INFORMATION RETRIEVAL

L09. LEARNING TO RANK

SUZAN VERBERNE 2022

Universiteit Leiden

REFLECTION ON CRITICAL REVIEWS

- I hope you got a taste of the range of research in the field
- SIGIR is more than IR
  - and IR is more than this course can cover
  - E.g. there is quite some work on recommender systems published at IR conferences
- I also hope you recognized some of the topics from the first half of the course in the paper you studied
  - Evaluation metrics?
  - A BM25 baseline?
  - BERT-based encoders?

TODAY'S LECTURE

- Presentation by group 38
- Machine learning for IR
- Learning to rank strategies
- Query and document representations
- Learning from implicit feedback

INTRODUCTION TO RANK LEARNING

RELEVANCE CRITERIA

- Query-document similarity (content overlap) is central in IR:
  - Word-based: BM25, LM
  - Embeddings-based: neural rankers (e.g. ColBERT)
- But similarity is not the only relevance criterion:
  - Document popularity on the web (PageRank)
  - Document popularity in the search engine
  - Source (domain)
  - Recency
  - …
- Multiple features for one document -> learn a ranking

LEARN A RANKING

- Given
  - A set of queries
  - A set of 100 documents pre-retrieved for each query
  - Relevance assessments (binary) for all these query-document pairs
  - We want to re-rank the documents so that the relevant documents are on top of the list
  - Each query-document pair is represented by a vector
- How do we learn a ranking model? (learning problem, optimization)
- How do we apply the model to rank documents for a new query?
- Discuss with your neighbour

LEARN A RANKING

		f_1	f_2	f_3	…	f_k	label
Q_1	D_1						0
	D_2						1
	…						…
	D_n						0
Q_2	D_1						1
	…						…
	D_n						0

LEARN A RANKING

What did you come up with?

1. Learn a (probabilistic) classifier or regression model on query-document pairs with relevance labels
2. Apply to unseen pairs, get a score for each query-document pair
3. Rank documents per query by prediction score

➢ We see later how that works and what alternatives we have

MACHINE LEARNING FOR IR

IIR CHAPTER 15, SECTION 15.4

CLASSIFICATION FOR RELEVANCE

Simplest approach, as explained in the IIR book:

- IR can be considered a binary classification problem with labels ‘relevant’ and ‘nonrelevant’
- Toy example with 2 features:
  - (1) the vector space cosine similarity (α) between query and document
  - (2) the minimum window width $\boldsymbol{\omega}$ within which the query terms lie

Example	DocID	Query	Cosine score	ω	Judgment
Φ_1	37	linux operating system	0.032	3	*relevant*
Φ_2	37	penguin logo	0.02	4	*nonrelevant*
Φ_3	238	operating system	0.043	2	*relevant*
Φ_4	238	runtime environment	0.004	2	*nonrelevant*

IIR table 15.3

CLASSIFICATION FOR RELEVANCE

- Simple linear classifier:

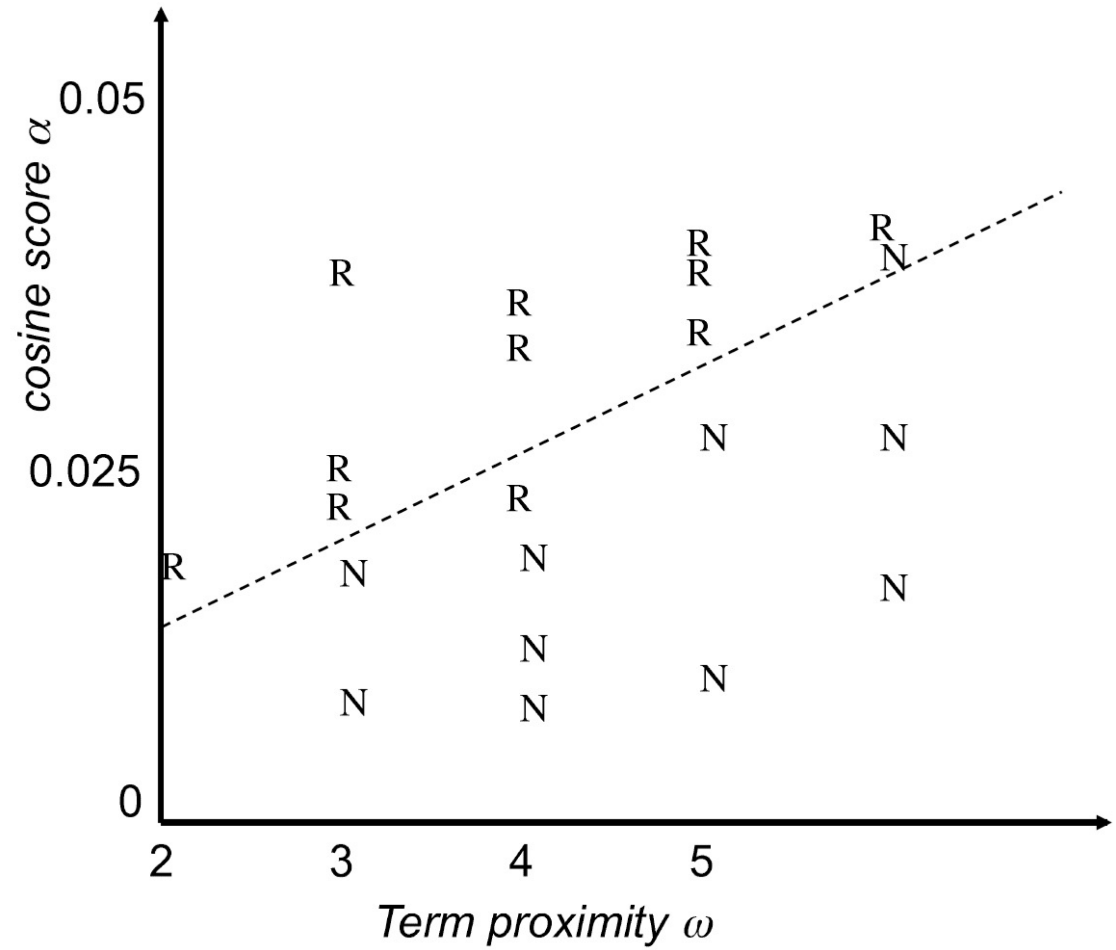
$$Score\ (d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c$$

- with the coefficients a, b, c to be learned from the training data

CLASSIFICATION FOR RELEVANCE

Thresholding:

- Pick a threshold value $\boldsymbol{\tau}$
- If $Score(\alpha, \omega) > \tau$ we declare the document to be relevant, else we declare the document to be nonrelevant
- All points that satisfy $Score(\alpha, \omega) = \tau$ form a line
- We have a linear classifier that separates relevant from nonrelevant instances



IIR figure 15.7

MACHINE LEARNING FOR RANKING

- Supervised machine learning problems:
  - classification problems: predict a categorical variable
  - regression problems: predict a numerical variable

- Ranking is different from classification or regression
  - documents are grouped per query
  - for ranking their relative relevance matters, not their absolute

LEARNING TO RANK APPROACHES

MITRA & CRASWELL CHAPTER 5

STRATEGIES FOR LEARNING A RANKING

- Strategies are loss functions
  - This is largely independent of the model or the features
  - In all strategies: each document gets a score given a query, then we sort the documents by those scores
  - The loss function is used during learning/optimization

- The strategy that we discussed earlier is called pointwise learning
  - learning the relevance value per query-document pair

TRAINING APPROACHES

- Pointwise: learning a ranking from individual (q, d, rel) pairs
  - Train a model for the labels $rel \in \{0,1\}$
  - Optimize for the difference between the true and assigned score
  - Example loss function: regression loss

$$\mathcal{L}_{squared} = \left\| rel_q(d) - score(q,d) \right\|^2 \qquad \text{(eq. 5.1)}$$

  - Then average the loss over all query-document pairs:

$$\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_{squared}$$

  - On inference, let the classifier assign a score to each (q, d) pair and rank the documents for each query

POINTWISE LEARNING

	True: $rel_q(d)$	Model A output: $score_A(q,d)$	$\mathcal{L}_{squared,A}$	Model B output: $score_B(q,d)$	$\mathcal{L}_{squared,B}$
d1	1	0.2		0.9	
d2	0	0.3		0.5	
d3	0	0.1		0.5	
d4	0	0.1		0.5	
d5	0	0.1		0.5	
		Average Loss:			

- Example of regression loss for pointwise learning

POINTWISE LEARNING

	True: $rel_q(d)$	Model A output: $score_A(q,d)$	$\mathcal{L}_{squared,A}$	Model B output : $score_B(q,d)$	$\mathcal{L}_{squared,B}$
d1	1	0.2	0.64	0.9	0.01
d2	0	0.3	0.09	0.5	0.25
d3	0	0.1	0.01	0.5	0.25
d4	0	0.1	0.01	0.5	0.25
d5	0	0.1	0.01	0.5	0.25
		Average Loss:	**0.152**		**0.202**

$$\mathcal{L}_{squared} = \left\| rel_q(d) - score(q,d) \right\|^2$$

POINTWISE LEARNING

	True: $rel_q(d)$	Model A output: $score_A(q,d)$	$\mathcal{L}_{squared,A}$	Model B output: $score_B(q,d)$	$\mathcal{L}_{squared,B}$
d1	1	0.2	0.64	0.9	0.01
d2	0	0.3	0.09	0.5	0.25
d3	0	0.1	0.01	0.5	0.25
d4	0	0.1	0.01	0.5	0.25
d5	0	0.1	0.01	0.5	0.25
		Average Loss:	**0.152**		**0.202**

- Limitation of pointwise learning: loss function does not consider relative ranking between items in the same list, only absolute numbers
- We do not need to estimate the true relevance label as long as our model ranks the relevant documents over the nonrelevant documents

TRAINING APPROACHES

- Pairwise:
  - Consider pairs of relevant and nonrelevant documents for the same query
  - Minimize the number of incorrect inversions in the ranking
    - i.e., d_i is more relevant for q than d_j but d_j is ranked higher than d_i
  - Example loss function: hinge loss

$$\mathcal{L}_{hinge} = \max\left(0{,}1 - \left(score(q, d_i) - score\left(q, d_j\right)\right)\right)$$

  - Then sum the loss over all pairs $\left(d_i, d_j\right)$ with d_i more relevant than d_j:

$$\sum_{y(d_i)>y(d_j)} \mathcal{L}_{hinge}$$

PAIRWISE LEARNING

	True: $rel_q(d)$	Model A output: $score_A(q,d)$	$\mathcal{L}_{hinge,A}$	Model B output: $score_B(q,d)$	$\mathcal{L}_{hinge,B}$
d1	1	0.2		0.9	
d2	0	0.3		0.5	
d3	0	0.1		0.5	
d4	0	0.1		0.5	
d5	0	0.1		0.5	
		Sum Loss:			

PAIRWISE LEARNING

	True: $rel_q(d)$	Model A output: $score_A(q,d)$	$\mathcal{L}_{hinge,A}$	Model B output: $score_B(q,d)$	$\mathcal{L}_{hinge,B}$
d1	1	0.2		0.9	
d2	0	0.3		0.5	
d3	0	0.1		0.5	
d4	0	0.1		0.5	
d5	0	0.1		0.5	
		Sum Loss:			

- $\max(0, 1-(0.2-0.3)) = 1.1$
- $\max(0, 1-(0.2-0.1)) = 0.9$
- etc. for all pairs (d_i, d_j) with d_i more relevant than d_j

PAIRWISE LEARNING

	True: $\boldsymbol{rel_q(d)}$	Model A output: $score_A(q,d)$	$\mathcal{L}_{hinge,A}$	Model B output: $score_B(q,d)$	$\mathcal{L}_{hinge,B}$
d1	1	0.2		0.9	
d2	0	0.3	1.1	0.5	0.6
d3	0	0.1	0.9	0.5	0.6
d4	0	0.1	0.9	0.5	0.6
d5	0	0.1	0.9	0.5	0.6
		Sum Loss:	3.8		**2.4**

- $\max(0, 1 - (0.2 - 0.3)) = 1.1$
- $\max(0, 1 - (0.2 - 0.1)) = 0.9$
- etc. for all pairs (d_i, d_j) with d_i more relevant than d_j

PAIRWISE LEARNING

Successful methods for pairwise learning

- RankSVM (Joachims et al. 2002)
- RankNet (Burges et al., 2005)
  - has been a popular choice for training neural ranking models
  - was an industry favourite for web search for many years

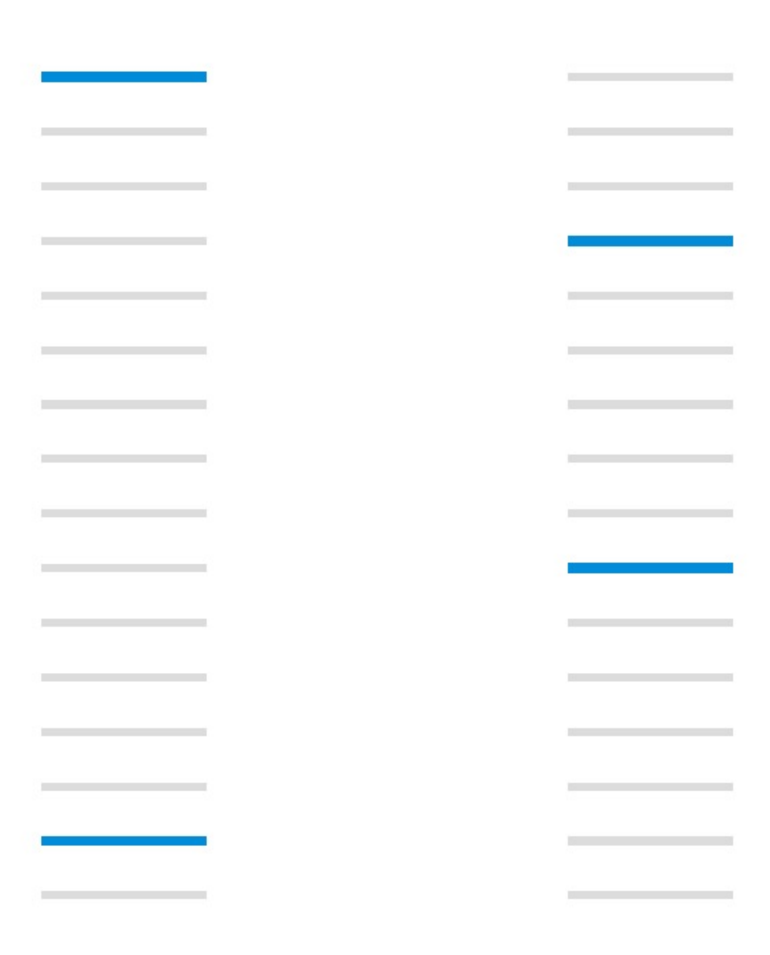
(see Mitra & Craswell p. 59 for more details)

Universiteit Leiden

Joachims (2002). Optimizing search engines using clickthrough data
Burges et al. (2005). Learning to rank using gradient descent

PAIRWISE LEARNING

- Limitation of pairwise learning: every document pair is treated equally important, but misrankings in higher positions are more severe than misrankings in lower positions
- Left: 13 pairwise errors
- Right: 11 pairwise errors
- But for IR metrics like nDCG or MAP, the left ranking is preferable



Burges, C. J. (2010). From RankNet to LambdaRank to LambdaMART: An overview. *Learning*, *11*(23-581), 81.

Universiteit Leiden

RANK-BASED OPTIMIZATION

- Idea: a loss function that optimizes for the order of the complete ranking
- Solution: add an evaluation metric, e.g. DCG, to the loss function

$$DCG(L) = r_1 + \sum_{i=2}^{n} \frac{r_i}{\log_2 i}$$

RANK-BASED OPTIMIZATION

- LambdaRank: a pairwise method
  - but with the addition of nDCG to the loss
  - multiplying the RankNet loss by the size of the change in nDCG given by swapping the rank positions

$$\lambda_{LambdaRank} = \lambda_{RankNet} \cdot |\Delta nDCG|$$

  - It was shown empirically that this optimizes nDCG directly

QUERY AND DOCUMENT REPRESENTATIONS

FEATURE-BASED LEARNING TO RANK

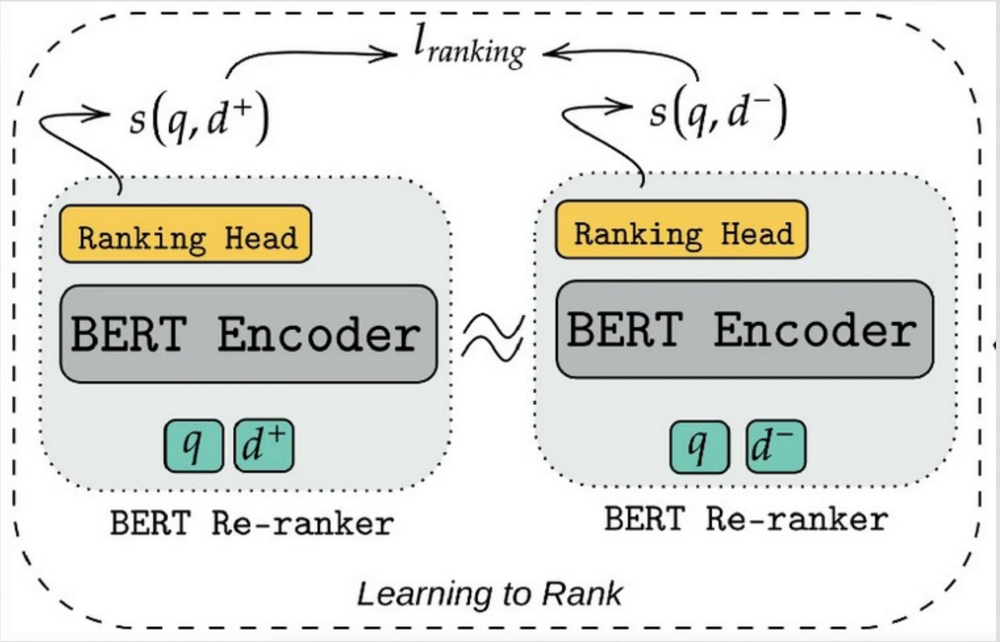
- LtR models can be used to combine multiple relevance criteria
  - Features:
    - Query-independent features (e.g., PageRank, document length)
    - Query-dependent features (e.g., BM25, cosine similarity)
    - Query-level features (e.g., query length, iDFs of query terms)
    - Much more: http://research.microsoft.com/en-us/projects/mslr/feature.aspx

- Or to combine multiple rankers
  - Get a score for each ranker, then supervised re-ranking

EMBEDDINGS-BASED RANK LEARNING

- Pairwise ranking loss can also be added on top of BERT-based encoders to learn a ranking
- Contrastive loss:
  - used for representation learning
  - minimizes the distance between a relevant pair, while increasing the distance between a non-relevant pair

(see Mitra and Craswell, p. 54)



Abolghasemi, A., Verberne, S., Azzopardi, L. (2022). Improving BERT-based Query-by-Document Retrieval with Multi-task Optimization

LEARNING FROM IMPLICIT FEEDBACK

CREDITS TO HARRIE OOSTERHUIS

LEARNING FROM INTERACTION DATA

- What if we don't have explicit relevance assessments?
- Solution: use interaction data in the search engine instead
  - Interactions are virtually free if you have users
  - User behaviour is indicative of their preferences
  - = Implicit feedback
- Assumption: when someone clicks on a result, it is relevant to them

LEARNING FROM INTERACTION DATA

- Implicit feedback is noisy:
  - A non-relevant document might be clicked
  - A relevant document might not be clicked

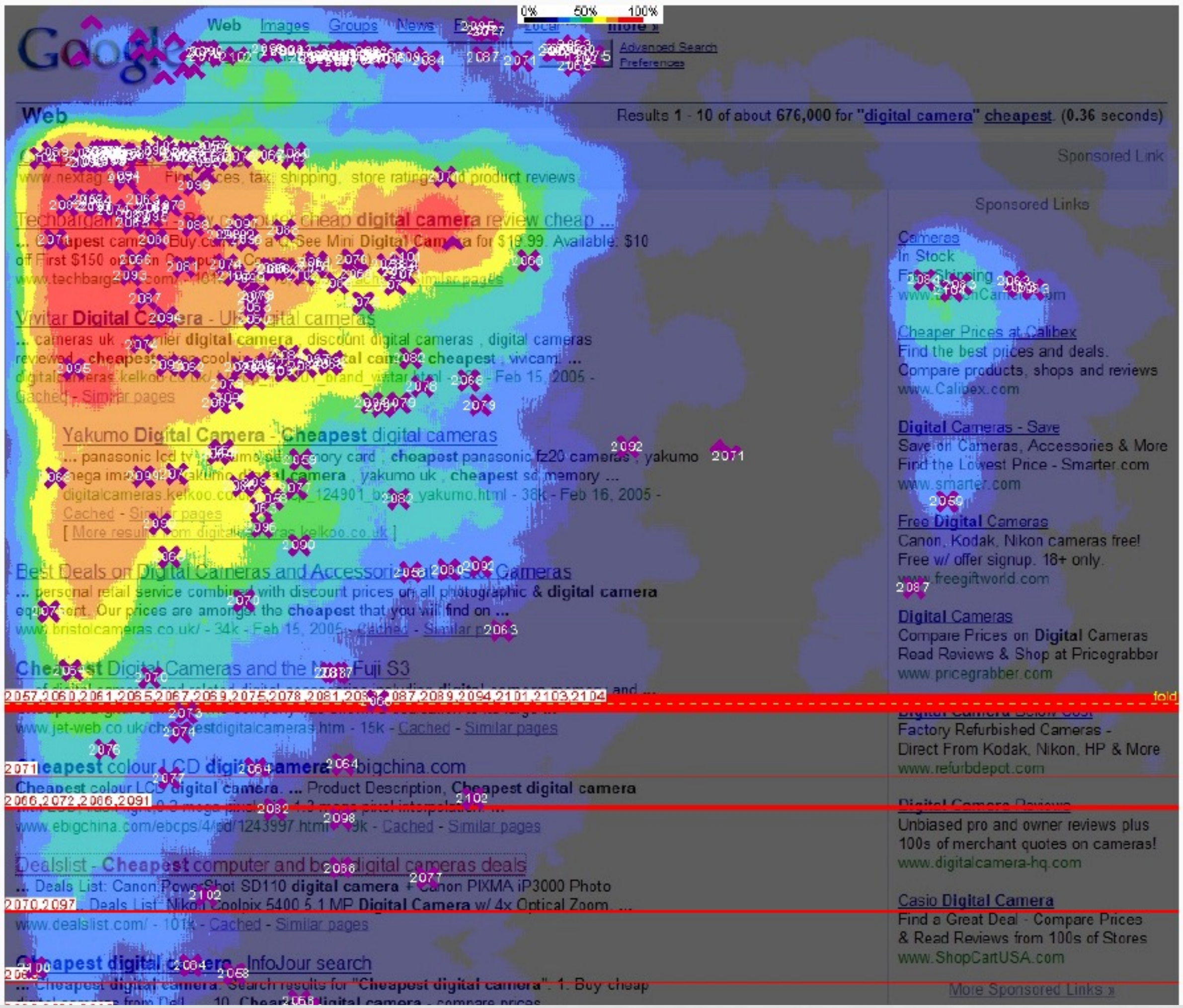
- Implicit feedback is biased: clicks for reasons other than relevance
  - Position bias: higher ranked documents get more attention.
  - Selection bias: interactions are limited to the presented documents.
  - Presentation bias: results that are presented differently will be treated differently

LEARNING FROM INTERACTION DATA

- What is the interpretation of a non-click?
  - Either the document didn't seem relevant to the user
  - Or the user did not see (observe) the document

- Generally, the lower in the list, the lower the chance that the document is observed by the user

LEARNING FROM INTERACTION DATA

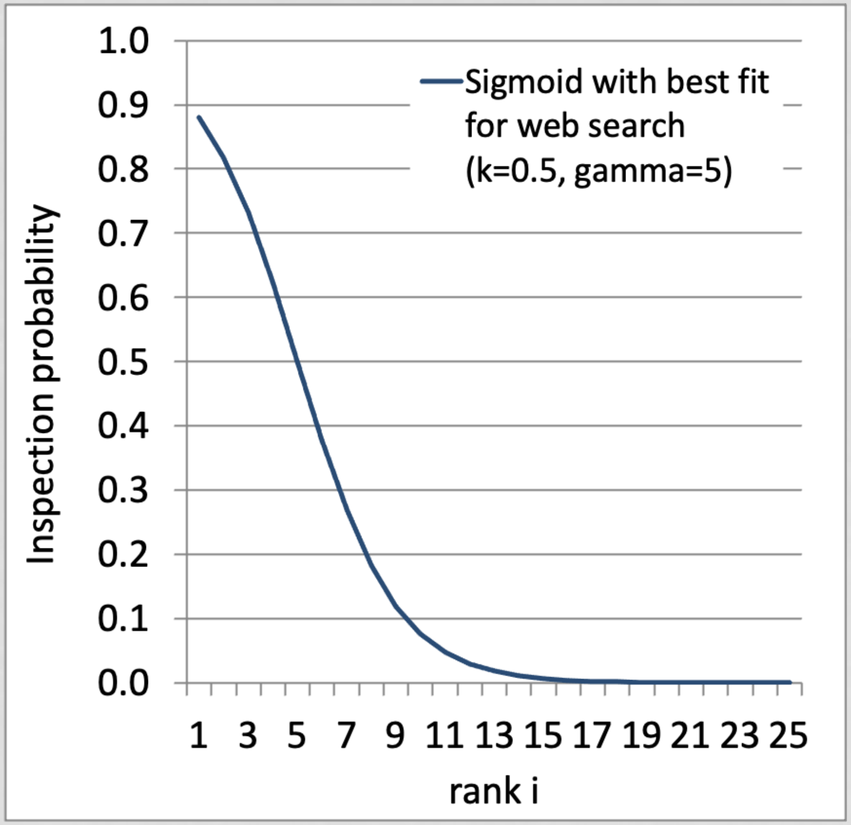


- Eye-tracking data: where do users look on the result page?

POSITION BIAS

- Position bias:
  - Documents lower in the list have a smaller probability to be observed
  - hence a smaller probability to be clicked
  - hence a smaller probability to be recorded as being relevant

- We want to learn relevance preferences and not the bias



PROBABILISTIC MODEL OF USER CLICKS

- Joachims et al. (2017): model user clicks as:

$$P(clicked(d)|relevance(d), position(d)) = \\ P(clicked(d)|relevance(d), observed(d)) \times \\ P(observed(d)|position(d))$$

- Ranking should be based on the unbiased part:

$$P(clicked(d)|relevance(d), observed(d))$$

- This can be estimated from clicks if we know the effect of position bias:

$$P(observed(d)|position(d))$$

Universiteit Leiden

Joachims et al. (2017). Unbiased learning-to-rank with biased feedback

ESTIMATION OF POSITION BIAS

- How to measure the effect of the position bias?
- Idea: if we change the position of a document then we don't change its relevance. So all changes in click behaviour come from the position bias
- Measure the position bias by intervention in the ranking:
  1. swap two documents in the ranking
  2. present the modified ranking to some users (A/B test)
  3. record the clicks on the document in both original and modified rankings
  4. measure the probability of a document being observed based on the clicks
- This can be done in environments with large numbers of interactions (e.g. Google, Bol.com, Amazon.com)

CORRECT FOR POSITION BIAS

- Inverse Propensity Scoring (IPS) estimators can remove bias
- Propensity of observation = probability that a document is observed by the user
- Main idea: weight clicks depending on their observation probability
  - Clicks near the top of the ranked list: have high observation probability, get assigned small weight
  - Clicks near the bottom of the ranked list: have low observation probability , get assigned large weight

Joachims et al. (2017). Unbiased learning-to-rank with biased feedback

CORRECT FOR POSITION BIAS

$$\mathcal{L}_{IPS}(f_\theta, D, c) = \sum_{d_i \in D} \frac{\lambda\big(rank(d_i|f_\theta, D)\big)}{P(o_i = 1|R, d_i)} \cdot c_i$$

- $\lambda\left(rank(d_i|f_\theta, D)\right)$: score function based on rank of document d_i by ranker f_θ on the document set D

divided by

- $P(o_i = 1|R, d_i)$: probability that d_i is observed in ranking R

- c_i: observed click of the document in the search engine log (0 or 1)

Joachims et al. (2017). Unbiased learning-to-rank with biased feedback

EFFECT OF PROPENSITY BASED LEARNING

- Joachims et al. did a real-world experiment
  - trained the model on real-world click logs
  - deployed it in Google Scholar

Table 1: Per-query balanced interleaving results for detecting relative performance between the hand-crafted production ranker used for click data collection (Prod), Naive SVM-Rank and Propensity SVM-Rank.

Interleaving Experiment	Propensity SVM-Rank wins	Propensity SVM-Rank loses	Propensity SVM-Rank ties
against Prod	87	48	83
against Naive SVM-Rank	95	60	102

Evaluation method where results from multiple rankers are combined in one list and clicks are recorded as votes

Universiteit Leiden

Joachims et al. (2017). Unbiased learning-to-rank with biased feedback

CONCLUSIONS

HOMEWORK (1)

- Read:
  - IIR book: section 15.4
  - Mitra and Craswell, Introduction to Neural IR, chapter 5

HOMEWORK (2)

- IIR exercise 15.7, 15.8, 15.9
- A computation and comparison of pointwise and pairwise loss for two example rankings
- See Brightspace -> assignments
- Deadline: Sunday April 24, 23.59

AFTER THIS LECTURE…

- You can explain the procedure for pointwise ranking
- You can explain the procedure for pairwise ranking
- You can compute regression loss for pointwise ranking
- You can compute hinge loss for pairwise ranking
- You can explain the rationale behind rank-based optimization
- You can list three types of bias in interaction data
- You can explain how position bias can be estimated through interventions
- You can explain inverse propensity scoring on a conceptual level