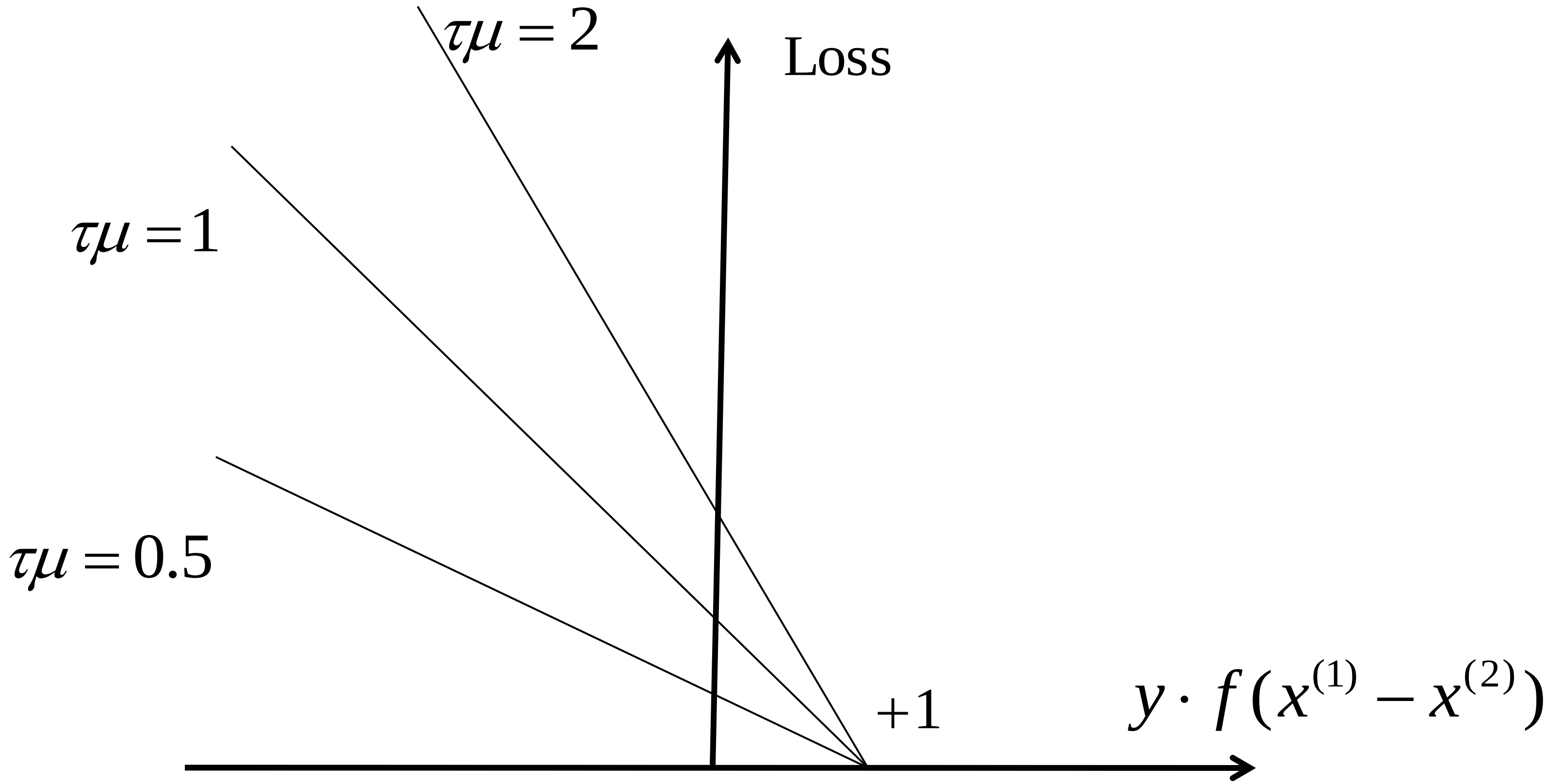
  Ranking SVM is biased toward q2

IR SVM

- Solving the two problems of Ranking SVM
- Higher weight on important grade pairs $\tau_{k(i)}$
- Normalization weight on pairs in query $\mu_{q(i)}$
- IR SVM = Ranking SVM using modified hinge loss

Modified Hinge Loss function

$$\min_{w} \sum_{i=1}^{l} \tau_{k(i)} \mu_{q(i)} \left[1 - y_i \left\langle w, x_i^{(1)} - x_i^{(2)} \right\rangle \right]_+ + \lambda \| w \|^2$$



Learning of IR SVM

$$\min_{w} \sum_{i=1}^{l} \tau_{k(i)} \mu_{q(i)} \left[1 - y_i \left\langle w, x_i^{(1)} - x_i^{(2)} \right\rangle \right]_+ + \lambda \| w \|^2$$

$$\Updownarrow$$

$$\min_{w,\xi} \frac{1}{2} \| w \|^2 + \sum_{i=1}^{l} C_i \xi_i$$

$$y_i \left\langle w, x_i^{(1)} - x_i^{(2)} \right\rangle \geq 1 - \xi_i \quad i = 1, \cdots, l$$

$$\xi_i \geq 0$$

$$C_i = \frac{\tau_{k(i)} \mu_{q(i)}}{2\lambda}$$

ListNet

Plackett-Luce Model (Permutation Probability)

- Probability of permutation π is defined as

$$P(\pi) = \prod_{i=1}^{n} \frac{s_{\pi(i)}}{\sum_{j=i}^{n} s_{\pi(j)}}$$

- Example:

$$P(\mathrm{ABC}) = \boxed{\frac{s_A}{s_A + s_B + s_C}} \cdot \boxed{\frac{s_B}{s_B + s_C}} \cdot \boxed{\frac{s_C}{s_C}}$$

P(A ranked No.1)

P(B ranked No.2 | A ranked No.1)

P(C ranked No.3 | A ranked No.1, B ranked No.2)

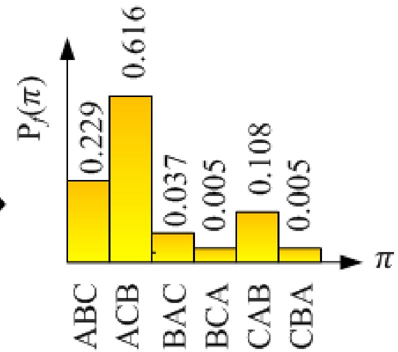
Properties of Plackett-Luce Model

- Objects: ABC
- Scores: $s_{\mathrm{A}} = 5, s_{\mathrm{B}} = 3, s_{\mathrm{C}} = 1$
- Property 1: *P*(ABC) is largest, *P*(CBA) is smallest
- Property 2: swap B and C in ABC, *P*(ABC) > *P*(ACB)

KL Divergence between Permutation Probability Distributions

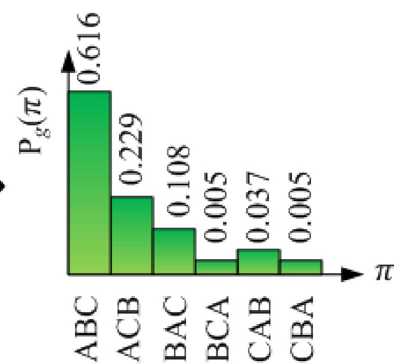
f: $f(\text{A}) = 3, f(\text{B})=0, f(\text{C})=1$;
Ranking by f: ABC

$\leftrightarrow$



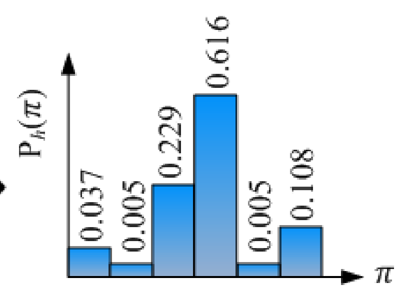
g: $g(\text{A}) = 6, g(\text{B})=4, g(\text{C})=3$;
Ranking by g: ABC

$\leftrightarrow$



h: $h(\text{A}) = 4, h(\text{B})=6, h(\text{C})=3$;
Ranking by h: ACB

$\leftrightarrow$



Plackett-Luce Model (Top-k Probability)

- Computation of permutation probabilities is intractable
- Top-k probability
  - Defining Top-k subgroup $G(o_1 \ldots o_k)$ containing all permutations whose top-k objects are $o_1, \ldots, o_k$
  - $P(G(o_1 \cdots o_k)) = \prod_{i=1}^{k} \frac{s_{o_i}}{\sum_{j=i}^{n} s_{o_j}}$
  - Time complexity of computation : from $n!$ to $n!/(n-k)!$
- Example: $P(G(\mathrm{A})) = \frac{s_A}{s_A + s_B + s_C}$

ListNet

- Parameterized Plackett-Luce Model

$$s = \exp(f(x;w))$$

$$P(G(x_1 \cdots x_k)) = \prod_{i=1}^{k} \frac{s_{x_i}}{\sum_{j=i}^{n} s_{x_j}}$$

- Ranking Model: $f(x;w)$ = Neural Net

ListNet (cont')

- Loss function = KL-divergence between two Top-k probability distributions from ground truth and ranking model

$$L(w) = -\sum_{\pi \in \Omega^k} \left(\prod_{i=1}^{k} \frac{\exp(y_i))}{\sum_{j=i}^{n} \exp(y_j)} \right) \log \left(\prod_{i=1}^{k} \frac{\exp(f(x_i; w))}{\sum_{j=i}^{n} \exp(f(x_j; w))} \right)$$

- Algorithm = Gradient Descent

AdaRank

Listwise Loss

$$
q_1 \left\{ \begin{array}{l} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{1,n_1} \end{array} \right. \quad \boxed{\begin{array}{ll} \pi_{1,1} & y_{1,1} \\ \pi_{1,2} & y_{1,2} \\ & \\ \pi_{1,n_1} & y_{1,n_1} \end{array}}
$$

$$\vdots$$

$$\vdots$$

$$
q_m \left\{ \begin{array}{l} x_{m,1} \\ x_{m,2} \\ \vdots \\ x_{m,n_m} \end{array} \right. \quad \boxed{\begin{array}{ll} \pi_{m,1} & y_{m,1} \\ \pi_{m,2} & y_{m,2} \\ & \\ \pi_{m,n_m} & y_{m,n_m} \end{array}}
$$

$$\max_{f \in \mathcal{F}} \sum_{i=1}^{m} E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)$$

$$\Downarrow$$

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} (1 - E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i))$$

AdaRank

- Optimizing exponential loss function
- Algorithm: AdaBoost-like algorithm for ranking

Loss Function of AdaRank

$$\max_{f \in \mathcal{F}} \sum_{i=1}^{m} \boxed{E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)}$$

Any evaluation measure taking value between [-1,+1]

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} (1 - E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i))$$

$$e^{-x} \geq 1 - x$$

$$\min_{f \in \mathcal{F}} \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f), \mathbf{y}_i)\}$$

$$f(\vec{x}) = \sum_{t=1}^{T} \alpha_t h_t(\vec{x})$$

$$\min_{h_t \in \mathcal{H}, \alpha_t \in \mathfrak{R}^+} L(h_t, \alpha_t) = \sum_{i=1}^{m} \exp\{-E(\pi(q_i, \mathbf{d}_i, f_{t-1} + \alpha_t h_t), \mathbf{y}_i)\}$$

AdaRank Algorithm

Input: $S = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{m}$
Parameter: T (number of iterations)
Evaluation measure: E
Initialize $P_1(i) = 1/m$
For $t = 1, \cdots, T$

- Create weak ranker h_t with weighted distribution $\boldsymbol{P}_t$ on training data S
- Choose α_t

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{\sum_{i=1}^{m} P_t(i)(1 + E(\pi_i, \mathbf{y}_i))}{\sum_{i=1}^{m} P_t(i)(1 - E(\pi_i, \mathbf{y}_i))}$$

- where $\pi_i = \text{sort}_{h_t}(\mathbf{x}_i)$
- Create f_t

$$f_t(x) = \sum_{k=1}^{t} \alpha_k h_k(x)$$

- Update $\boldsymbol{P}_{t+1}$

$$P_{t+1}(i) = \frac{\exp(-E(\pi_i, \mathbf{y}_i))}{\sum_{j=1}^{m} \exp(-E(\pi_j, \mathbf{y}_j))}$$

- where $\pi_i = \text{sort}_{f_t}(\mathbf{x}_i)$

End For
Output: the ranking model $f(x) = f_T(x)$

SVM MAP

Scoring Function

$$S(\mathbf{x}_i, \pi_i) = \langle w, \sigma(\mathbf{x}_i, \pi_i) \rangle,$$

$$\sigma(\mathbf{x}_i, \pi_i) = \frac{2}{n_i(n_i - 1)} \sum_{k,l:k<l} z_{kl}(x_{ik} - x_{il}),$$

$z_{kl} = +1$ if $\pi_i(k) < \pi_i(l)$ (x_{ik} is ranked ahead of x_{il} in π_i), and -1, otherwise.

Scoring Function (cont')

- Ranking model is linear model
- The scoring function gives
  - highest score to the perfect ranking
  - lower scores to imperfect rankings

Example of Scoring Function

Objects: A, B, C

$f_A = \langle w, x_A \rangle$, $f_B = \langle w, x_B \rangle$, $f_C = \langle w, x_C \rangle$

Suppose $f_A > f_B > f_C$

For example:

Permutation1: ABC

Permutation2: ACB

$S_{ABC} = \frac{1}{6}\langle w, ((x_A - x_B) + (x_B - x_C) + (x_A - x_C))\rangle$

$S_{ACB} = \frac{1}{6}\langle w, ((x_A - x_C) + (x_C - x_B) + (x_A - x_B))\rangle$

$S_{ABC} > S_{ACB}$

Loss Function

$$\sum_{i=1}^{m}\left[\max_{\pi_i^*\in\Pi_i^*,\pi_i\in\Pi_i\setminus\Pi_i^*}\left(\left(E(\pi_i^*,\mathbf{y}_i)-E(\pi_i,\mathbf{y}_i)\right)-\left(S(\mathbf{x}_i,\pi_i^*)-S(\mathbf{x}_i,\pi_i)\right)\right)\right]_+ ,$$

Difference between Evaluation Measures

Difference between Scoring Functions

π_i^* Perfect ranking

π_i Imperfect ranking

SVM MAP

$$\min_{w;\xi\geq 0} \frac{1}{2}||w||^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i$$
$$s.t. \quad \forall i, \forall \pi_i^* \in \Pi_i^*, \forall \pi_i \in \Pi_i \setminus \Pi_i^* :$$
$$S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i) \geq E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i) - \xi_i,$$

$$\sum_{i=1}^{m} \left[\max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) - (S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i)) \right]_+ + \lambda ||w||^2.$$

Borda Count

Ranking Function

- Sum of number of objects ranked behind

$$S_D = F(\Sigma) = \sum_{i=1}^{k} S_i$$

$$S_i \equiv \begin{pmatrix} s_{i,1} \\ \vdots \\ s_{i,j} \\ \vdots \\ s_{i,n} \end{pmatrix}$$

$$s_{i,j} = n - \sigma_i(j),$$

Example

- Three basic rankings

$$\begin{array}{ccc} \sigma_1 & \sigma_2 & \sigma_3 \\ \begin{pmatrix} A \\ B \\ C \end{pmatrix} & \begin{pmatrix} A \\ C \\ B \end{pmatrix} & \begin{pmatrix} B \\ A \\ C \end{pmatrix} \end{array}$$

- Ranking scores

$$S_D = \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ 1 \end{pmatrix}$$

- Ranking

$$\begin{array}{c} \pi \\ \begin{pmatrix} A \\ B \\ C \end{pmatrix} \end{array}$$

5. Learning to rank Applications

Learning to rank Applications

- Web Search
- Recommender System
- Key Phrase Extraction
- Query Dependent Summarization
- Machine Translation

Recommender System (Collaborative Filtering)

- Problem formulation
  - Input: users’ ratings on some items
  - Output: users’ ratings on other items
  - Assumption: users sharing same ratings on input items tend to agree on new items
- Solutions
  - Classification
  - Ordinal Regression
  - Learning to Rank

Recommender System

	Item1	**Item2**	**Item3**	**...**	
User1	5	4			
User2	1		2		2
...		?	?	?	
UserM	4	3			

Recommender System Using RankBoost

- Ranking items according to users
- Justification: users tend to rate on different scales
- Method: RankBoost
- Result: RankBoost > Nearest Neighbor

Key Phrase Extraction

- Problem formulation
  - Input: document
  - Output: keyphrases of document
  - Two steps: phrase extraction and keyphrase identification
- Traditional approach
  - Classification: keyphrase vs non-keyphrase

Key Phrase Extraction Using Ranking SVM

- Ranking of phrases as keyphrases
- Justification: keyphrase or non-keyphrase is relative
- Method: Ranking SVM
- Result: Ranking SVM > SVM

6. Theory of Learning to Rank

Statistical Learning Formulation

- Input space $\mathcal{X}$: lists of feature vectors
- Output space $\mathcal{Y}$: lists of grades
- Input $\mathbf{x}$: list of feature vectors
- Output $\mathbf{y}$: list of grades
- Training data: $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_m, \mathbf{y}_m).$
- Global ranking model: $F(\mathbf{x}) = [f(x_1), f(x_2), \cdots, f(x_n)]$
- Loss function: $L(F(\mathbf{x}), \mathbf{y}).$

Statistical Learning Formulation

- Risk function: $R(F) = \int_{\mathcal{X}\times\mathcal{Y}} L(F(\mathbf{x}), \mathbf{y}) dP(\mathbf{x}, \mathbf{y}).$
- Empirical risk: $\hat{R}(F) = \frac{1}{m}\sum_{i=1}^{m} L(F(\mathbf{x}_i), \mathbf{y}_i).$
- Surrogate loss function: $L'(F(\mathbf{x}), \mathbf{y}).$

Loss Functions

- True loss function

$$L(F(\mathbf{x}), \mathbf{y}) = 1 - NDCG$$

$$L(F(\mathbf{x}), \mathbf{y}) = 1 - MAP.$$

- Difficult to optimize
  - Use of sorting
  - Non continuous
- Using surrogate loss functions

Loss Functions

- Pointwise loss (squared): $L'(F(\mathbf{x}),\mathbf{y}) = \sum_{i=1}^{n}(f(x_i) - y_i)^2.$
- Pairwise loss (hinge, exponential, logistic)

$$L'(F(\mathbf{x}),\mathbf{y}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}[1 - \mathrm{sign}(y_i - y_j)(f(x_i) - f(x_j))]_+, \quad \text{when } y_i \neq y_j,$$

$$L'(F(\mathbf{x}),\mathbf{y}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\exp\left(-\mathrm{sign}(y_i - y_j)(f(x_i) - f(x_j))\right), \quad \text{when } y_i \neq y_j.$$

$$L'(F(\mathbf{x}),\mathbf{y}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\log\left(1 + \exp(-\mathrm{sign}(y_i - y_j)(f(x_i) - f(x_j)))\right), \quad \text{when } y_i \neq y_j.$$

Relations between Surrogate Loss and True Loss

- Pointwise loss

$$1 - NDCG \leq \frac{1}{G_{max}} \left(2 \sum_{i=1}^{n} D(\pi(i))^2 \right)^{1/2} L'(F(\mathbf{x}), \mathbf{y})^{1/2},$$

- Pairwise loss

$$1 - NDCG \leq \frac{\max_i (G(i) D(\pi(i)))}{G_{max}} L'(F(\mathbf{x}), \mathbf{y}),$$

- Listwise loss

$$1 - NDCG \leq \frac{\max_i (G(i) D(\pi(i)))}{\ln 2 \cdot G_{max}} L'(F(\mathbf{x}), \mathbf{y}),$$

Theoretical Analysis

- Generalization ability
- Consistency

7. Ongoing and Future Work

Future and Ongoing Work

- Training data creation
- Semi-supervised learning and active learning
- Feature learning
- Scalable and efficient training
- Domain adaptation
- Ranking by ensemble learning
- Global ranking
- Ranking of objects in graph

Summary

Outline of Tutorial

1. Learning to Rank
2. Learning for Ranking Creation
3. Learning for Ranking Aggregation
4. Methods of Learning to Rank
5. Applications of Learning to Rank
6. Theory of Learning to Rank
7. Ongoing and Future Work

Contact: hangli@microsoft.com