

# Learning to rank search results

Unsupervised rank fusion and Learning to re-ranking methods

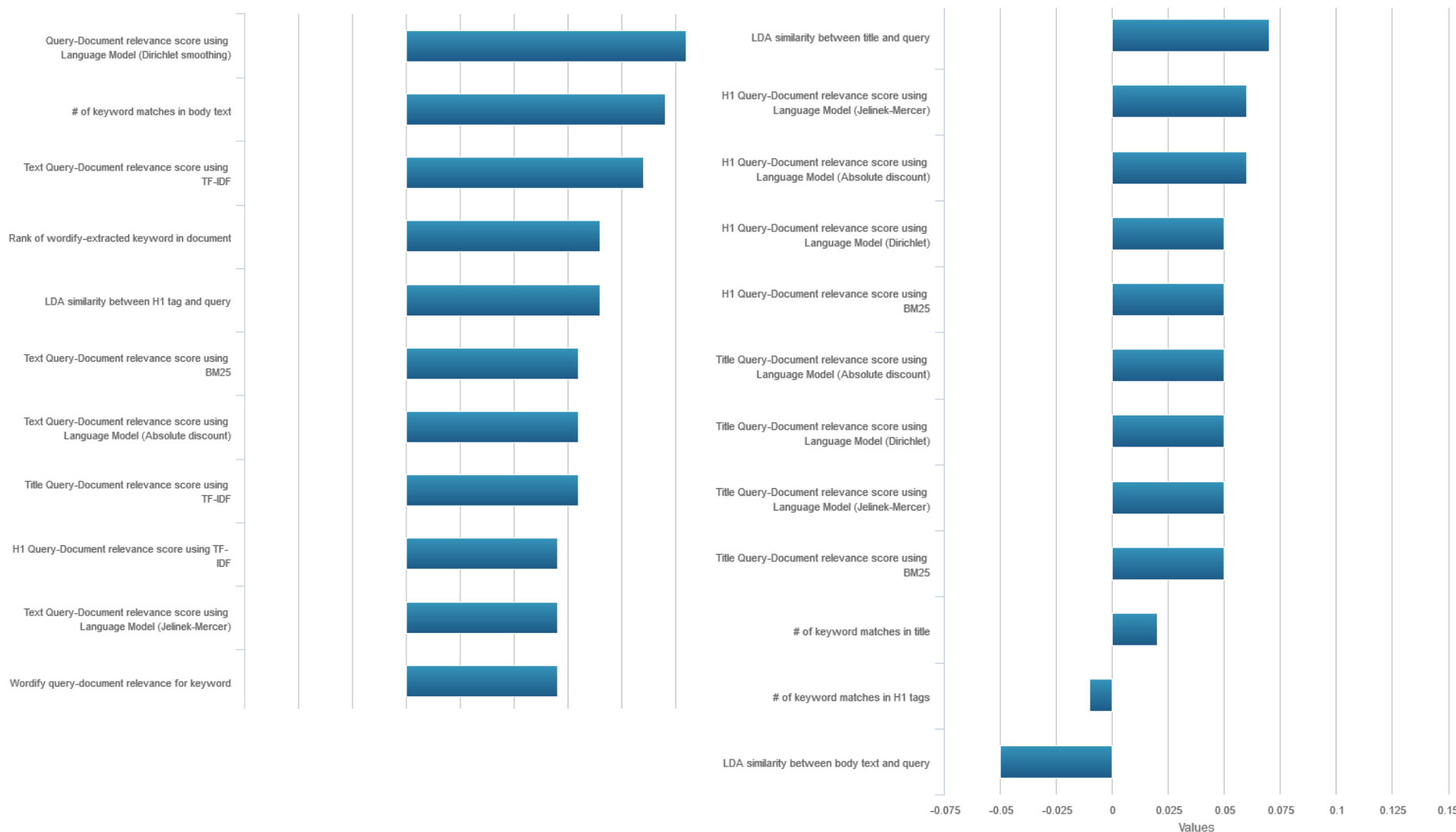
## Information Retrieval

Method	Stop	Stem	TREC45				GOV2			
			1998		1999		2004		2005	
			P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP
bm25	No	No	0.424	0.178	0.440	0.205	0.471	0.242	0.534	0.277
bm25	No	Yes	0.440	0.199	<b>0.464</b>	0.247	<b>0.500</b>	0.266	0.600	0.334
bm25	Yes	No	0.424	0.178	0.438	0.205	0.467	0.243	0.538	0.276
bm25	Yes	Yes	0.440	0.199	<b>0.464</b>	0.247	<b>0.500</b>	0.266	0.592	0.333
bm25_b=0	No	No	0.402	0.177	0.406	0.207	0.418	0.171	0.538	0.207
bm25_notf_b=0	No	No	0.256	0.141	0.224	0.147	0.069	0.050	0.106	0.083
dfr-gb2	No	No	0.426	0.183	0.446	0.216	0.465	0.248	0.550	0.269
dfr-gb2	No	Yes	0.448	<b>0.204</b>	0.458	0.253	0.471	0.252	0.584	0.319
dfr-gb2	Yes	No	0.426	0.183	0.446	0.216	0.465	0.248	0.550	0.269
dfr-gb2	Yes	Yes	0.448	<b>0.204</b>	0.458	0.253	0.471	0.252	0.584	0.319
lm-dirichlet-1000	No	No	0.450	0.193	0.428	0.226	0.484	0.244	0.580	0.293
lm-dirichlet-1000	No	Yes	<b>0.464</b>	<b>0.204</b>	0.434	0.262	0.492	0.270	0.600	<b>0.343</b>
lm-dirichlet-1000	Yes	No	0.448	0.193	0.430	0.226	0.494	0.247	0.568	0.291
lm-dirichlet-1000	Yes	Yes	0.462	<b>0.204</b>	0.436	<b>0.262</b>	0.488	<b>0.272</b>	<b>0.602</b>	0.341
lm-jelinek-0.5	No	No	0.390	0.179	0.432	0.208	0.416	0.211	0.494	0.257
lm-jelinek-0.5	No	Yes	0.406	0.192	0.434	0.248	0.437	0.225	0.522	0.302
lm-jelinek-0.5	Yes	No	0.390	0.179	0.432	0.209	0.414	0.212	0.482	0.253
lm-jelinek-0.5	Yes	Yes	0.406	0.192	0.436	0.249	0.445	0.225	0.508	0.298
lm-unsmoothed	No	No	0.354	0.114	0.396	0.141	0.402	0.171	0.492	0.231
lm-unsmoothed	No	Yes	0.384	0.134	0.416	0.180	0.433	0.196	0.538	0.285
lm-unsmoothed	Yes	No	0.352	0.114	0.394	0.141	0.400	0.172	0.484	0.230
lm-unsmoothed	Yes	Yes	0.384	0.134	0.416	0.180	0.439	0.196	0.518	0.283
prox	No	No	0.396	0.124	0.370	0.146	0.424	0.173	0.560	0.230
prox	No	Yes	0.418	0.139	0.430	0.184	0.453	0.207	0.576	0.283
prox	Yes	No	0.396	0.123	0.370	0.146	0.422	0.173	0.546	0.232
prox	Yes	Yes	0.416	0.139	0.430	0.184	0.447	0.204	0.556	0.282
vsm-lintf-logidf	No	No	0.266	0.106	0.240	0.120	0.298	0.092	0.282	0.097
vsm-logtf-logidf	No	No	0.264	0.126	0.252	0.135	0.120	0.060	0.194	0.092
vsm-logtf-noidf-raw	No	No	0.342	0.132	0.328	0.154	0.400	0.144	0.466	0.151

# Google rank correlation analysis

- <https://moz.com/search-ranking-factors/correlations>
- Analysis of the correlation between query/document features and the results returned by Google
- In 2008, Google reported using over 200 features ([Amit Singhal, NYT, 2008-06-03](#))
- In 2016, it's over 300 features (Jeff Dean, WSDM 2016)
  - How can we take advantage of all types of features for ranking?

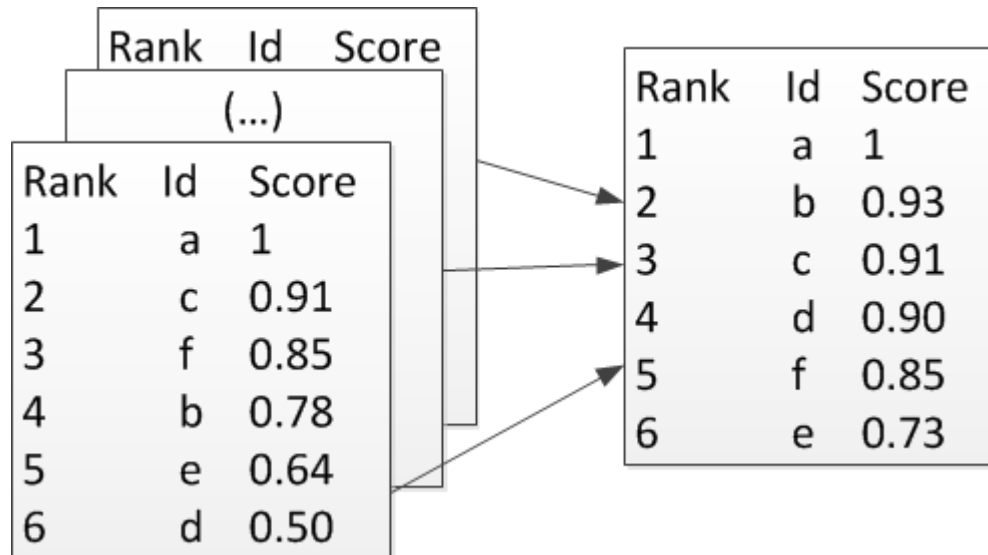
# Google rank correlation analysis (query based)



# How to handle different search ranks?

- Which model should we select for our production system?
  - Not trivial. Would require even more relevance judgments.
- Can we merge these ranks into a single, better, rank?

**Yes, we can!**



# Example

- Consider the following three ranks of five documents (tweets), for a given query:

	Tweet Desc. BM25*		Tweet Desc. LM*		Tweet count (user)	
Position	id	Score	id	Score	id	Score
1	D5	2.34	D5	1.23	D4	19685
2	D4	2.12	D4	1.02	D1	18756
3	D3	1.93	D3	1.00	D2	2342
4	D2	1.43	D1	0.85	D5	2341
5	D1	1.34	D2	0.71	D3	123

\*similarity between query text and tweet description, as returned by retrieval model (e.g. BM25, LM)

- On a given rank  $i$ , a document  $d$  has a score  $s_i(d)$  and is placed on the  $r_i(d)$  position.
- Ranks are sorted by score.

# Unsupervised reranking methods

- Score-based methods
  - Comb\*
- Rank-based fusion
  - Bordafuse
  - Condorcet
  - Reciprocal Rank Fusion (RRF)

# Comb\*

- Use score of the document on the different lists as the main ranking factor:
  - This can be the Retrieval Status Value of the retrieval model.

$$CombMAX(d) = \max\{s_0(d), \dots, s_n(d)\}$$

$$CombMIN(d) = \min\{s_0(d), \dots, s_n(d)\}$$

$$CombSUM(d) = \sum_i s_i(d)$$



# CombSUM example

- CombSUM is used by Elasticsearch to combine results from multi-field queries:

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D4	2.12	1.02	19685	19688.14
D1	1.34	0.85	18756	18758.19
D5	2.34	1.23	2341	2344.57
D2	1.43	0.71	2342	2344.14
D3	1.93	1.00	123	125.93

- Ranges of the features may greatly influence ranking
  - Less prevalent on scores from retrieval models

# CombSUM example

- CombSUM is used by Elasticsearch to combine results from multi-field queries:

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D4	1.80	1.59	2.02	5.40
D5	2.30	2.66	0.23	5.19
D3	1.36	1.48	0.00	2.84
D1	0.00	0.72	1.92	2.64
D2	0.21	0.00	0.23	0.44

Normalized assuming Normal distribution:  $\frac{score - \mu}{\sigma}$

- ElasticSearch already normalizes scores returned by retrieval models
- However, scores may not follow normal distribution or **be biased on small samples** (e.g. 1000 documents retrieved by ElasticSearch)

# wComb\*

- ElasticSearch can also give higher/lower weight to scores from different fields
- These weights are then multiplied by the scores:

$$wCombSUM(d) = \sum_i w_i s_i(d)$$

- How to find these weights?
  - Manually
  - Machine learning (more on this later)

# CombMNZ

- CombMNZ multiplies the number of ranks where the document occurs by the sum of the scores obtained across all lists.

$$\text{CombMNZ}(d) = |\{i | d \in \text{Rank}_i\}| \cdot \sum_i s_i(d)$$

- Despite normalization issues common in score-based methods, CombMNZ is competitive with rank-based approaches.

# Borda fuse

- A voting algorithm based on the positions of the candidates.
  - Invented by Jean-Charles de Borda in 18<sup>th</sup> century in France
- For each rank, the document gets a score corresponding to its (inverse) position on the rank.

	Tweet Desc. BM25*	Tweet Desc. LM*	Tweet count (user)
<b>Position</b>	<b>id</b>	<b>id</b>	<b>id</b>
1	D5	D5	D4
2	D4	D4	D1
3	D3	D3	D2
4	D2	D1	D5
5	D1	D2	D3

# Borda fuse

- A voting algorithm based on the positions of the candidates.
  - Invented by Jean-Charles de Borda in 18<sup>th</sup> century in France
- For each rank, the document gets a score corresponding to its (inverse) position on the rank.
- The *fused rank* is based on the sum of all per-rank scores.

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D4				
D5				
D1				
D3				
D2				

# Borda fuse

- A voting algorithm based on the positions of the candidates.
  - Invented by Jean-Charles de Borda in 18<sup>th</sup> century in France
- For each rank, the document gets a score corresponding to its (inverse) position on the rank.
- The *fused rank* is based on the sum of all per-rank scores.

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D4	$(5-2)=3$	$(5-2)=3$	$(5-1)=4$	10
D5				
D1				
D3				
D2				

# Borda fuse

- A voting algorithm based on the positions of the candidates.
  - Invented by Jean-Charles de Borda in 18<sup>th</sup> century in France
- For each rank, the document gets a score corresponding to its (inverse) position on the rank.
- The *fused rank* is based on the sum of all per-rank scores.

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D4	3	3	4	<b>10</b>
D5	4	4	1	9
D1	0	1	3	4
D3	2	2	0	4
D2	1	0	2	3



# Condorcet

- Voting algorithm that started as a way to select the best candidate on an election
  - Marquis de Condorcet, also in 18<sup>th</sup> century France
- Based on a *majoritarian* method
  - Uses pairwise comparisons,  $r(d1) > r(d2)$ .
  - For each pair (d1,d2) we compare the number of times d1 beats d2.
  - The best candidate found through the pairwise comparisons.
- Generalizing Condorcet to produce a rank can have a high computationally complexity.
  - There are solutions to compute the rank with low complexity.

# Condorcet example

## Pairwise comparison

	D1	D2	D3	D4	D5
D1					
D2					
D3					
D4					
D5					

Tweet Desc. BM25: D2 > D1

Tweet Desc. LM : D1 > D2

Tweet count : D1 > D2

# Condorcet example

## Pairwise comparison

	D1	D2	D3	D4	D5
D1	-	2,0,1			
D2	1,0,2				
D3					
D4					
D5					

Tweet Desc. BM25: D2 > D1

Win, Draw, Lose

Tweet Desc. LM : D1 > D2

D1 vs D2 1, 0, 2

Tweet count : D1 > D2

D2 vs D1 2, 0, 1

# Condorcet example

## Pairwise comparison

	D1	D2	D3	D4	D5
D1	-	2,0,1	1,0,2	0,0,3	1,0,2
D2	1,0,2	-	1,0,2	0,0,3	2,0,1
D3	2,0,1	2,0,1	-	0,0,3	0,0,3
D4	3,0,0	3,0,0	3,0,0	-	1,0,2
D5	2,0,1	2,0,1	3,0,0	2,0,1	-

# Condorcet example

## Pairwise comparison

	D1	D2	D3	D4	D5
D1	-	2,0,1	1,0,2	0,0,3	1,0,2
D2	1,0,2	-	1,0,2	0,0,3	2,0,1
D3	2,0,1	2,0,1	-	0,0,3	0,0,3
D4	3,0,0	3,0,0	3,0,0	-	1,0,2
D5	2,0,1	2,0,1	3,0,0	2,0,1	-

## Pairwise winners

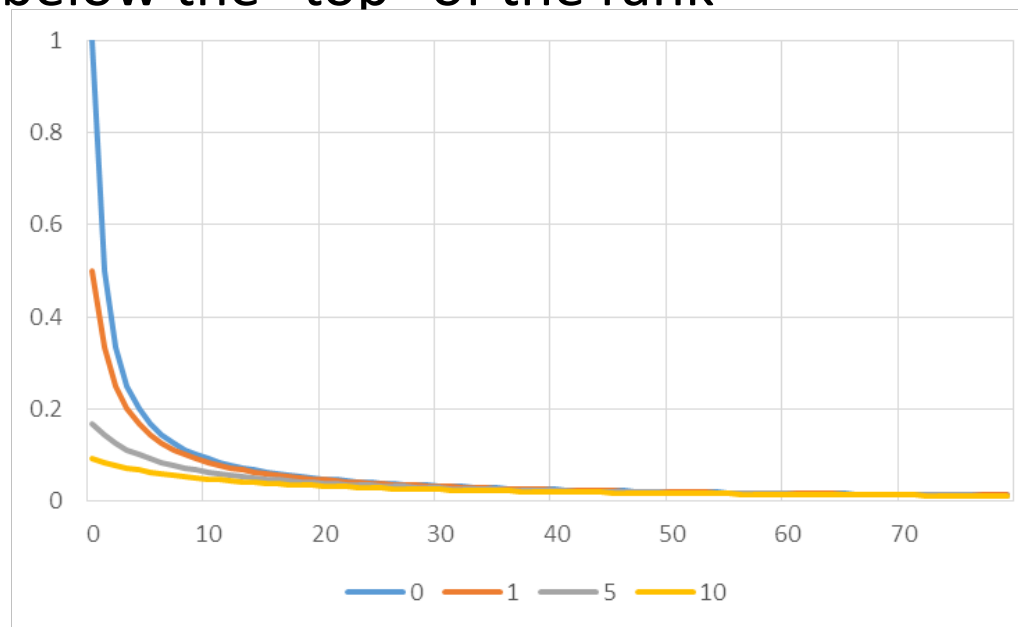
	Win	Tie	Lose	Score
D4	10	0	2	8
D5	9	0	3	6
D3	4	0	8	-4
D1	4	0	8	-4
D2	4	0	8	-4

# Reciprocal Rank Fusion (RRF)

- The reciprocal rank fusion weights each document with the inverse of its position on the rank.
- Favours documents at the “top” of the rank.
- Penalizes documents below the “top” of the rank

$$RRFscore(d) = \sum_i \frac{1}{k + r_i(d)},$$

where  $k = 60$



# RRF example

$$RRFscore(d) = \sum_i \frac{1}{k + r_i(d)}, \quad k = 0 \text{ (for this example)}$$

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D5				
D4				
D1				
D3				
D2				

# RRF example

$$RRFscore(d) = \sum_i \frac{1}{k + r_i(d)}, \quad k = 0 \text{ (for this example)}$$

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D5	1/1	1/1	1/4	2.250
D4				
D1				
D3				
D2				



# RRF example

$$RRFscore(d) = \sum_i \frac{1}{k + r_i(d)}, \quad k = 0 \text{ (for this example)}$$

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion score
D5	1/1	1/4	1/1	<b>2.250</b>
D4	1/2	1/2	1/1	2.000
D1	1/5	1/4	1/2	0.950
D3	1/3	1/3	1/5	0.866
D2	1/4	1/5	1/3	0.783

# Experimental comparison

Method	TREC45				Gov2			
	1998		1999		2005		2006	
	P@10	MAP	P@10	MAP	P@10	MAP	P@10	MAP
VSM	0.266	0.106	0.240	0.120	0.298	0.092	0.282	0.097
BIN	0.256	0.141	0.224	0.148	0.069	0.050	0.106	0.083
2-Poisson	0.402	0.177	0.406	0.207	0.418	0.171	0.538	0.207
BM25	0.424	0.178	0.440	0.205	0.471	0.243	0.534	0.277
LMJM	0.390	0.179	0.432	0.209	0.416	0.211	0.494	0.257
LMD	0.450	0.193	0.428	0.226	0.484	0.244	0.580	0.293
BM25F					0.482	0.242	0.544	0.277
BM25+PRF	0.452	<b>0.239</b>	0.454	0.249	<b>0.567</b>	0.277	<b>0.588</b>	0.314
RRF	<b>0.462</b>	0.215	<b>0.464</b>	<b>0.252</b>	0.543	<b>0.297</b>	0.570	<b>0.352</b>
Condorcet	0.446	0.207	0.462	0.234	0.525	0.281	0.574	0.325
CombMNZ	0.448	0.201	0.448	0.245	0.561	0.270	0.570	0.318
LR			0.446	0.266			<b>0.588</b>	0.309
RankSVM			0.420	0.234			0.556	0.268

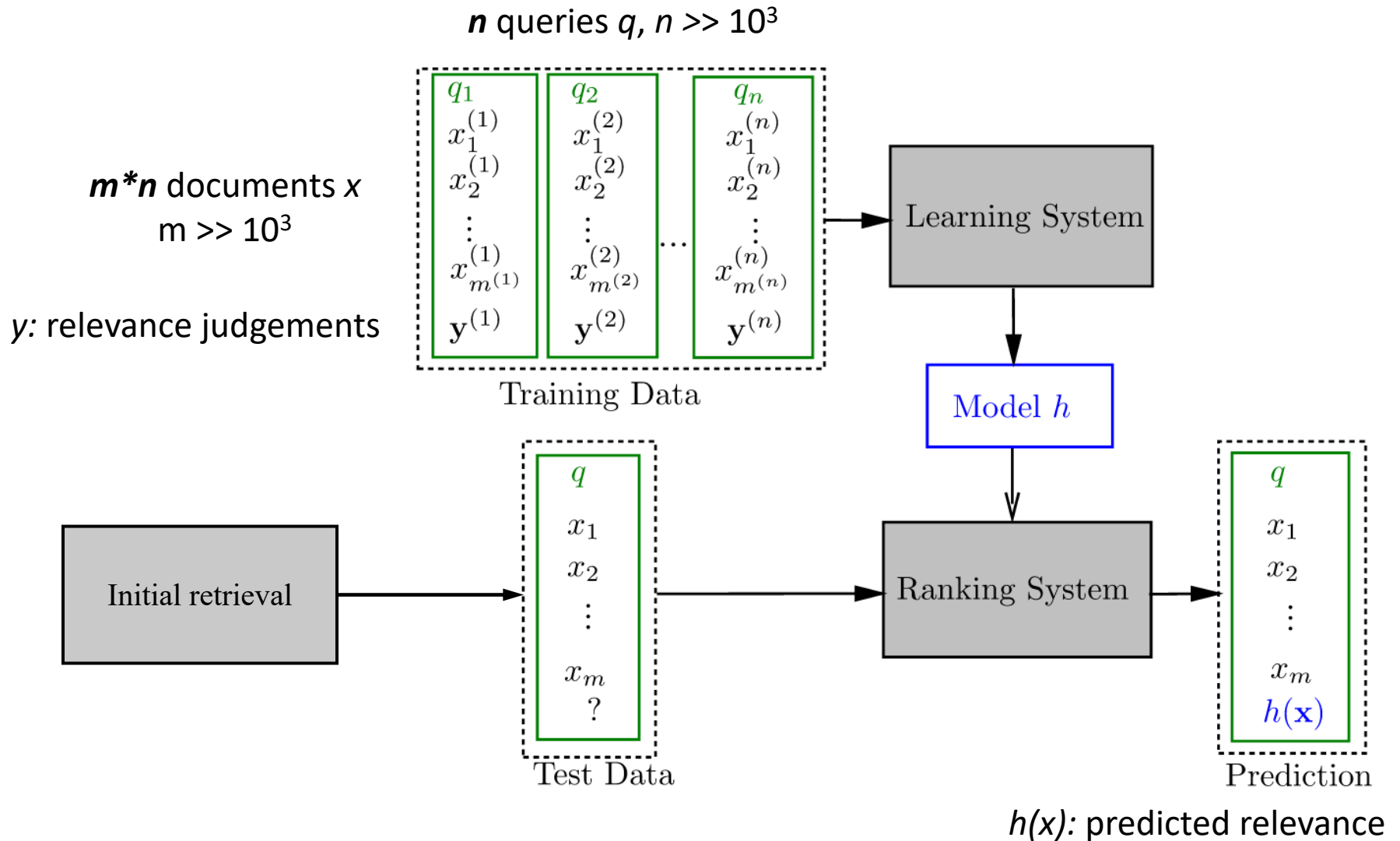


Quick Break

# Learning to rank

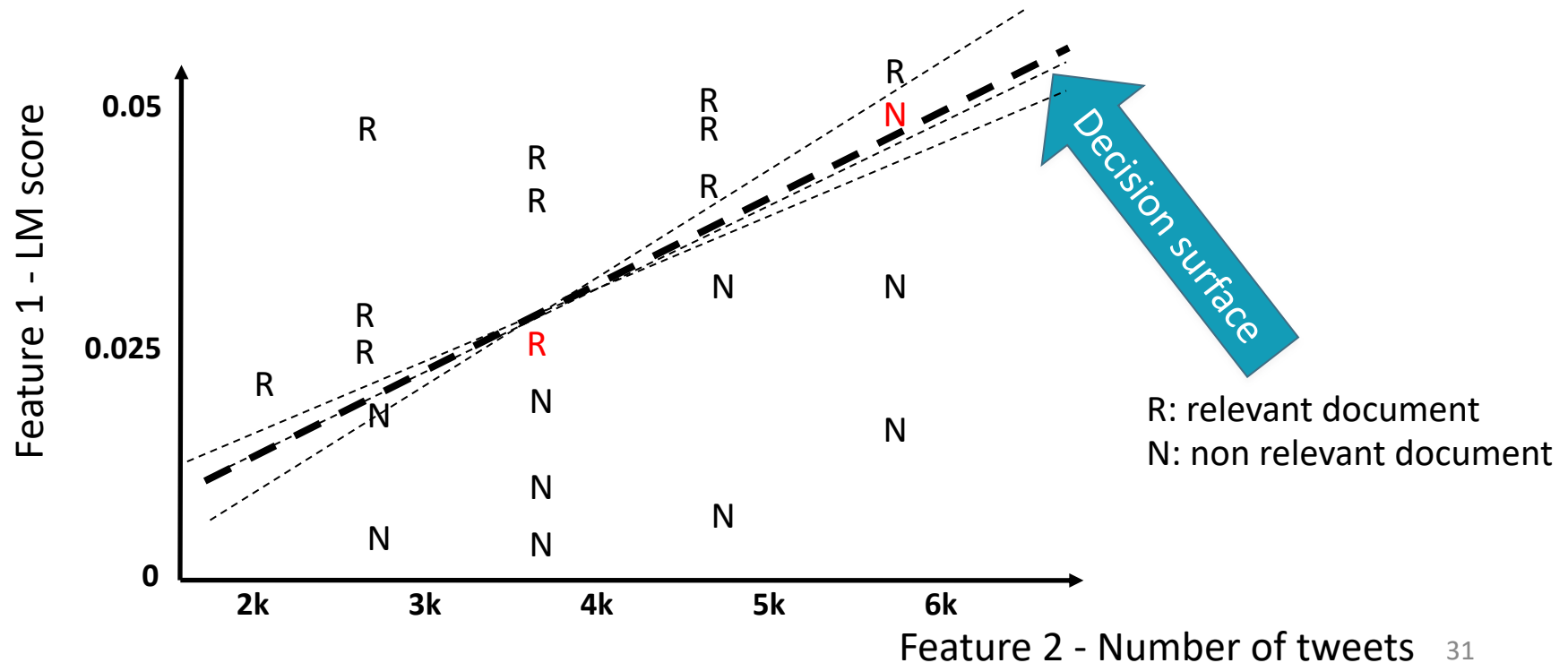
- Use machine learning techniques to automatically learn a function to rank results effectively
- **Pointwise approaches**
  - regress the relevance score, classify (q,doc) into *Rel* or *Non Rel*
- **Pairwise approaches**
  - given two documents, predict partial ranking:  $d_1 > d_2$  or  $d_2 > d_1$
- **Listwise approaches**
  - Takes ranked lists as inputs and trains a ranking function to combine the lists

# LETOR Experimental setup



# Pointwise approach

- Collect a training corpus of  $(q, d, R=\{\text{rel}, \text{nrel}\})$  triples
- Train a machine learning model on **(document-query)** samples to predict their class R



# Learning to rank features

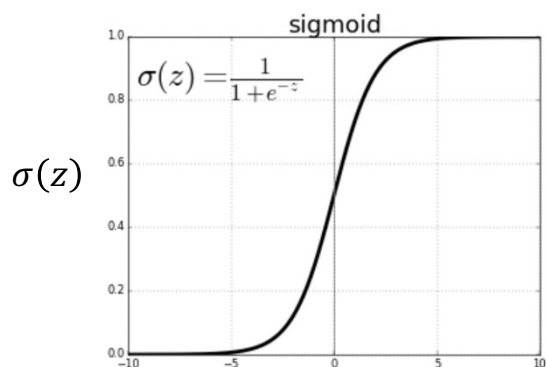
- Training samples are computed from the pair (query, document) and their similarities/features

Sample	(Q, D)	Features	R/NR
1	(1, 1)	(..., ..., ..., ..., ..., ...)	R
2	(1, 2)	(..., ..., ..., ..., ..., ...)	N
3	(1, 3)	(..., ..., ..., ..., ..., ...)	R
4	(1, 4)	(..., ..., ..., ..., ..., ...)	N
5	(2, 1)	(..., ..., ..., ..., ..., ...)	N
6	(2, 2)	(..., ..., ..., ..., ..., ...)	N
7	(2, 3)	(..., ..., ..., ..., ..., ...)	R
8	(2, 4)	(..., ..., ..., ..., ..., ...)	N
...	...	(..., ..., ..., ..., ..., ...)	...

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
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	PageRank
42	Inlink number
43	Outlink number
44	Number of slash in URL
45	Length of URL
46	Number of child page

# Learning to rank with pointwise approach

$\sigma(z)$  is the probability  
 $p(d|q, r = 1)$

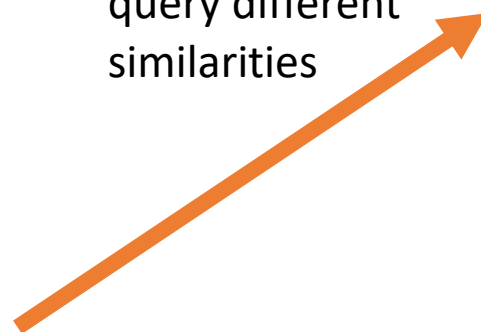


$$z = w_0 + w_1x_1 + \dots + w_nx_n$$

$w$  variables are the weights of each  
 type of query-document similarity  $x$

$z$  variable provides the ranking score of documents

X variables are  
 the document-  
 query different  
 similarities

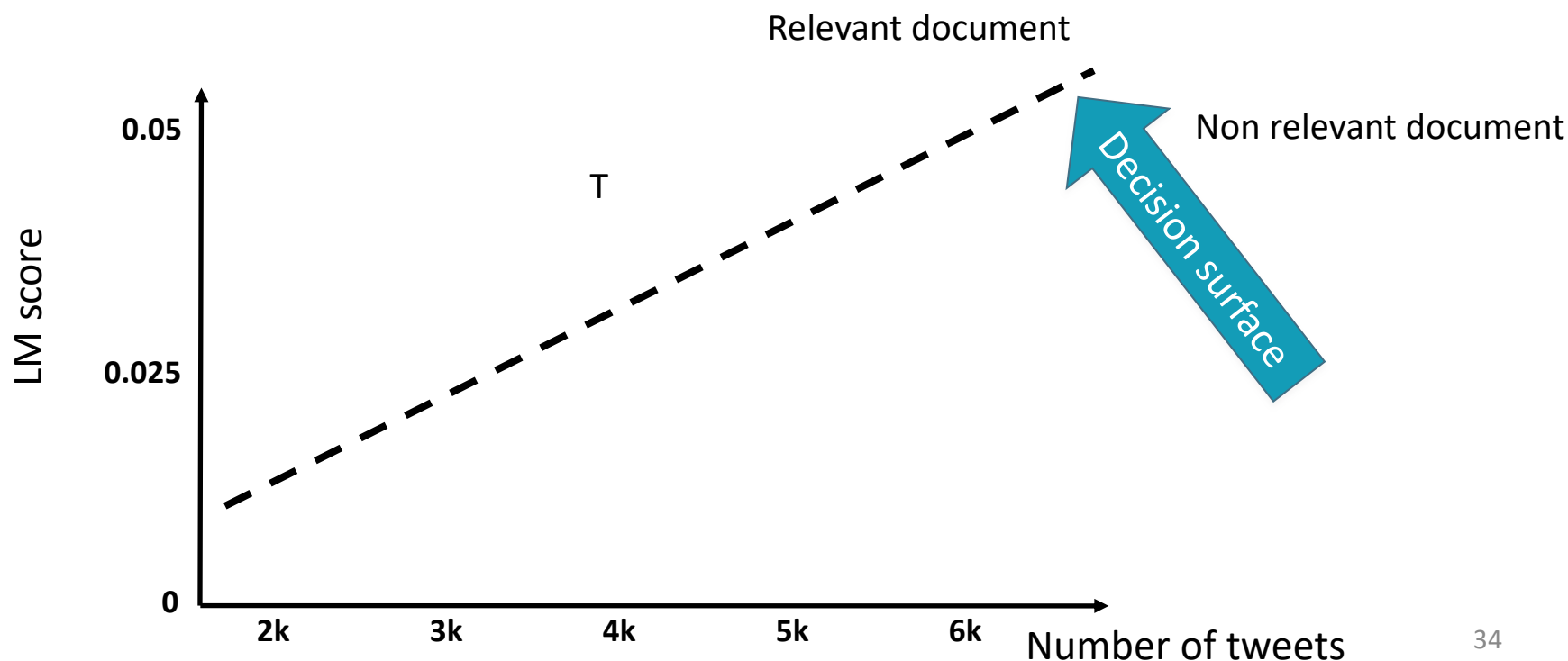


ID	Feature description
1	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in body
2	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in anchor
3	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in title
4	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in URL
5	$\sum_{t_i \in q \cap d} TF(t_i, d)$ in the whole document
6	$\sum_{t_i \in q} IDF(t_i)$ 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
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	PageRank
42	Inlink number
43	Outlink number
44	Number of slash in URL
45	Length of URL
46	Number of child page



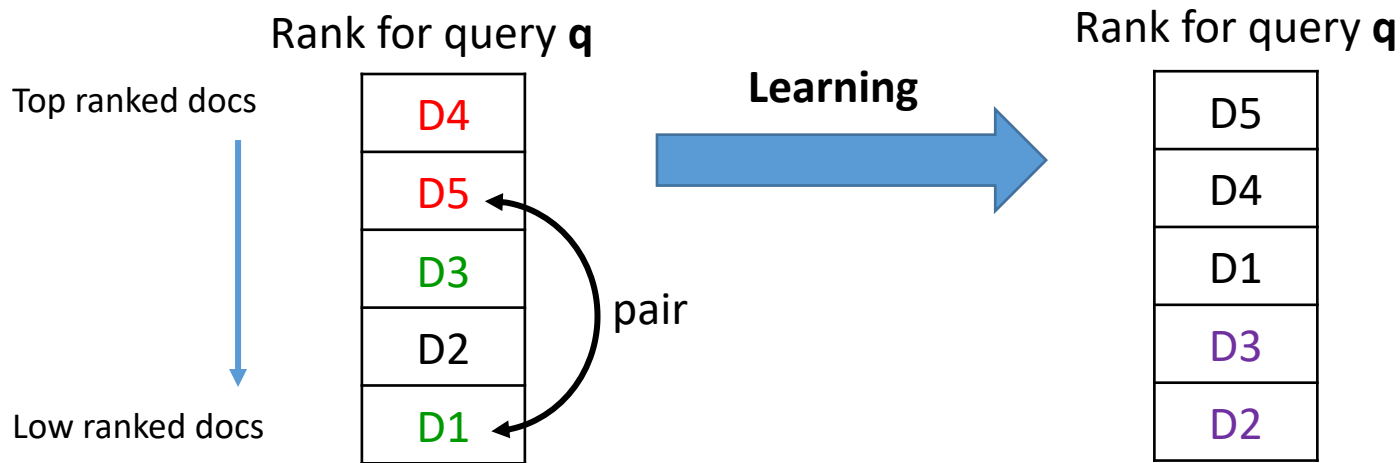
# Pointwise approach: test phase

- For each test triple (**test\_query**, **doc**, **R={rel,nrel}**)
- Predict the relevance **R** of the pair (**test\_query**, **doc**)
- Compare the predicted relevance against the true relevance.



# Pairwise approach

- Find a global order by predicting partial ranking of the documents:



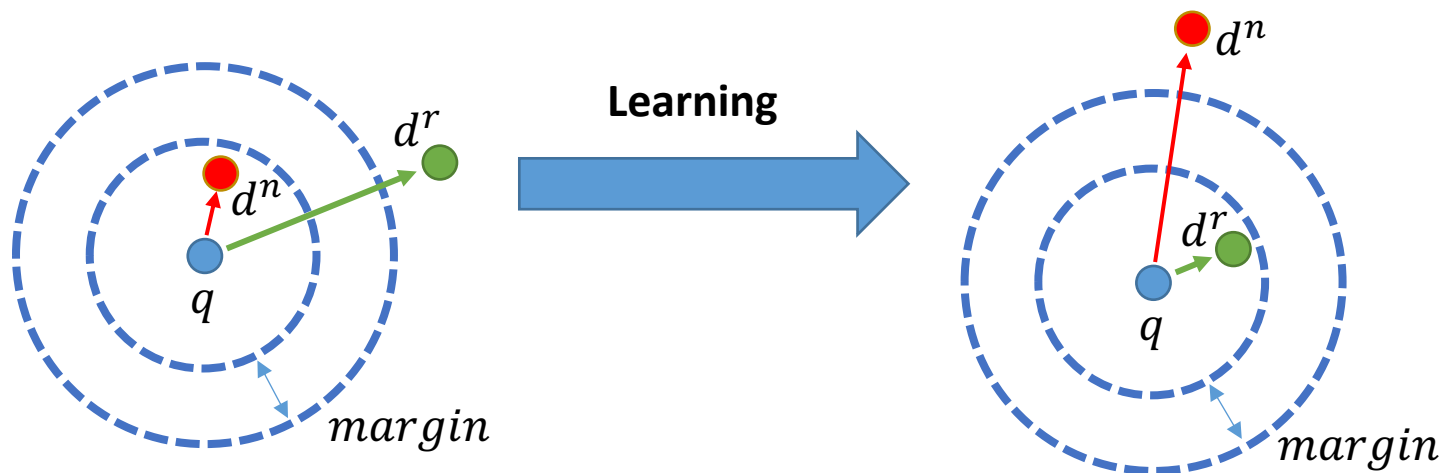
- In the above example a rank violating pair creates the training triplet sample:

$$\text{sample} = (q, d_5, d_1)$$

# Pairwise ranking loss

- The pairwise ranking loss is a function that penalizes the order of documents

$$\text{loss}(q, d^n, d^r) = \max\{0, \text{margin} - s(q, d^r) + s(q, d^n)\}$$



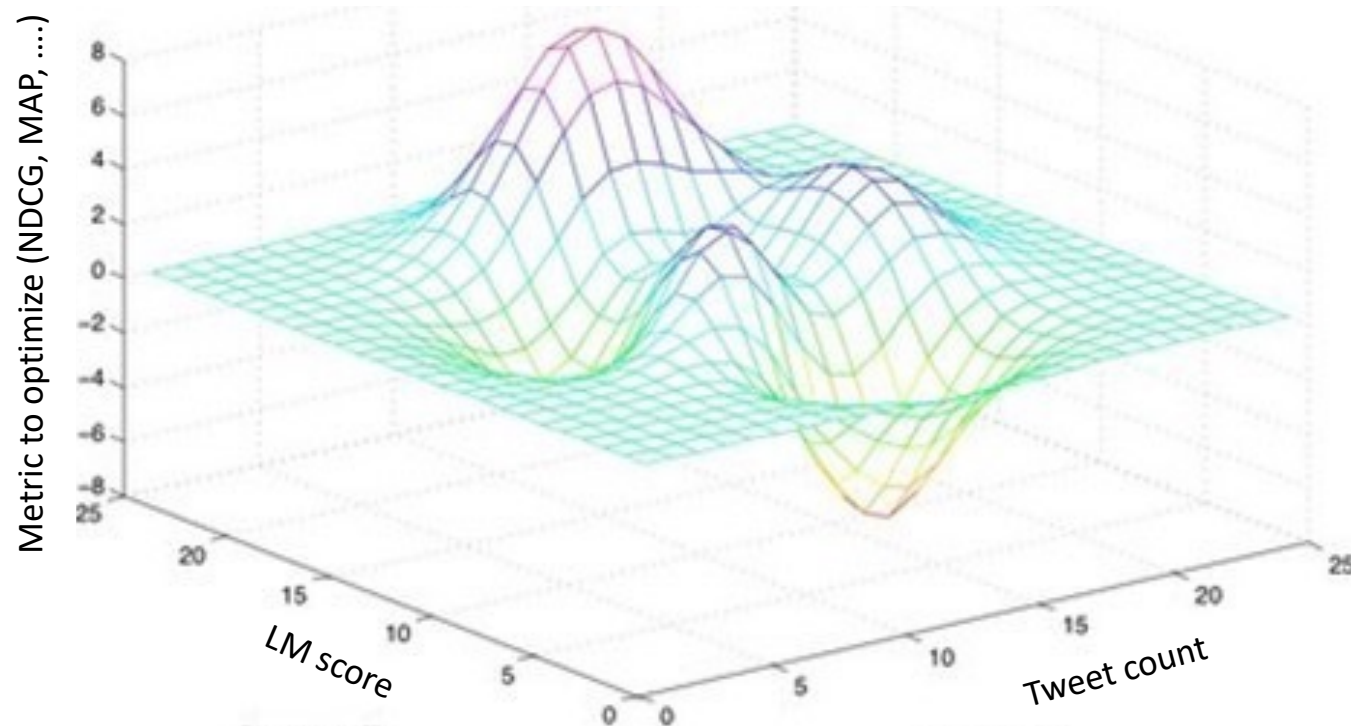
- The query plays the role of the triplet anchor and the two documents are the triple positive sample and the triplet negative sample

# Listwise approach

- Consider a number of ranking features.
- The linear model optimizes the order of the final rank.
- **The loss function is the MAP or the NDCG metric.**
  - Only models that have these objective functions can be used
  - For example, the weighted linear model:

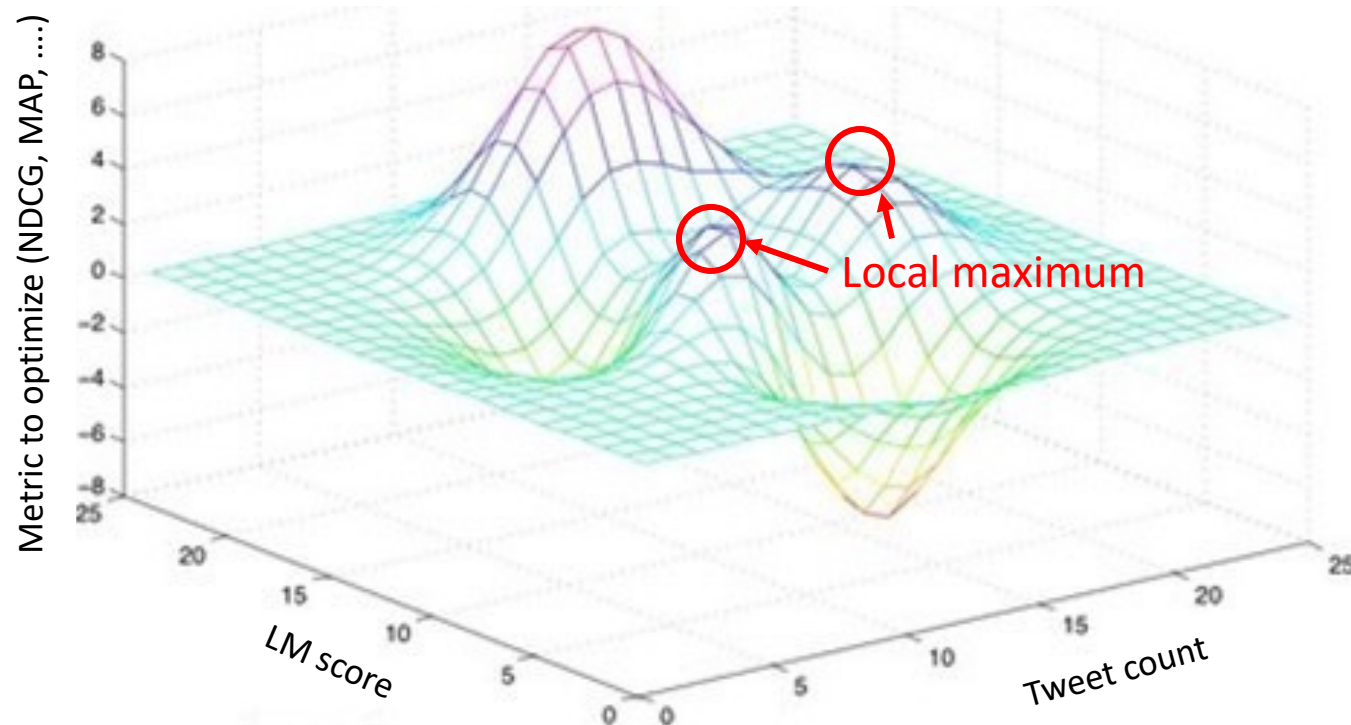
$$ReRanker(d) = w_1 s_1(d) + w_2 s_2(d) + \dots + w_n s_n(d)$$

# Listwise: Coordinate Ascent



- Find the weights for the features that maximize the metric to optimize
  - e.g.:  $LM\ score \times \mathbf{0.93} + user\ tweet\ count \times \mathbf{0.07}$

# Listwise: Coordinate Ascent



- Find the weights for the features that maximize the metric to optimize
  - e.g.:  $LM\ score \times \mathbf{0.93} + user\ tweet\ count \times \mathbf{0.07}$

# Example

- Now that we've learned what we can use to compute weights, let's apply them for fusion:

$$ReRanker(d) = \sum_i w_i s_i(d)$$

Doc	Tweet Desc. BM25	Tweet Desc. LM	User tweet count	Fusion Score
Weights	<b>0.5</b>	<b>0.4</b>	<b>0.1</b>	
D5	2.30* <b>0.5</b>	2.66* <b>0.4</b>	0.23* <b>0.1</b>	2.237
D4	1.80* <b>0.5</b>	1.59* <b>0.4</b>	2.02* <b>0.1</b>	1.738
D3	1.36* <b>0.5</b>	1.48* <b>0.4</b>	0.00* <b>0.1</b>	1.272
D1	0.00* <b>0.5</b>	0.72* <b>0.4</b>	1.92* <b>0.1</b>	0.480
D2	0.21* <b>0.5</b>	0.00* <b>0.4</b>	0.23* <b>0.1</b>	0.128

# Fitting LETOR in a live system

- Fetch >1000 candidates with each unsupervised retrieval model (fast over millions)
  - Filter with binary features (e.g. is retweet)
  - Filter with range features (e.g. timeframe or location)
- Rerank the >1000 candidates with the learning to rank model
  - Generate new features: e.g. time delta between the query and the document publication time
  - Binary, categorical features may not be ideal as a “direct input” for fusion

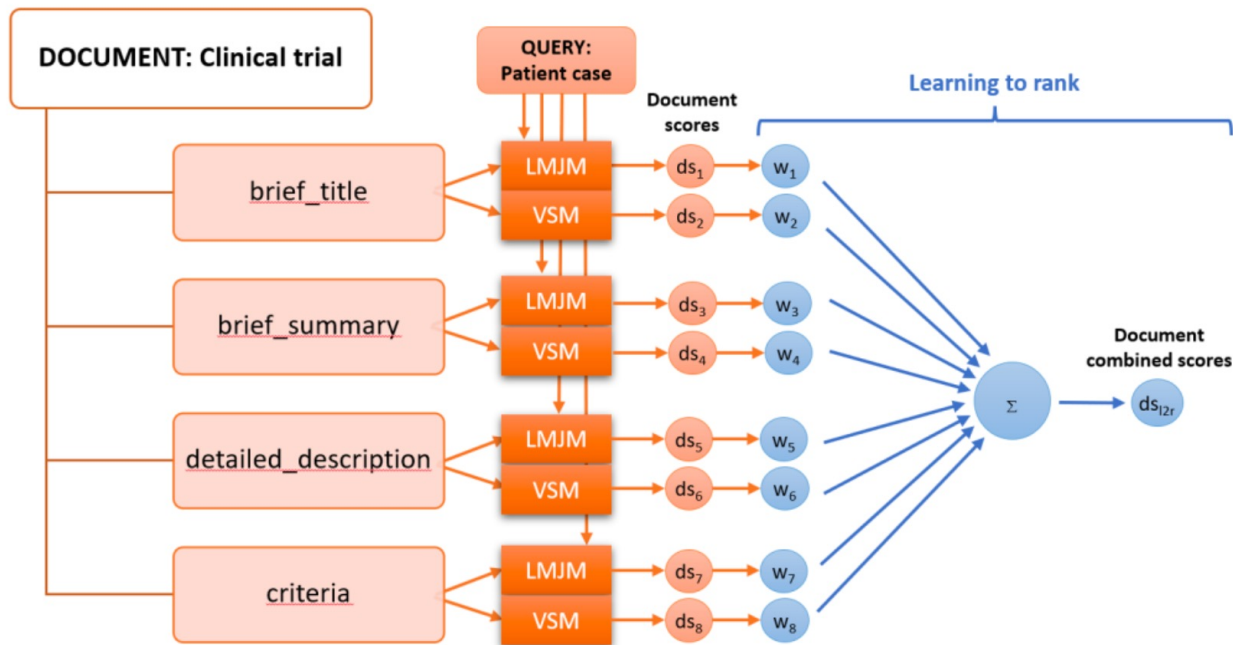


# Summary

- Combining ranks from multiple features can lead to better performance than the best individual rank;
- All approaches are still dependent on the quality of the features:
  - Be careful with binary, categorical or irrelevant features!
- Unsupervised approaches (e.g. RRF) may offer higher retrieval effectiveness than supervised approaches;
- Learning to rank works well for specific use-cases and with thousands or millions of examples (queries + relevant documents)

# Project – Phase 2

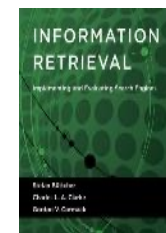
Do you remember this figure, from the project assignment?



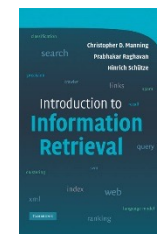
# Summary

- Unsupervised methods
  - Score-based methods (Comb\*)
  - Rank-based fusion (Bordafuse, Condorcet, Reciprocal Rank Fusion)
- Learning to Rank
  - Pointwise approach
  - Pairwise approach
  - Listwise approach

Section 11.1:



Section 15.4:



Based on the slides of Christopher D. Manning, Honglin Wang and Jiepu Jiang slides

## 2.1 Loss functions in learning to rank

Let  $\mathbf{x} = \{x_1, \dots, x_n\}$  be the objects to be ranked.<sup>3</sup> Suppose the labels of the objects are given as multi-level ratings  $\mathcal{L} = \{l(1), \dots, l(n)\}$ , where  $l(i) \in \{r_1, \dots, r_K\}$  denotes the label of  $x_i$  [11]. Without loss of generality, we assume  $l(i) \in \{0, 1, \dots, K-1\}$  and name the corresponding labels as  $K$ -level ratings. If  $l(i) > l(j)$ , then  $x_i$  should be ranked before  $x_j$ . Let  $\mathcal{F}$  be the function class and  $f \in \mathcal{F}$  be a ranking function. The optimal ranking function is learned from the training data by minimizing a certain loss function defined on the objects, their labels, and the ranking function. Several approaches have been proposed to learn the optimal ranking function.

In the *pointwise approach*, the loss function is defined on the basis of single objects. For example, in subset regression [5], the loss function is as follows,

$$L^r(f; \mathbf{x}, \mathcal{L}) = \sum_{i=1}^n (f(x_i) - l(i))^2. \quad (1)$$

In the *pairwise approach*, the loss function is defined on the basis of pairs of objects whose labels are different. For example, the loss functions of Ranking SVM [7], RankBoost [6], and RankNet [2] all have the following form,

$$L^p(f; \mathbf{x}, \mathcal{L}) = \sum_{s=1}^{n-1} \sum_{i=1, l(i) < l(s)}^n \phi(f(x_s) - f(x_i)), \quad (2)$$

where the  $\phi$  functions are hinge function ( $\phi(z) = (1 - z)_+$ ), exponential function ( $\phi(z) = e^{-z}$ ), and logistic function ( $\phi(z) = \log(1 + e^{-z})$ ) respectively, for the three algorithms.

In the *listwise approach*, the loss function is defined on the basis of all the  $n$  objects. For example, in ListMLE [16], the following loss function is used,

$$L^l(f; \mathbf{x}, y) = \sum_{s=1}^{n-1} \left( -f(x_{y(s)}) + \ln \left( \sum_{i=s}^n \exp(f(x_{y(i)})) \right) \right), \quad (3)$$

where  $y$  is a randomly selected permutation (i.e., ranked list) that satisfies the following condition: for any two objects  $x_i$  and  $x_j$ , if  $l(i) > l(j)$ , then  $x_i$  is ranked before  $x_j$  in  $y$ . Notation  $y(i)$  represents the index of the object ranked at the  $i$ -th position in  $y$ .