Incorporating Query Difference for Learning Retrieval Functions in Web Search

Hongyuan Zha, Zhaohui Zheng, Haoying Fu, Gordon Sun

The Pennsylvania State University/Yahoo! Inc.

MMDS, Stanford, June 21-24, 2006

Outline

- Web search
- A Risk Minimization Framework/Multi-task Learning
- aTVT: Incorporating Query Difference
- Experiments

Introduction

- Retrieval functions: rank documents in response to user queries
- Retrieval models and methods (research done in IR/WWW/NA):
  - vector space model
  - probabilistic model
  - many more ...

[*Algorithms and methods from machine learning*]

Search Engine Ranking Problems

Input: user queries

Output: ranked lists of documents

Basic procedure (Multi-stage ranking)

1) Intial stages: select a list of documents potentially relevant to the query using cheaper features
2) Later stages: use more expensive features to generate ranked list of documents

Examples

1) select documents that contain all the required query words: intersecting inverted word lists (basic IR methods)

2) ranking based on simple linear combinations of features

3) (cluster of machines) parallel execution of the above

Examples (Cont'd)

Extracting other features:

1. Query-document feastures

   title

   url path

   abstract/description (from the metatags)

   keyphrases (comma separated list of phrases)

   body (rest of document)

   anchortext (anchortext pointing to document)

   ...

2. Document features (link, spam, etc.)
3. Query features (length, language, category, etc.)

Designing Ranking Functions

The feature vector $x = [x_1, ..., x_n]$ is extracted for each query-document pair, the goal is to construct a function $h(x)$ such that

$$h(x) > h(x')$$

implies that x is more relevant than x', i.e., list of documents can be sorted according to $\{h(x)\}$.

1) Manual Tuning (function form and/or parameter values)

2) Using machine learning methods: collect training set with labeled data, learn ranking functions either as a problem of

classification/regression/ranking

Generate Training Set

1. Sample queries from query logs
2. Obtain query-url pairs
3. Judges score query-url pairs by assigning grades: perfect, good, ...

$\Rightarrow$ training data in the form of *labeled* feature vectors for query-document pairs $\{(x^i, y^i)\}_{i=1}^N$.

Need to find a function h to match judges' grades, i.e.,

$$h(x^i) \approx y^i, \quad i = 1, \ldots, N.$$

Outline

- Web search
- A Risk Minimization Framework/Multi-task Learning
- $aTVT$: Incorporating Query Difference
- Experiments

A Risk Minimization Framework

$\mathcal{D}$, the set of all the documents
$\mathcal{L}$, the set of labels (perfect, good, ...)
$\mathcal{Q}$, the set of all potential user queries

Model each query $q \in \mathcal{Q}$ as a probabilistic distribution P_q over $\mathcal{D} \times \mathcal{L}$,

$$P_q(d, \ell), \quad d \in \mathcal{D}, \;\; \ell \in \mathcal{L}$$

i.e., the probability of document d being labeled as ℓ w.r.t. query q.

A loss function L over the set $\mathcal{L} \times \mathcal{L}$,

$$L : \mathcal{L} \times \mathcal{L} \mapsto \mathcal{R}^1_+.$$

A class of functions $\mathcal{H}$ to select the retrieval function,

$$h : \mathcal{D} \mapsto \mathcal{L}.$$

For a specific query q, the learning problem (classification or regression problem): find $h^*_q \in \mathcal{H}$

$$h^*_q = \arg\min_{h \in \mathcal{H}} \mathcal{E}_{P_q(d,l)} L(\ell, h(d)).$$

Minimizing expected loss!

Many Queries

1) Web search is not about learning h_q^* for some particular q

2) Learn a retrieval function h^* that will be good for all $q \in \mathcal{Q}$

3) Conceptually, need to deal with potentially *infinite* number of *related* learning problems, each for one of the query $q \in \mathcal{Q}$.

A multi-task learning problem

Specify a distribution over $\mathcal{Q}$: $P_Q(q)$ indicate the probability that a specific query q is issued, approximated by frequency in the query logs.

Risk Minimization

$$h^* = \arg\min_{h \in \mathcal{H}} \mathcal{E}_{P_Q} \mathcal{E}_{P_q} L(\ell, h(q, d))$$

Empirically ...

1) Sample a set of queries $\{q_i\}_{i=1}^{Q}$ from the distribution P_Q, for each q, and sample a set of documents from $\mathcal{D}$ for labeling $\Rightarrow$ training data,

$$\{d_{qj}, l_{qj}\}, \quad q = 1, \ldots, Q,\ j = 1, \ldots, n_q$$

2) Empirical risk minimization with regularization,

$$h^* = \arg\min_{h\in\mathcal{H}} \sum_{q=1}^{Q} \sum_{j=1}^{n_q} L(\ell_{qj}, h(q, d_{qj})) + \lambda \underbrace{\Omega(h)}_{\text{reg. term}}$$

An Oversimplified Example

q_1: "harvard university", 13 million search results on Yahoo
q_2: "college of san mateo", two orders of magnitude less results

A retrieval function $h(x)$ using $x = \#$ inbound links to a document.

$$\begin{array}{llll} q_1: & x = 100000\ (P), & 80000(G), & 50000(B) \\ q_2: & x = 1000\ (P), & 800\ (P), & 500\ (B) \end{array}$$

$$0 \Leftrightarrow \text{perfect},\ 1 \Leftrightarrow \text{good},\ 2 \Leftrightarrow \text{bad}.$$

Need to find a monotonically decreasing function h such that for q_1

$$h(100000) \approx 0,\ h(80000) \approx 1,\ h(50000) \approx 2$$

and for q_2,

$$h(1000) \approx 0,\ h(800) \approx 1,\ h(500) \approx 2.$$

Query Classes

Training set, $[d_{qj}, q] \Leftrightarrow x_{qj}$, $l_{qj} \Leftrightarrow y_{qj}$,

$$\{x_{qj}, y_{qj}\}, \quad q = 1, \ldots, Q, \; j = 1, \ldots, n_q,$$

x_{qj} denotes the feature vector for the query-document pair $\{q, d_{qj}\}$.
Split x_{qj} into three parts,

$$x_{qj} = [x_{qj}^{Q}, \; x_{qj}^{D}, \; x_{qj}^{QD}]$$

Two extremes:

1) Only use $[x_{qj}^{D}, \; x_{qj}^{QD}]$, ignoring query difference

2) Have one $h_q([x_{qj}^{D}, \; x_{qj}^{QD}])$ for each query $q \in \mathcal{Q}$.

Better:

A single function $h_q([x_{qj}^{D}, \; x_{qj}^{D}, \; x_{qj}^{QD}])$.

But it is hard to figure out the right (granularity of) x_{qj}^{D}.

Nuisance Parameters/Latent Variables

Basic idea: let the data implicitly capture this set of adequate query-features and bypass its explicit construction.

[Introduction of nuisance parameters (functions)/latent variables]

Incorporating Query Difference

We use regression with squared-error loss function,

$$L(h) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} (y_{qj} - h(x_{qj}))^2.$$

To incorporate query-dependent effects, we seek to find h and $g_q, q = 1, \ldots, Q$, to minimize

$$L(h, g) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - g_q(h(x_{qj}))]^2, \quad (1)$$

where $g_q(\cdot)$ is a general monotonically increasing function, and $g = [g_1, \ldots, g_Q]$.

- g_q captures the difference of queries
- related to response transformation in regression
- for a new query $q^* \Rightarrow g_{q^*}$, but rankings based on $g_{q^*}(h)$ and h are exactly the same

For simplicity, we focus on the linear case,

$$g_q(x) = \beta_q + \alpha_q x, \quad q = 1, \ldots, Q$$

with $\alpha_q \geq 0$.

Optimization on the regularized empirical risk

$$L(h, \beta, \alpha) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 +$$

$$+\lambda_\beta \|\beta\|_p^p + \lambda_\alpha \|\alpha\|_p^p + \lambda_h \Omega(h),$$

where $\beta = [\beta_1, \ldots, \beta_Q]$ and $\alpha = [\alpha_1, \ldots, \alpha_Q]$, and $\lambda_\beta, \lambda_\alpha$ and λ_h are regularization parameters.

Coordinate Descent

Basic idea: alternate between optimizing against h and optimizing against β and α.

Nonlinear regression

For fixed β and α, define the modified residuals

$$\hat{y}_{qj} = (y_{qj} - \beta_q)/\alpha_q, \quad q = 1, \ldots, Q,\ j = 1, \ldots, n_q.$$

Then find h to solve the following *weighted* nonlinear regression problem

$$\sum_{q=1}^{Q} \sum_{j=1}^{n_q} \alpha_q^2 [\hat{y}_{qj} - h(x_{qj})]^2 + \lambda_h \Omega(h).$$

We use gradient boosting to estimate h.

Optimize against β and α

For fixed h,

$$\min_{\beta,\alpha} \sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 + \lambda_\beta \|\beta\|_p^p + \lambda_\alpha \|\alpha\|_p^p.$$

Decouple into Q separate optimization problems, for $q = 1, \ldots, Q$,

$$\min_{\beta_q,\alpha_q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 + \lambda_\beta |\beta_q|^p + \lambda_\alpha |\alpha_q|^p.$$

Convergence Analysis

$\mathcal{H}$ a reproducing kernel Hilbert space (RKHS) with kernel function K, and $\Omega(h) = \|h\|_K^2$.

Theorem 1. *The optimal function h^* for the optimization has the following form,*

$$h^*(x) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} c_{qj} K(x_{qj}, x),$$

where $c_{qj}, q = 1, \ldots, Q, j = 1, \ldots, n_q$, are real numbers.

Theorem 2. *Every limit point of $\{c^k, \beta^k, \alpha^k\}_{k=1}^{\infty}$ is a stationary point of $L(c, \beta, \alpha)$.*

Data Collection

- randomly sample a certain number of queries from the query logs.
- label documents
- we finally represent each query-url pair with a feature vector.

$\#$ of queries $\sim O(10^3)$ and $\#$ of query-url pairs $\sim O(10^5)$

Feature Engineering

For each query-document pair (q, d) with $q \in \mathcal{Q}$ and $d \in \mathcal{D}$, a feature vector $x = [x^Q, x^D, x^{QD}]$ is generated,

- Query-feature vector x^Q, e.g., the number of terms in the query, whether or not the query is a person name, etc.
- Document-feature vector x^D, e.g., the number of inbound links pointing to the document, the amount of anchor-texts in bytes for the document, and the language identity of the document, etc.
- Query-document feature vector x^{QD}, e.g., the number of times each term in the query q appears in the document d, the number of times each term in the query q appears in the anchor-texts of the document d, etc.

Experiment methodology

$$\min_{\{\alpha_q,\beta_q\}} \sum_{j=1}^{n_q} (\alpha_q y_{qj} + \beta_q - h(x_{qj}))^2 + \lambda_\alpha |(\alpha_q - 1)|_p^p + \lambda_\beta |\beta_q|_p^p.$$

Algorithm. *Adaptive Target Value Tranformation* (aTVT).

For each choice of regularization parameters λ_α and λ_β,

1) initialize y_{qj}^0 to the assigned numerical values for each query-document pair (q, d);

2) iterate until the α_q^k and β_q^k do not change much, do the following,

   a) fit a nonlinear function on $\{y_{qj}^{k-1}\}$ using gradient boosting.

   b) obtain optimal values for the α_q^k and β_q^k.

   c) $y_{qj}^k \leftarrow \alpha_q^k y_{qj}^{k-1} + \beta_q^k$.

Then $\alpha_q = \prod_{k=1}^K \alpha_q^k$, and $\beta_q = \sum_{k=1}^K (\beta_q^k \prod_{i=k+1}^K \alpha_q^i)$.

Performance Measures

(K. Järvelin and J. Kekäläinen, 2002)

1 . Precision-recall used in IR

2 . DCG (Discounted Cumulative Gain): Gain values G.

List of K ranked documents with gain vector G,

$$\mathrm{DCG} = \sum_{i=1}^{K} \frac{G(i)}{\log_2(1+i)}.$$

Cross-validation for Comparison

- Randomly split the set of queries in data set into 10 folds and obtain 10 9-vs-1 combinations.
- For each 9-vs-1 combination, do the following:
  - learn a retrieval function using the data in the 9 folds as training data with and without aTVT, respectively.
  - test the learned retrieval function on the remaining one fold by computing the DCG values for the queries in the one fold.
- concatenate the above lists from the 10 combinations together to obtain a full list of query-dcg pairs for all the queries in the data set.
- Finally, we conduct Wilcoxon signed rank tests on the two lists and obtain the p-values.

Table 1: The dcgs and percentage dcg increases of retrieval function with aTVT over without aTVT and p values.

	λ_β=1	10
λ_α=1	(+1.6%,p=0.01)	(+1.7%,p=0.006)
10	**(+2.00%,p=0.002)**	(+1.9%,p=0.002)
50	(+1.8%,p=0.002)	(+1.8%,p=0.008)
100	(+1.8%,p=0.007)	(+1.7%,p=0.003)

Table 2: dcg gains and corresponding p-values for queries sorted according to $|\alpha^{0.1} - 1.0| + |\beta/10|$

# top queries	dcg gain of aTVT	p-value
200	5.3%	0.002
300	4.2%	0.002
400	4.0%	0.0004
500	3.4%	0.0006
600	2.8%	0.001
700	2.6%	0.0006
800	2.3%	0.0005
all	2.0%	0.002

Recap

- Web search $\Rightarrow$ multi-task learning$\Rightarrow$ aTVT
- Internet/Web: sources of interesting and challenging problems