Learning to Rank

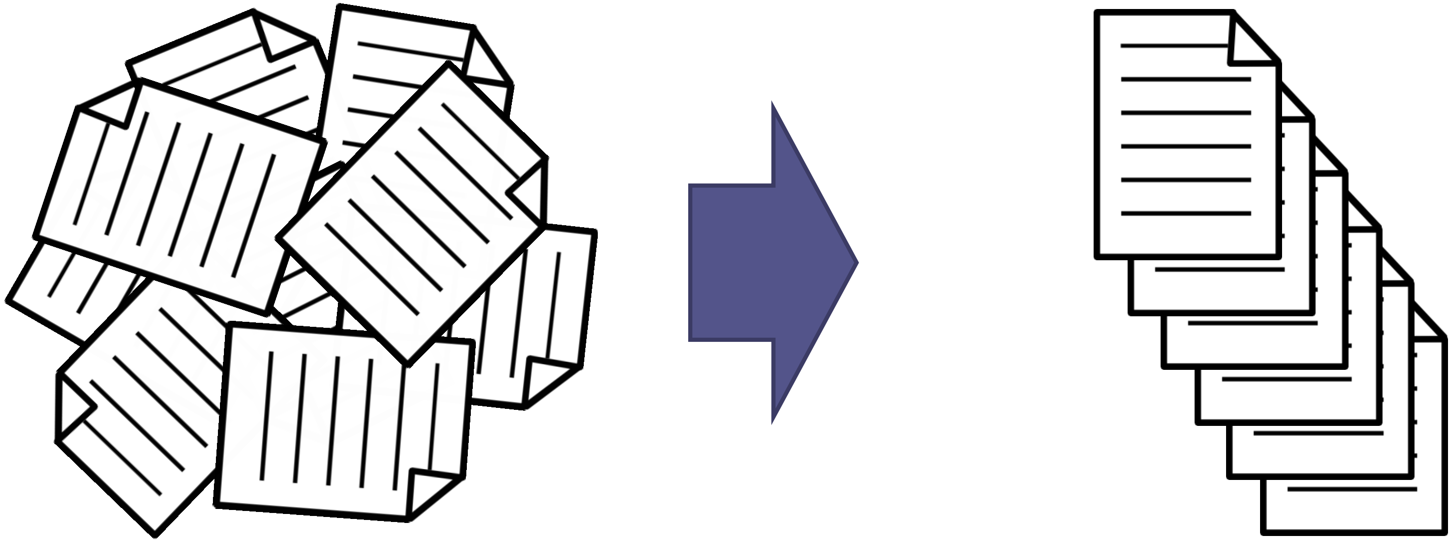
Karel Horák

Introduction to Ranking

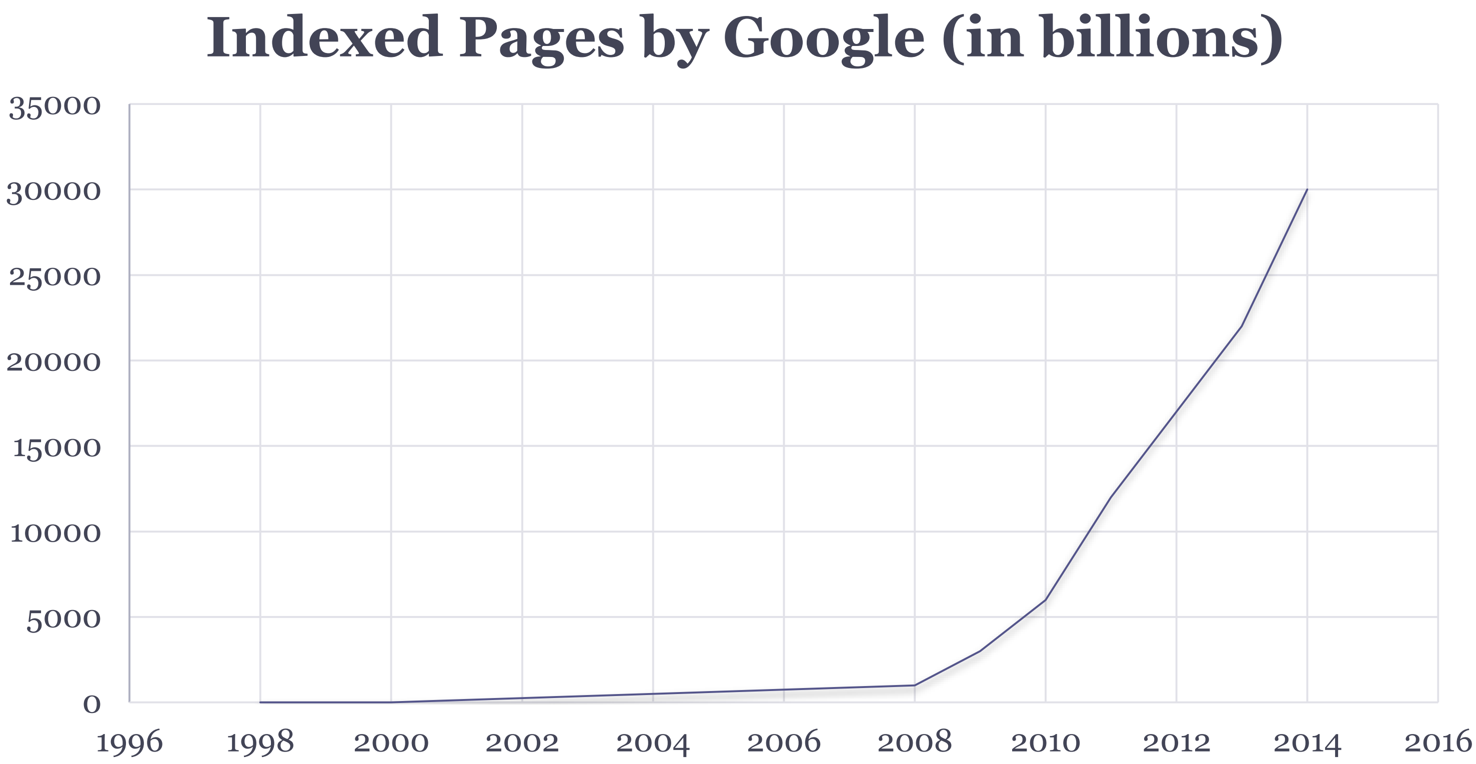
What is ranking?

- Information Retrieval

  → **WEB SEARCH**



Web Search Statistics



(http://www.statisticbrain.com/total-number-of-pages-indexed-by-google/)

Conventional Ranking Methods

- Assign score to each document and sort

- Query-dependent models
  - e.g. TF-IDF
- Query-independent models
  - PageRank

Query-Dependent Models

- Boolean Model
  - Are keywords present in the document?

Query-Dependent Models

- Vector Space Model
  - Relative term frequencies used
  - Cosine similarity

Query: the information retrieval

Term - t	Term frequency - TF(t)
The	0.33
Information	0.33
Retrieval	0.33

Document

Term – t	Term frequency – TF(t)
The	0.18
Be	0.08
…	
Information	0.02
Retrieval	0.005

Similarity: (0.18+0.02+0.005)/3=0.0683

Query-Dependent Models

- TF-IDF
  - Weight each component of product by importance

$$IDF(t) = \log \frac{N}{n(t)}$$

$$TFIDF(d,q) = \sum_{t \in q} TF(t) \cdot IDF(t)$$

Query-Independent Models

- PageRank
  - Probability of reaching a page by random walk

$$PR(d) = \sum_{d'} \frac{PR(d')}{U(d')}$$

Learning to Rank

Learning to Rank

- Commercial attention (web search engines)
- Lots of training data (click-through data)
- Hot topics involved
  - Big data
  - Online learning
  - Deep learning
  - …

Training/Test Sets

- Queries and associated documents
- Features extracted for each query-document pair
  - e.g. using conventional methods
- Target ranking
  - Relevance degree
  - Preference relation
  - Full ranking

Goal

- Get close to the human-assigned ranking (on previously unseen queries)
- Evaluation measures:
  - MAP (Mean Average Precision)

$$AP(q) = \frac{\sum_{k=1}^{m} P@k(q) \cdot l_k}{\#(\text{relevant docs})}$$

  - NDCG (Normalized Discounted Cumulative Gain)
    - Gain for ranking a document at given position
    - Later ranked documents have lower contribution

True Story

- Hard to optimize for “evaluation measures”
  - Non-continuous
  - Non-differentiable

- Reality
  - Another objective is typically used
    (the correspondence to the original measure is limited)

Basic Approaches

- Pointwise
  - Documents treated separately
- Pairwise
  - Pairwise preference relation
- Listwise
  - Whole ranking considered

Pointwise Approach

- Goal: Predict relevance degree of a document

- Input space: Document features (query-based)
- Output space: Relevance degrees
- Loss function: regression/classification error

- Techniques
    - Regression
    - Classification

Pointwise Approach

+ Straightforward
+ Standard ML algorithms directly applicable

- Cannot use information about rank position
- Queries with many results dominate
- Forgets about the ranking goal

Pairwise Approach

- Goal: Learn pairwise preference

- Input: Document pairs (and their query-based features)
- Output: Preferred document from the pair

- Issue: Total order needed
  - *rank aggregation* – NP-hard problem

Pairwise Approach - RankNet

- Algorithm was used in practice (Microsoft)

- Learns scoring function f
- Preference defined as:

$$P_{u,v}(f) = \frac{\exp(\mathrm{f}(x_u) - \mathrm{f}(x_v))}{1 + \exp(\mathrm{f}(x_u) - \mathrm{f}(x_v))}$$

- Neural network used

Pairwise Approach - RankNet

- Loss function: cross entropy

$$L(f, x_u, x_v, y_{u,v})$$
$$= -\bar{P}_{u,v} \log P_{u,v}(f) - (1 - \bar{P}_{u,v}) \log\left(1 - P_{u,v}(f)\right)$$

\+ Easy to optimize (convex)

\- Unbounded (hard cases dominate)

\- Always positive

→ FRank – better results but harder optimization (non-convex)

Pairwise Approach

+ Ordering matters
+ Easy application of ML techniques
  (…, SVM, Boosting, …)

- Position information not used
- Queries with many results still dominate
  → even worse! Quadratic number of pairs

Listwise Approach

- Use the whole ranking

- Input: Set of document features
- Output: Ranking

- Objective:
  - Optimize the evaluation measure directly
  - Optimize consistency with desired ranking

Listwise Approach

- Direct optimization – **hard** problem
  - Non-continuous and non-differentiable objective

  - Options:
    - Genetic algorithms: RankGP
    - Smooth the objective: SoftRank
    - …

Listwise Approach - ListMLE

- Idea: Probabilistic distribution over rankings
  - Induced by ranking scores – *s*

- Luce model of permutation probability

$$P(\pi|s) = \prod_{i=1}^{m} \frac{\varphi(s_{\pi^{-1}(i)})}{\sum_{j=i}^{m} \varphi(s_{\pi^{-1}(j)})}$$

- Loss function ($f: X \rightarrow \mathbb{R}$ is the scoring function):

$$L(f, x, \pi_y) = -\log P(\pi_y|f)$$

Listwise Approach

+ Empirically best performance
+ Evaluation measures taken into account

Cons depend on the exact algorithm:

- Complexity of training process
- Often no positional discounting

Query-Dependent Ranking

Where is the problem?

- One model for all queries
  - Golden mean, but still suboptimal

- Example (Broder's taxonomy)
  - Navigational queries – locate a specific webpage
  - Informational queries – find information on a topic
  - Transactional queries

Options

- Train model based on the most similar queries
  - e.g. k-NN search
  - Questionable efficiency (learning in query phase)
  - Solution:
    - Pretrain finite number of models
    - Use the one with greatest overlap with nearest neighbors

Options

- Two-Layer approach
  - Make the ranking model depend on the query
  - Idea: Infinite number of models trained

$$\min_{\nu} \sum_{i=1}^{n} L\left(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}\right),$$

$$\text{s.t.} \quad \hat{\mathbf{y}}^{(i)} = \text{sort} \circ f\left(w^{(i)}, \mathbf{x}^{(i)}\right),$$

$$w^{(i)} = g\left(\nu, z^{(i)}\right),$$

References

- Liu, Tie-Yan. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

Thank you for your attention