Processamento e Recuperação de Informação

Learning to Rank

Departamento de Engenharia Informática
Instituto Superior Técnico

1º Semestre
2018/2019

Outline

Bibliography

- Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval, 2nd edition. Chapter 11.
- T.-Y. Liu, "Learning to rank for information retrieval," Foundations and Trends in Databases, vol. 3, no. 3, pp. 225-331, 2009.

Outline

Search Engine Ranking

Ranking is the hardest and most important function of a search engine

Main challenges:

- Evaluation
- Managing Web spam
- Identification of relevant content
- Defining the ranking function

Evaluating the Ranking

- Devise an adequate process of evaluating the ranking, in terms of relevance of results to the user
- Without such evaluation, it is close to impossible to fine tune the ranking function
- Without fine tuning the ranking, there is no state-of-the-art engine—this is an empirical field of science

Dealing with Web Spam

- Avoiding, preventing, managing Web spam
- Spammers are malicious users who try to trick search engines by artificially inflating signals used for ranking
- A consequence of the economic incentives of the current advertising model adopted by search engines

Defining Relevant Content

Evidence of quality can be indicated by several signals such as:

- Domain names
- Text content
- Links (e.g. PageRank)
- Web page access patterns

Additional useful signals are provided by the layout of the Web page, its title, metadata, font sizes, etc.

The Ranking Function

Following: from simple ranking functions to complex combinations of signals

Outline

Evidences for Relevance

Three main types of signals:

1. Content
2. Structure
3. Usage

In total we can have hundreds of distinct signals

- Bing claims to use > 1000 (see here)
- Google claims to use > 200 many with > 50 variations

Content Signals

- Related to the text itself
- Can vary from simple word counts to a full IR score, such as TF-IDF or BM25
- Can be provided by the layout, that is, the HTML source
  - Simple format indicators (more weight given to titles/headings)
  - Sophisticated indicators as the proximity of certain tags in the page

Structure Signals

- Intrinsic to the linked structure of the Web
- Some of them are textual in nature, such as anchor text
- Others pertain to the links themselves, such as in-links and out-links from a page
- Link-based signals find broad usage beyond classic search engine ranking

Web Usage Signals

- Main one is the implicit feedback provided by the user clicks (click-through)
- Other usage signals include:
  - information on the user's geographical context (IP address, language)
  - technological context (operating system, browser)
  - temporal context (query history by the use of cookies)
  - even site speed

Outline

Simple Ranking Scheme

- Use only text-based ranking
  - E.g. BM25 or cosine similarity
- Applied in early search engines

Or...

- Use a global ranking function such as PageRank
- Quality of a Web page in the result set is independent of the query
- The query only selects pages to be ranked

Simple Combination of Signals

- Use a linear combination of different ranking signals

Example

- Consider the pages p that satisfy query Q
- Rank score $R(p, Q)$ of page p with regard to query Q can be computed as

$$R(p, Q) = \alpha BM25(p, Q) + (1 - \alpha)PR(p)$$

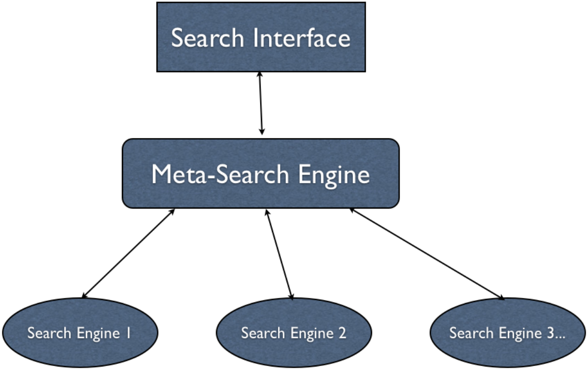
  - $\alpha = 1$: text-based ranking
  - $\alpha = 0$: link-based ranking, independent of the query

A More Complex Combination

- Current engines combine a text-based ranking with a link-based ranking, most of them a lot more complex than BM25 and PageRank
- Value of α is tuned experimentally using
  - Labeled data as ground truth, or
  - Clickthrough data
- α might even be query dependent
  - for *navigational* queries α could be made smaller than for *informational* queries

Outline

Principle


Search Interface
Meta-Search Engine
Search Engine 1
Search Engine 2
Search Engine 3...

Combining Similarity Scores

1. eliminate duplicates
2. apply a fusion algorithm
   - using similarity scores provided by underlying SE

these techniques can be used also to combine ranking functions within a search engine

Combination Using Similarities

- $CombMIN(d_j) = min(s_{1j}, s_{2j}, ..., s_{kj})$
  (use the minimum ranking)
- $CombMAX(d_j) = max(s_{1j}, s_{2j}, ..., s_{kj})$
- $CombSUM(d_j) = \sum s_{ij}$
  (add the similarity scores)
- $CombMNZ(d_j) = CombSUM(d_j) \times r_j$, where r_j is the number of systems that retrieved d_j

CombSUM and *CombMNZ* perform better. *CombMNZ* slighlty outperforms *CombSUM* in most cases.

Combination using ranking positions

Borda(1770) Ranking: each voter assigns a linear preference order of candidates, n to the first, $n-1$ to the second, etc. Unranked candidates divide the votes. Winner gets the most points.

Condorcet (1787) Ranking: do pairwise comparisons to count how many times a doc "wins", "loses" or "ties" against other documents (as in a soccer tournament). Doc with most wins gets highest score. Ties broken on number of losses.

Reciprocal ranking: assign a score $1/pos$ to each doc. Rank based on sum of scores.

Borda Ranking

5 underlying search engines, which have ranked four candidate pages a, b, c, d.

System 1: a,b,c,d
System 2: b,a,d,c
System 3: c,b,a,d
System 4: c,b,d
System 5: c,b

Scores:

$$Score(a) = 4 + 3 + 2 + 1 + 1.5 = 11.5$$
$$Score(b) = 3 + 4 + 3 + 3 + 3 = 16$$
$$Score(c) = 2 + 1 + 4 + 4 + 4 = 15$$
$$Score(d) = 1 + 2 + 1 + 2 + 1.5 = 7.5$$

The final ranking is: b, c, a, d

Condorcet Ranking

System 1:	a,b,c,d
System 2:	b,a,d,c
System 3:	c,b,a,d
System 4:	c,b,d
System 5:	c,b

comparisons (win:lose:tie):

pair	a	b	c	d
a	-	1:4:0	2:3:0	3:1:1
b	4:1:0	-	2:3:0	5:0:0
c	3:2:0	3:2:0	-	4:1:0
d	1:3:1	0:5:0	1:4:0	-

The final ranking is: b, c, a, d

Reciprocal Ranking

**5 underlying search engines,
which have ranked four candidate pages a, b, c, d.**

System 1: a,b,c,d
System 2: b,a,d,c
System 3: c,b,a,d
System 4: c,b,d
System 5: c,b

Scores:

$$Score(a) = 1 + 1/2 + 1/3 + 0 + 0 = 1.83$$
$$Score(b) = 1/2 + 1 + 1/2 + 1/2 + 1/2 = 3$$
$$Score(c) = 1/3 + 1/4 + 1 + 1 + 1 = 3.55$$
$$Score(d) = 1/4 + 1/3 + 1/4 + 1/3 + 0 = 1.17$$

The final ranking is: c, b, a, d

Outline

Why Learning to Rank?

- Manual parameter tuning is usually difficult
  - Especially when there are many parameters and the evaluation measures are non-smooth
- Manual parameter tuning sometimes leads to overfitting
- It is non-trivial to combine the large number of models proposed in the literature (e.g. BM25, etc.) to obtain an even more effective model

What is Learning to Rank?

L2R: apply machine learning techniques to learn the ranking of the results

- Use a learning algorithm fed with training data that contains ranking information
- loss function to minimize: number of mistakes done by the learned model

Outline

Supervised Learning

Input: $\{(x_i, y_i)\}_{i=1}^{N}$, $x_i \in \mathcal{R}^M, y_i \in \mathcal{R}$

Hypothesis space: $h^* \in H$

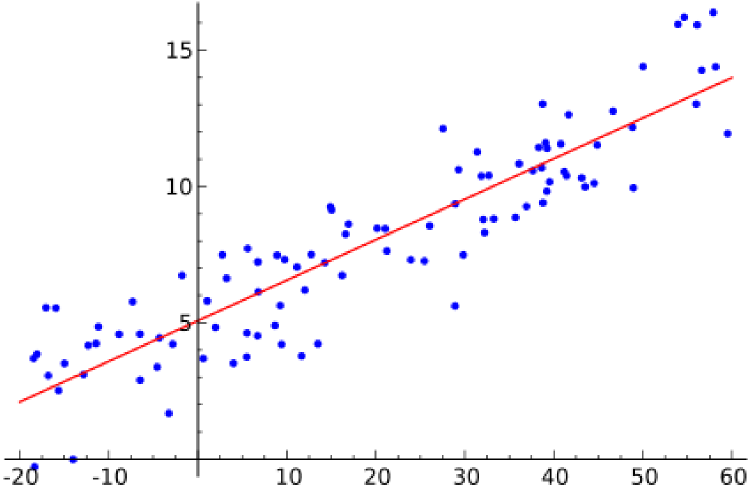
Loss function: $L(h(x), y)$

Learning Algorithm: $\hat{h} = A(\{(x_i, y_i)\}_{i=1}^{N})$, such that

$$\hat{h} = \text{argmin}_h \sum_{i=1}^{N} L(h(x_i), y_i)$$

I.e. given a set of training data as input, use learning algorithm A to discover the function $\hat{h}$ that minimizes the loss (e.g. the error)

An Example: Linear Regression



(source: wikipedia)

Linear Regression (cont.)

- The hypothesis space:

$$h_{\vec{w}}(x) = w_0 + w_1 x$$

where $\vec{w} = [w_0, w_1]$

- The loss function:

$$L(h_{\vec{w}}, y) = \sum_{i=1}^{N} (y_i - h_{\vec{w}}(x_i))^2$$

i.e. the sum of the squared error

- We want to find

$$w^* = \operatorname*{argmin}_{w} L(h_{\vec{w}}, y)$$

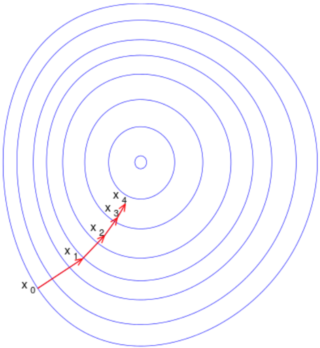
Minimizing the Loss

- In the most simple case, we can easily find one (or more) solution(s)
  - Just take the derivatives and equal to 0
- In many cases this is not possible (or we may want to enforce some constraints on the parameters)
- In practice, there are many ways to estimate w^*

An Example: Gradient Descent

$w \leftarrow$ any point in the parameter space
loop until convergence **do**
    for each w_i **in** $\vec{w}$ **do**
        $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} L(h_{\vec{w}}, y)$

$\alpha =$ learning rate



(source: wikipedia)

Other Types of Supervised Learning

(besides regression and ranking)

- Classification
  - Classify email as spam vs. ham
  - Loss: accuracy
- Structured prediction
  - Find faces in an image
  - Loss: Precision/Recall of faces

Outline

The L2R Framework

Processamento e Recuperação de Informação

Search Engine Ranking

Ranking Signals

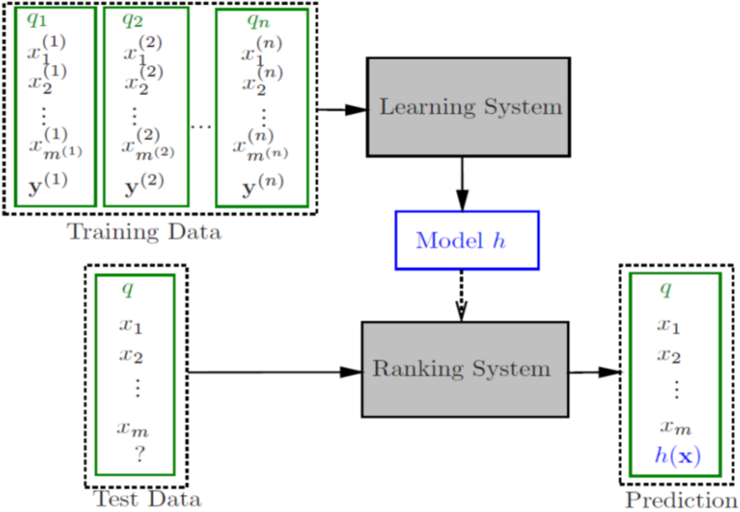
The Ranking Function

Unsupervised Rank Fusion

Learning to Rank

Some Context

Learning to Rank (cont.)



L2R Techniques

Three main approaches:

- **Pointwise:** focuses on individual pages
- **Pairwise:** focuses on comparing pairs of pages
- **Listwise:** focuses on the ranked list of pages

The Pointwise Approach

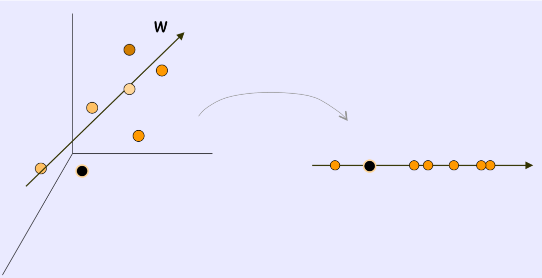
	The Pointwise Approach		
	Regression	Classification	Ordinal Regression
Input Space	Single documents y_j		
Output Space	Real values	Non-ordered Categories	Ordinal categories
Hypothesis Space	Scoring function $f(x)$		
Loss Function	Regression loss	Classification loss	Ordinal regression loss
	$L(f;x_j,y_j)$		

An Example: Ranking Perceptron

Koby Crammer and Yoram Singer, "Pranking with ranking," In Proceedings of the 14th International Conference on Neural Information Processing Systems (NIPS'01), 2001.

Adaptation of the *Perceptron algorithm*:

Input space: $\mathbf{x} = \{x_j\}_{j=1}^{m}$
Output space: $y_j \in \{1, 2, 3, \cdots\}$
Hypothesis Space: $f(x) = \mathbf{w} \cdot x$
Loss function: $L(f, x_j, y_j) = \sum_{j=1}^{T} |y_j - f(x_j)|$



Problem with the Pointwise Approach

- The position of documents in the ranked list is invisible to the loss functions
- The overall loss function will be dominated by queries with a large number of documents

The Pairwise Approach

	The Pairwise Approach
Input Space	Document pairs (x_u, x_v)
Output Space	Preference $y_{u,v} \in \{+1, -1\}$
Hypothesis Space	Preference function $h(x_u, x_v) = 2 \cdot I_{\{f(x_u) > f(x_v)\}} - 1$
Loss Function	Pairwise classification loss $L(h; x_u, x_v, y_{u,v})$

An Example: RankBoost

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," Journal of Machine Learning Research, vol. 4, pp. 933–969, 2003.

Input space: Document pairs (x_u, x_v)

Output space: Relative order $y_{u,v} \in \{-1, +1\}$

Hypothesis Space: $f(x) = \sum_t \alpha_t f_t(x)$

Loss function: $L(f; x_u; x_v; y_{u,v}) = e^{-y_{u,v}(f(x_u) - f(x_v))}$

Algorithm 1 Learning Algorithm for RankBoost

Input: document pairs

Given: initial distribution $\mathcal{D}_1$ on input document pairs.

For $t = 1, \ldots, T$

Train weak ranker f_t based on distribution $\mathcal{D}_t$.

Choose α_t

Update $\mathcal{D}_{t+1}(x_u^{(i)}, x_v^{(i)}) = \frac{1}{Z_t}\mathcal{D}_t(x_u^{(i)}, x_v^{(i)})\exp(\alpha_t(f_t(x_u^{(i)}) - f_t(x_v^{(i)})))$

where $Z_t = \sum_{i=1}^{n} \sum_{u,v:y_{u,v}^{(i)}=1} \mathcal{D}_t(x_u^{(i)}, x_v^{(i)})\exp(\alpha_t(f_t(x_u^{(i)}) - f_t(x_v^{(i)})))$.

Output: $f(x) = \sum_t \alpha_t f_t(x)$.

Improvement of Pairwise Approach

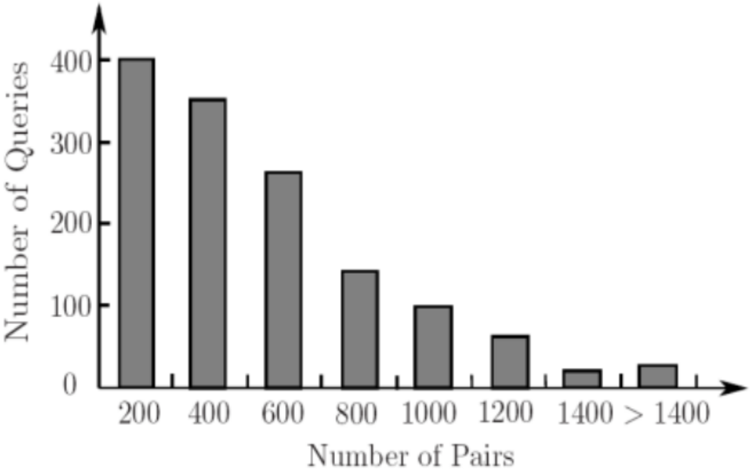
Advantage

Predicting relative order is closer to the nature of ranking than predicting class label or relevance score

Problems

- Relative order of two documents still does not predict their final position
- The distribution of document pair number is more skewed than the distribution of document rank, with respect to different queries

Document Pair Distribution



The Listwise Approach

	The Listwise Approach	
	Listwise Loss Minimization	Direct Optimization of IR Measure
Input Space	Document set $\mathbf{x} = \{x_j\}_{j=1}^{m}$	
Output Space	Permutation π_y	Ordered categories $\mathbf{y} = \{y_j\}_{j=1}^{m}$
Hypothesis Space	$h(\mathbf{x}) = \mathrm{sort} \circ f(\mathbf{x})$	$h(\mathbf{x}) = f(\mathbf{x})$
Loss Function	Listwise loss $L(h; \mathbf{x}, \pi_y)$	1-surrogate measure $L(h; \mathbf{x}, \mathbf{y})$

Direct Optimization of IR Measures

- It is natural to directly optimize what is used to evaluate the ranking results
- However, it is non-trivial
- Evaluation measures such as NDCG are non-continuous and non-differentiable since they depend on the rank positions
- It is challenging to optimize such objective functions, since most optimization techniques in the literature were developed to handle continuous and differentiable cases

Solutions

- Approximate the objective
  - Soften (approximate) the evaluation measure so as to make it smooth and differentiable
- Bound the objective
  - Optimize a smooth and differentiable upper bound of the evaluation measure
- Optimize the non-smooth objective directly
  - Use IR measure to update the distribution in Boosting
  - Use genetic programming

An Example: RankGP

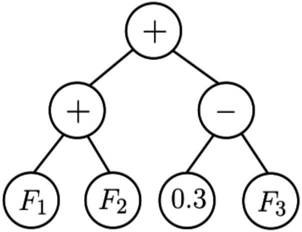
J.-Y. Yeh et al, "Learning to rank for information retrieval using genetic programming," in SIGIR 2007 Workshop in Learning to Rank for Information Retrieval, 2007.

Input space: $\mathbf{x} = \{x_j\}_{j=1}^{m}$

Output space: Relative order $\mathbf{y} = \{y_j\}_{j=1}^{m}$

Hypothesis Space: $f(x)$

Loss function: IR evaluation measure



Improvement of Listwise Approach

Advantages

- Take all the documents associated with the same query as the learning instance
- Rank position is visible to the loss function

Problems

- Complexity

Questions?