Learning to rank search results

Voting algorithms, rank combination methods

Web Search

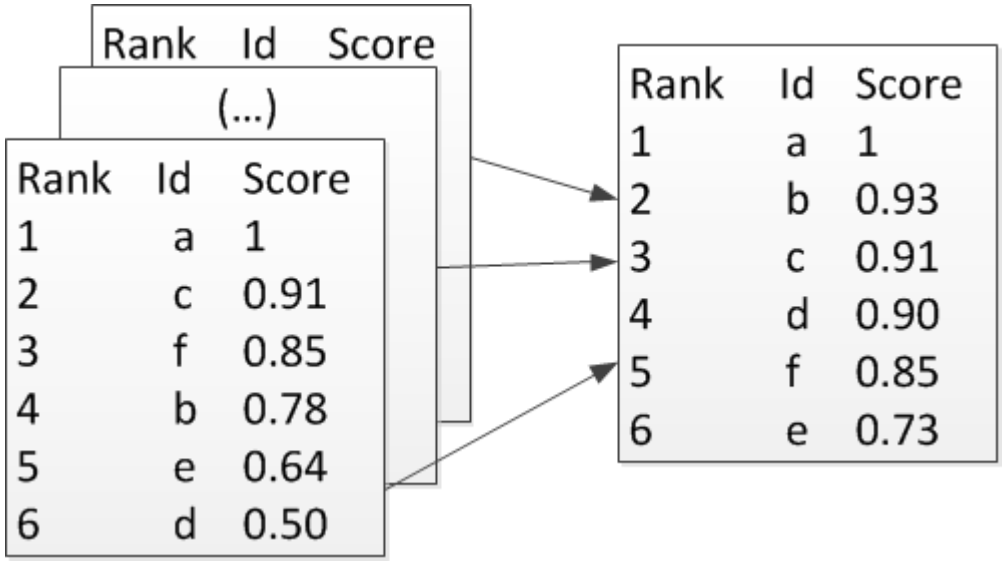
André Mourão, João Magalhães

			TREC45				GOV2			
			1998		1999		2004		2005	
Method	Stop	Stem	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	**0.464**	0.247	**0.500**	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	**0.464**	0.247	**0.500**	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	**0.204**	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	**0.204**	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	**0.464**	**0.204**	0.434	0.262	0.492	0.270	0.600	**0.343**
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	**0.204**	0.436	**0.262**	0.488	**0.272**	**0.602**	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

How can we merge these results?

- 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!



Standing on the shoulders of giants

- Vogt and Cottrell identified the following effects:
  - **Skimming Effect:** different retrieval models may retrieve different relevant documents for a single query;
  - **Chorus Effect:** potential for relevance is correlated with the number of retrieval models that suggest a document;
  - **Dark Horse Effect:** some retrieval models may produce more (or less) accurate estimates of relevance, relative to other models, for some documents.

C. Vogt, C. and G. Cottrell, *Fusion Via a Linear Combination of Scores.* Inf. Retr., 1999

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 $\mathrm{s_i}(d)$ and is placed on the $\mathrm{r_i(d)}$ position.
- Ranks are sorted by score.

Search-result fusion methods

- Unsupervised reranking methods
  - Score-based methods
    - Comb*
  - Rank-based fusion
    - Bordafuse
    - Condorcet
    - Reciprocal Rank Fusion (RRF)
- Learning to Rank

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.

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

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

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

Joon Ho Lee. Analyses of multiple evidence combination ACM SIGIR 1997.

CombSUM example

- CombSUM is used by Lucene 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 Lucene 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: $\frac{score - \mu}{\sigma}$

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

wComb*

- Lucene can also give higher/lower weight to scores from different fields

```
Query query = queryParserHelper.parse(queryString, "abstract");
query.setBoost(0.3f);
```

- These weights are then multiplied by the scores:

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

$$\text{w}CombMNZ(d) = |\{i | d \in Rank_i\}| \cdot wCombSUM(d)$$

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

CombMNZ

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

$$CombMNZ(d) = |\{i | d \in Rank_i\}| \cdot \sum_i s_i(d)$$

- 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 18th century
- 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				

Javed A. Aslam , Mark Montague, Models for metasearch, ACM SIGIR 2001

Borda fuse

- A voting algorithm based on the positions of the candidates.
  - Invented by Jean-Charles de Borda in 18th century
- 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				

Javed A. Aslam , Mark Montague, Models for metasearch, ACM SIGIR 2001

Borda fuse

- A voting algorithm based on the positions of the candidates.
  - Invented by Jean-Charles de Borda in 18th 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	**10**
D5	4	4	1	9
D1	0	1	3	4
D3	2	2	0	4
D2	1	0	2	3

Javed A. Aslam , Mark Montague, Models for metasearch, ACM SIGIR 2001

Condorcet

- Voting algorithm that started as a way to select the best candidate on an election
  - Marquis de Condorcet, also in 18th 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.

Mark Montague and Javed A. Aslam. Condorcet fusion for improved retrieval. ACM CIKM 2002.

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					

				Win,	Draw,	Lose
Tweet Desc. BM25:	D2 > D1					
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

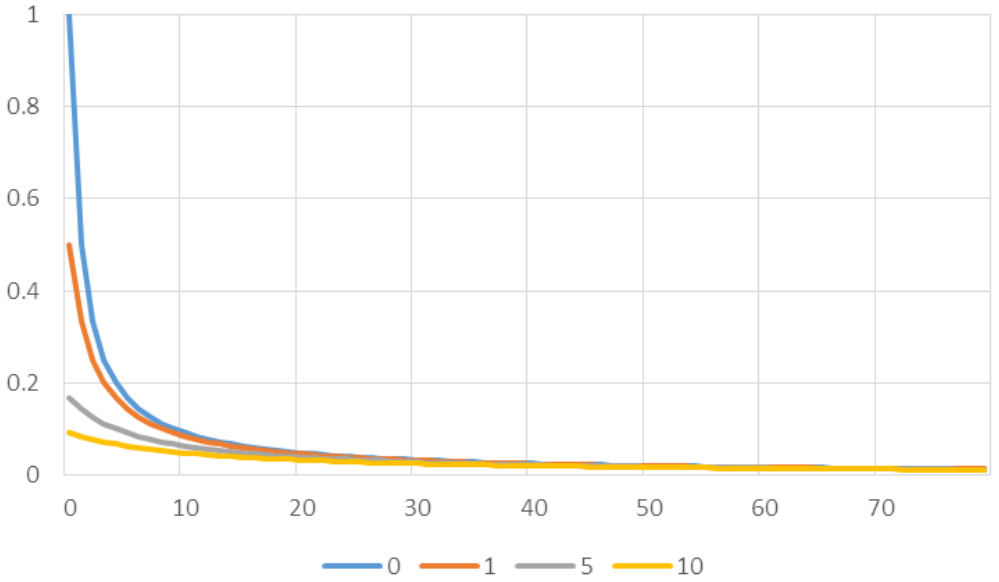
	Win	Tie	Lose	Score
D4	10	0	2	**8**
D5	9	0	3	6
D3	4	0	8	-4
D1	4	0	8	-4
D2	4	0	8	-4

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



Gordon Cormack, Charles LA Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. ACM SIGIR 2009.

RRF example

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

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

RRF example

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

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

RRF example

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

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

Experimental comparison

	TREC45				Gov2			
	1998		1999		2005		2006	
Method	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	**0.239**	0.454	0.249	**0.567**	0.277	**0.588**	0.314
RRF	**0.462**	0.215	**0.464**	**0.252**	0.543	**0.297**	0.570	**0.352**
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			**0.588**	0.309
RankSVM			0.420	0.234			0.556	0.268

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?

What is Learning to Rank (LETOR)?

- Use machine learning techniques to learn a function automatically to rank results effectively

- Pointwise approaches
  - regress the relevance score, classify docs into *Relevant* and *Non Rel*
- Pairwise approaches
  - given two documents, predict partial ranking: $d_1 > d_2$ or $d_2 > d_1$
- Listwise approaches
  - given two ranked list of the same items, which is better?

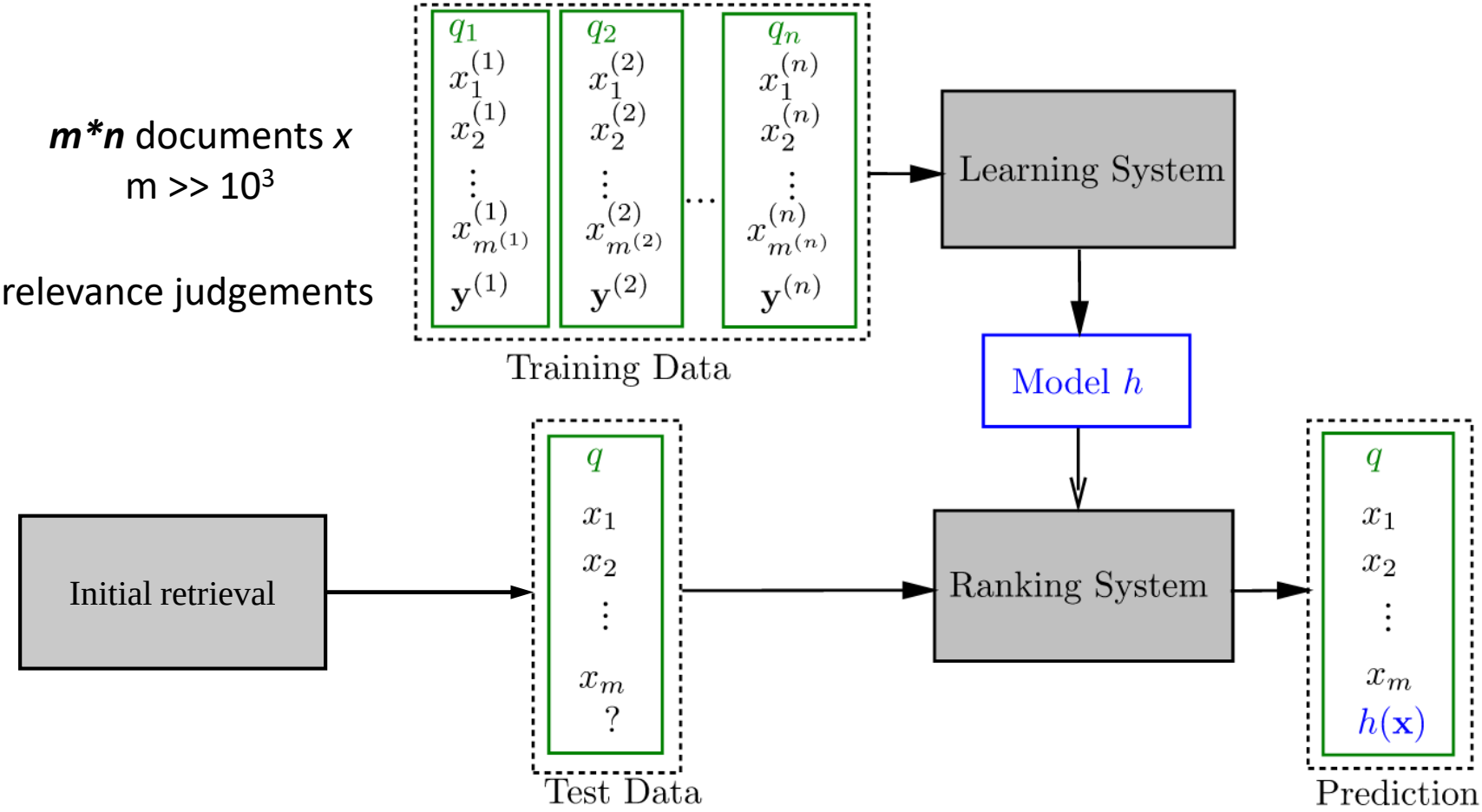
https://sourceforge.net/p/lemur/wiki/RankLib/

LETOR Experimental setup

n queries *q*, *n* >> 10^3

m*n documents *x*
m >> 10^3

y: relevance judgements



h(x): predicted relevance

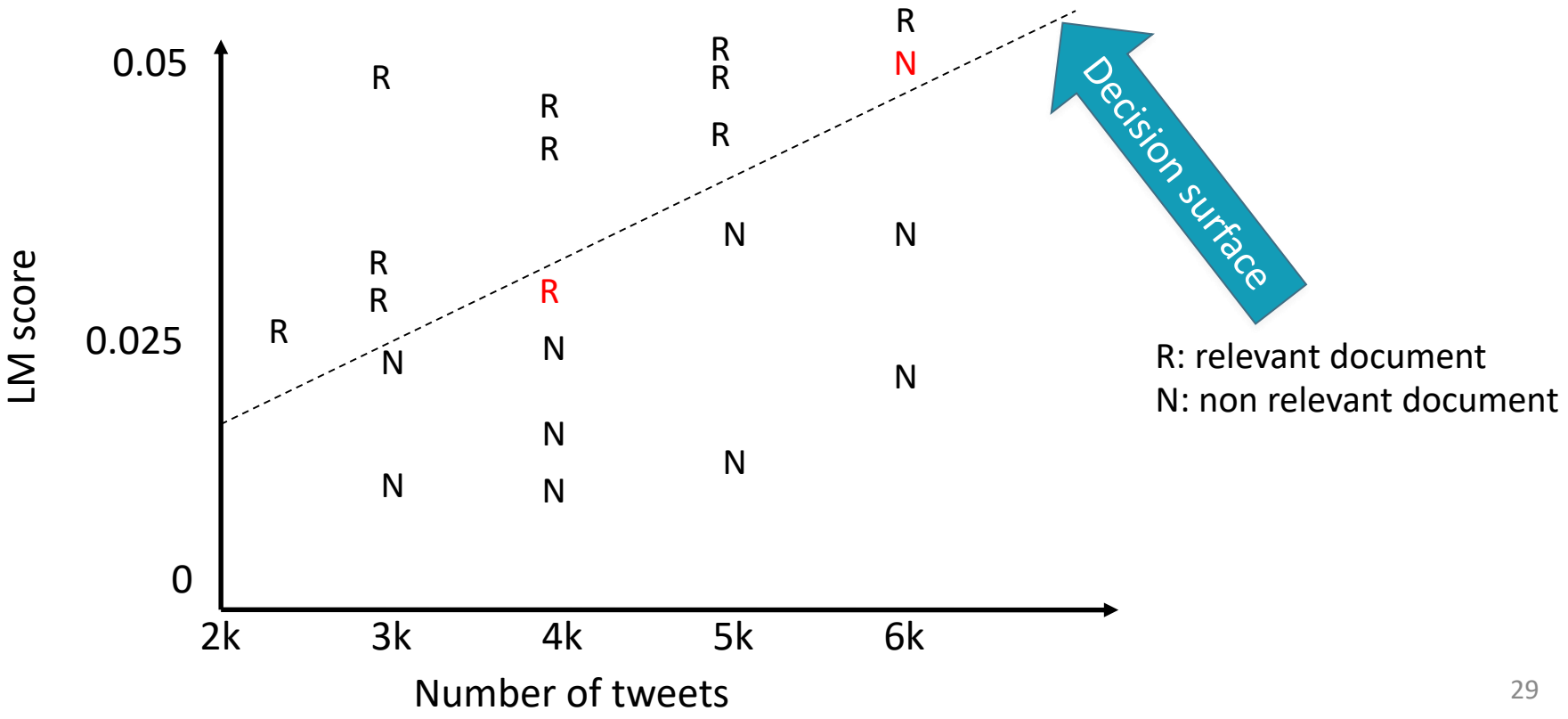
Learning to rank features

Table 10.4 Features for the "Gov2" corpus

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

- Collect a training corpus of (q, d, r) triples
- Train a machine learning model to predict the class *r* of a document-query pair



Pairwise approach

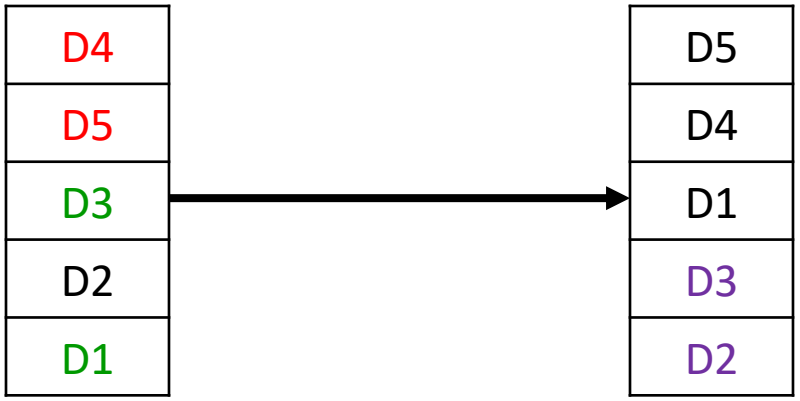
- Find a global order by predicting partial ranking of the documents:

D4
D5
D3
D2
D1

Misordered pairs: 2

Pairwise approach

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



Misordered pairs: 2

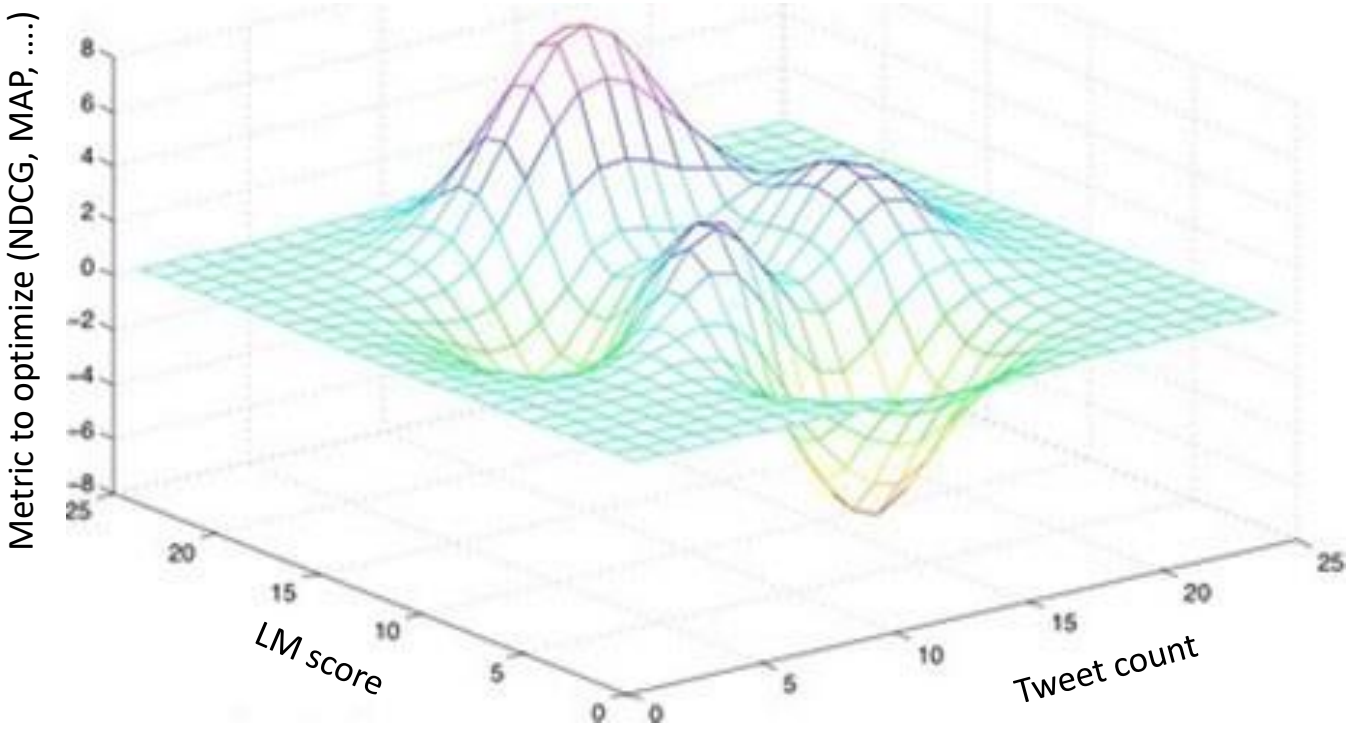
Misordered pairs: 1

Listwise approach

- Consider a number of ranking features.
- The ranking model is a weighted linear model.
- The linear model optimizes the order of the final rank.

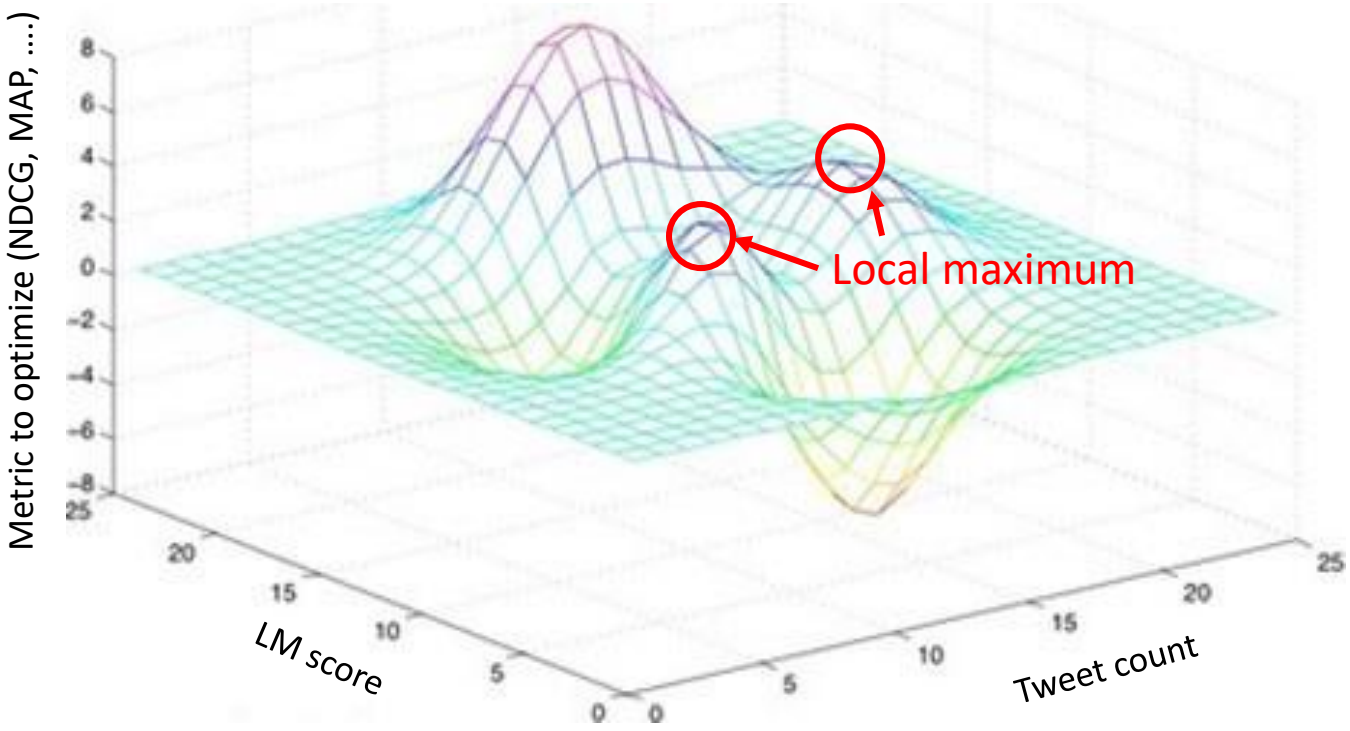
$$ReRanker(d) = w_1\, s_1(d) + w_2\, s_2(d) + \ldots + w_n\, s_n(d)$$

Listwise: Coordinate Ascent



- Find the weights for the features that maximize the metric to optimize
  - e.g.: *LM score* x **0.93** + *user tweet count* x **0.07**

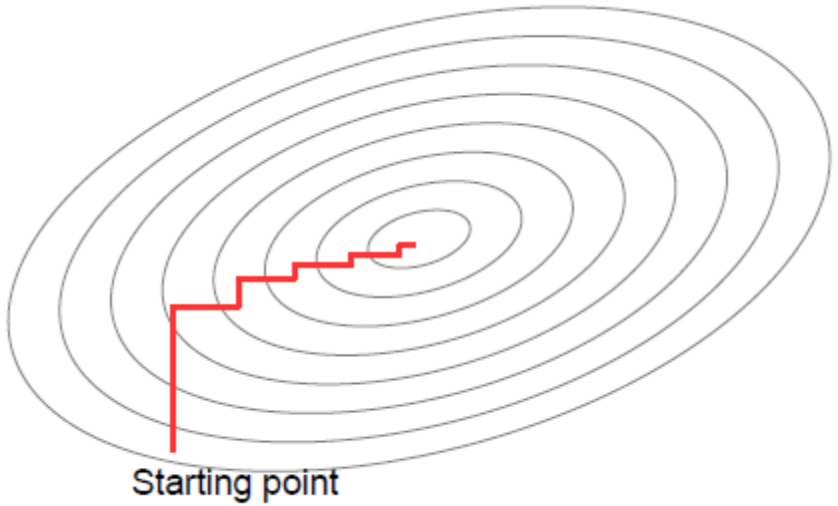
Listwise: Coordinate ascent



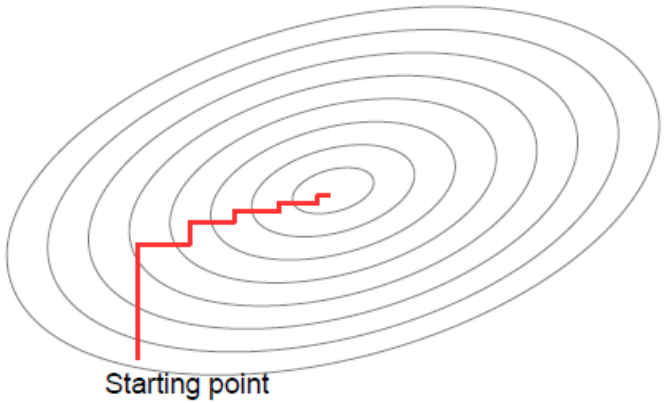
- Find the weights for the features that maximize the metric to optimize
  - e.g.: *LM score* x **0.93** + *user tweet count* x **0.07**

Coordinate Ascent

- Coordinate descent algorithm performs successive line searches along the axes.



Algorithm




```
for iter_descent = 1:100
   for rank = 1: rank_total
          for iter = 1:100
                 for i,j where r(i,j) !=0
                        update rank weight
                 end
          end
   end
end
```

Example

- Now that we've learned what we can use to compute weights, lets 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	**0.5**	**0.4**	**0.1**	
D5	2.30***0.5**	2.66***0.4**	0.23***0.1**	2.237
D4	1.80***0.5**	1.59***0.4**	2.02***0.1**	1.738
D3	1.36***0.5**	1.48***0.4**	0.00***0.1**	1.272
D1	0.00***0.5**	0.72***0.4**	1.92***0.1**	0.480
D2	0.21***0.5**	0.00***0.4**	0.23***0.1**	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 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!

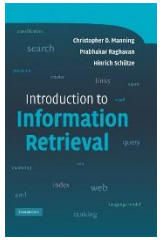
- Unsupervised approaches (e.g. RRF) can offer higher retrieval effectiveness than supervised approaches;
  - Learning to rank works well for specific use-cases and with thousands or millions of examples (queries + relevant documents)

Summary

- Unsupervised methods
  - Comb*
  - Bordafuse
  - Condorcet
  - Reciprocal Rank
- Learning to Rank

Section 11.1:

Section 15.4:



Some slides are derived from Christopher D. Manning, Honglin Wang and Jiepu Jiang slides