Document indexing, similarities and retrieval in large scale text collections

Eric Gaussier

Univ. Grenoble Alpes - LIG
Eric.Gaussier@imag.fr

Course objectives

- Introduce the main concepts, models and algorithms for computing indexing text documents and computing similarities in large scale text collections
- We will focus on:
  - Document indexing and representations in large scale collections
  - Standard models for Information Retrieval (IR)
  - PageRank (computing the importance of a page on the Web)
  - Learning to rank models

Application domains

- Information retrieval
  - Query indexing module
  - Documents indexing module
  - Module to match queries and documents
- Classification
  - Binary, multi-class; mono-/multi-label
  - Flat vs hierarchical
- Clustering
  - Hard vs soft clustering
  - Flat vs hierarchical

Part 1: Indexing, similarities, information retrieval

Content

Indexing steps

1. Segmentation
   - Segment a text into words:

     the importance of retrieving the good information
     the, importance, of, retrieving, the, good, information

     7 words but only 6 word types; depending on languages, may require a dictionary
2. Stop-word removal (stop-word list)
3. Normalization
   - Upper/lower-case, inflected forms, lexical families
   - Lemmatization, stemming

$\rightarrow$ Bag-of-words: *importance, retriev, inform*

Vector space representation

- The set of all word types constitute the vocabulary of a collection. Let M be the size of the vocabulary and N be the number of documents in the collection $\rightarrow$ M-dimensional vector space (each axis corresponds to a word type)
- Each document is represented by a vector the coordinates of which correspond to:
    - Presence/absence or number of occurrences of the word type in the doc: $w_i^d = \mathrm{tf}_i^d$
    - Normalized number of occurrences: $w_i^d = \frac{\mathrm{tf}_i^d}{\sum_{i=1}^M \mathrm{tf}_i^d}$
    - *tf*idf*:
      $$w_i^d = \frac{\mathrm{tf}_i^d}{\sum_{i=1}^M \mathrm{tf}_i^d} \underbrace{\log \frac{N}{\mathrm{df}_i}}_{\mathrm{idf}_i}$$
      where df_i is the number of docs in which word (type) i occurs

A sparse representation

Most of the words (terms) only occur in few documents and most coordinates of each document are null; storage space is thus saved by considering only words present in documents $\rightarrow$ sparse representation

Example

$$\text{document } d \left\{ \begin{array}{ll} \text{int l} & \text{(doc length)} \\ \text{ArrWords int[l]} & \text{(sorted word indices)} \\ \text{ArrWeights float[l]} & \text{(word weights)} \\ \ldots & \end{array} \right.$$

How to compute a dot product between documents?

Dot product with sparse representations

Inverted file

It is possible, with sparse representations, to speed up the comparison between docs by relying on an *inverted file* that provides, for each term, the list of documents they appear in:

$$\text{word } i \begin{cases} \text{int l} & \text{(number of docs)} \\ \text{ArrDocs int[l]} & \text{(sorted doc indices)} \\ \ldots & \end{cases}$$

Remark Advantageous with measures (distances, similarities) that do not rely on words not present in docs; dot/scalar product?, cosine?, Euclidean distance?

Building an inverted file

With a static collection, 3 main steps:

1. Extraction of id pairs *(term, doc)* (complete pass over the collection)
2. Sorting acc. to term id, then doc id
3. Grouping pairs corresponding to same term

Easy to implement when everything fits into memory

How to proceed with large collections?

Insufficient memory

Intermediate "inverted files" are temporarily stored on disk. As before, 3 main steps:

1. Extraction of id pairs *(term, doc)* (previous algo.) and writing on file F
2. Reading file F by blocks that can fit into memory; inversion of each block (previous algo.) and writing in a series of files
3. Merging all local files to create global inverted file

$\rightarrow$ *Blocked sort-based indexing* (BSBI) algorithm

BSBI (1)

```
1 $n \leftarrow 0$
2 while (some docs have not been processed)
3 do
4     $n \leftarrow n + 1$
5     block $\leftarrow$ ParseBlock()
6     BSBI-Invert(block)
7     WriteBlockToDisk(block,$f_n$)
8 MergeBlocks($f_1$, ..., $f_n$;$f_{\text{merged}}$)
```

BSBI (2)

The inversion (in BSBI) consists in sorting pairs on two different keys (term and doc ids). Complexity in $O(T \log T)$ where T represents the number of (term,doc) pairs

Example

$t_1 =$ "brutus", $t_2 =$ "caesar", $t_3 =$ "julius", $t_4 =$ "kill", $t_5 =$ "noble"

$t_1 : d_1$	$t_2 : d_4$	$t_2 : d_1$
$t_3 : d_{10}$	$t_1 : d_3$	$t_4 : d_8$
$t_5 : d_5$	$t_2 : d_2$	$t_1 : d_7$

Standard IR models

The different standard models

- Boolean model
- Vector-space model
- Prob. models

Notations

x_w^q	Nbr of occ. of w in query q
x_w^d	Nbr of occ. of w in doc d
t_w^d	Normalized version of x_w^d (weight)
N	Nbr of docs in collection
M	Nbr of words in collection
F_w	Total nbr of occ. of w: $F_w = \sum_d x_w^d$
N_w	Nbr of docs in which w occurs: $N_w = \sum_d I(x_w^d > 0)$
y_d	Length of doc d
m	Longueur moyenne dans la collection
L	Longueur de la collection
RSV	Retrieval Status Value (score)

Boolean model (1)

Simple model based on set theory and Boole algebra, characterized by:

- Binary weights (presence/absence)
- Queries as boolean expressions
- Binary relevance
- System relevance: satisfaction of the boolean query

Boolean model (2)

Example

$q =$ programming $\wedge$ language $\wedge$ (C $\vee$ java)

($q =$ [prog. $\wedge$ lang. $\wedge$ C] $\vee$ [prog. $\wedge$ lang. $\wedge$ java])

	programming	language	C	java	$\cdots$
d_1	3 (1)	2 (1)	4 (1)	0 (0)	$\cdots$
d_2	5 (1)	1 (1)	0 (0)	0 (0)	$\cdots$
d_0	0 (0)	0 (0)	0 (0)	3 (1)	$\cdots$

Relevance score

$RSV(d_j, q) = 1$ iff $\exists\, q_{cc} \in q_{dnf}$ s.t. $\forall w, t_w^d = t_w^q$; 0 otherwise

Boolean model (3)

Algorithmic considerations

Sparse term-document matrix: inverted file to select all document in conjonctive blocks (can be processed in parallel) - intersection of document lists

	d_1	d_2	d_3	$\cdots$
programming	1	1	0	$\cdots$
langage	1	1	0	$\cdots$
C	1	0	0	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	

Boolean model (4)

Advantages and disadvantages

$+$ Easy to implement (at the basis of all models with a union operator)

- Binary relevance not adapted to topical overlaps
- From an information need to a boolean query

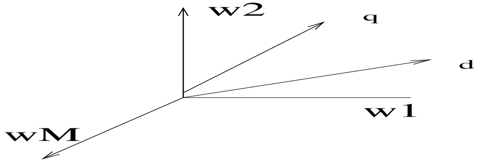
Remark At the basis of many commercial systems

Vector space model (1)

Corrects two drawbacks of the boolean model: binary weights and relevance

It is characterized by:

- Positive weights for each term (in docs and queries)
- A representation of documents and queries as vectors (see before on bag-of-words)



Vector space model (2)

Docs and queries are vectors in an M-dimensional space the axes of which corresponds to word types

Similarity Cosine between two vectors

$$RSV(d_j, q) = \frac{\sum_w t_w^d t_w^q}{\sqrt{\sum_w (t_w^d)^2}\sqrt{\sum_w (t_w^q)^2}}$$

Proprerty The cosine is maximal when the document and the query contain the same words, in the same proportion! It is minimal when they have no term in common (similarity score)

Vector space model (3)

Advantages and disadvantages

+ Total order (on the document set): distinction between documents that completely or partially answer the information need

- Framework relatively simple; not amenable to different extensions

Complexity Similar to the boolean model (dot product only computed on documents that contain at least one query term)

Probabilistic models

- *Binary Independence Model* and BM25 (S. Robertson & K. Sparck Jones)
- *Inference Network Model* (Inquery) - *Belief Network Model* (Turtle & Croft)
- *(Statistical) Language Models*
  - *Query likelihood* (Ponte & Croft)
  - *Probabilistic distance retrieval model* (Zhai & Lafferty)
- *Divergence from Randomness* (Amati & Van Rijsbergen) - *Information-based models* (Clinchant & Gaussier)

Generalities

Boolean model	$\rightarrow$	binary relevance
Vector space model	$\rightarrow$	similarity score
Probabilistic model	$\rightarrow$	probability of relevance

Two points of view: document generation (probability that the document is relevant to the query - BIR, BM25), query generation (probability that the document "generated" the query - LM)

Introduction to language models: two die

Let D_1 and D_2 two (standard) die such that, for small ϵ:

For D_1, $P(1) = P(3) = P(5) = \frac{1}{3} - \epsilon$, $P(2) = P(4) = P(6) = \epsilon$
For D_2, $P(1) = P(3) = P(5) = \epsilon$; $P(2) = P(4) = P(6) = \frac{1}{3} - \epsilon$

Imagine you observe the sequence $Q = (1, 3, 3, 2)$. Which dice most likely produced this sequence?

Answer

$P(Q|D_1) = (\frac{1}{3} - \epsilon)^3\epsilon$; $P(Q|D_2) = (\frac{1}{3} - \epsilon)\epsilon^3$

Introduction to language models: two die

Let D_1 and D_2 two (standard) die such that, for small ϵ:

For D_1, $P(1) = P(3) = P(5) = \frac{1}{3} - \epsilon$, $P(2) = P(4) = P(6) = \epsilon$
For D_2, $P(1) = P(3) = P(5) = \epsilon$; $P(2) = P(4) = P(6) = \frac{1}{3} - \epsilon$

Imagine you observe the sequence $Q = (1, 3, 3, 2)$. Which dice most likely produced this sequence?

Answer

$$P(Q|D_1) = (\tfrac{1}{3} - \epsilon)^3\epsilon \,;\; P(Q|D_2) = (\tfrac{1}{3} - \epsilon)\epsilon^3$$

Language model - QL (1)

Documents are die; a query is a sequence $\rightarrow$ What is the probability that a document (dice) generated the query (sequence)?

$$(RSV(q,d) =)P(q|d) = \prod_{w \in q} P(w|d)^{x_w^q}$$

How to estimate the quantities $P(w|d)$?

$\rightarrow$ Maximum Likelihood principle *Rightarrow* $p(w|d) = \frac{x_w^d}{\sum_w x_w^d}$

Problem with query words not present in docs

Language model - QL (2)

Solution: smoothing

One takes into account the collection model:

$p(w|d) = (1 - \alpha_d) \frac{x_w^d}{\sum_w x_w^d} + \alpha_d \frac{F_w}{\sum_w F_w}$

Example with Jelinek-Mercer smoothing: $\alpha_d = \lambda$

- $\mathcal{D}$: development set (collection, some queries and associated relevance judgements)
- $\lambda = 0$:
- Repeat till $\lambda = 1$
  - IR on $\mathcal{D}$ and evaluation (store evaluation score and associated λ)
  - $\lambda \leftarrow \lambda + \epsilon$
- Select best λ

Language model - QL (3)

Advantages and disadvantages

+ Theoretical framework: simple, well-founded, easy to implement and leading to very good results

+ Easy to extend to other settings as cross-language IR

- Training data to estimate smoothing parameters

- Conceptual deficiency for (pseudo-)relevance feedback

Complexity similar to vector space model

Evaluation of IR systems

Relevance judgements

- Binary judgements: the doc is relevant (1) or not relevant (0) to the query
- Multi-valued judgements:
  Perfect $>$ *Excellent* $>$ *Good* $>$ *Correct* $>$ *Bad*
- Preference pairs: doc d_A more relevant than doc d_B to the query

Several (large) collections with many ($>$ 30) queries and associated (binary) relevance judgements: TREC collections (trec.nist.gov), CLEF (www.clef-campaign.org), FIRE (fire.irsi.res.in)

Common evaluation measures

- MAP (Mean Average Precision)
- MRR (Mean Reciprocal Rank)
  - For a given query q, let r_q be the rank of the first relevant document retrieved
  - MRR: mean of r_q over all queries
- WTA (Winner Takes All)
  - If the first retrieved doc is relevant, $s_q = 1$; $s_q = 0$ otherwise
  - WTA: mean of s_q over all queries
- NDCG (Normalized Discounted Cumulative Gain)

NDCG

- NDCG at position k:

$$N(k) = \overbrace{Z_k}^{\text{normalization}} \underbrace{\sum_{j=1}^{k}}_{\text{cumul}} \overbrace{(2^{p(j)} - 1)}^{\text{gain}} \; / \; \underbrace{\log_2(j+1)}_{\text{position discount}}$$

- Averaged over all queries

G : Gain

Relevance	Value (gain)
Perfect (5)	$31 = 2^5 - 1$
Excellent (4)	$15 = 2^4 - 1$
Good (3)	$7 = 2^3 - 1$
Correct (2)	$3 = 2^2 - 1$
Bad (0)	$0 = 2^1 - 1$

DCG : Discounted CG

Discounting factor: $\frac{\ln(2)}{\ln(j+1)}$

Doc. (rg)	Rel..	Gain	CG	DCG
1	*Perf. (5)*	31	31	31
2	*Corr. (2)*	3	$34 = 31 + 3$	$32,9 = 31 + 3 \times 0,63$
3	*Exc. (4)*	15	49	$40,4$
4	*Exc. (4)*	15	64	$46,9$
...	...	...	...	...

Ideal ranking: max DCG

Document (rank)	Relevance	Gain	max DCG
1	*Perfect (5)*	31	31
3	*Excellent (4)*	15	$40,5$
4	*Excellent (4)*	15	48
...	...	...	...

Normalized DCG

Doc. (rang)	Rel.	Gain	DCG	max DCG	NDCG
1	*Perfect (5)*	31	31	31	1
2	*Correct (2)*	3	$32,9$	$40,5$	$0,81$
3	*Excellent (4)*	15	$40,4$	48	0.84
4	*Excellent (4)*	15	$46,9$	$54,5$	0.86
...	...	...	...	...	

Remarks on evaluation measures

- Measures for a given position (e.g. list of 10 retrieved documents)
- NDCG is more general than MAP (multi-valued relevance vs binary relevance)
- Non continuous (and thus non derivable)

Part 2: IR on the web - PageRank and Learning to Rank

Content

1. PageRank
2. IR and ML: Learning to Rank (L2R)
3. Which training data?

What is the particularity of the web?

$\rightarrow$ A collection with hyperlinks, the graph of the web, and anchor texts

1. Possibility to augment the standard index of a page with anchor texts
2. Possibility to use the importance of a page in the retrieval score (PageRank)
3. Possibility to augment the representation of a page with new features

What is the particularity of the web?

$\rightarrow$ A collection with hyperlinks, the graph of the web, and anchor texts

1. Possibility to augment the standard index of a page with anchor texts
2. Possibility to use the importance of a page in the retrieval score (PageRank)
3. Possibility to augment the representation of a page with new features

PageRank

What is the importance of a page?

1. Number of incoming links
2. Ratio of incoming/outgoing links
3. A page is important if it is often linked by important pages

What is the importance of a page?

1. Number of incoming links
2. Ratio of incoming/outgoing links
3. A page is important if it is often linked by important pages

A simple random walk

Imagine a walker that starts on a page and randomly steps to a page pointed to by the current page. In an infinite *random walk*, he/she will have visited pages according to their "importance" (*the more important the page is, the more likely the walker visits it*)

Problems

1. Dead ends, black holes
2. Cycles

Solution: teleportation

- At each step, the walker can either randomly choose an outgoing page, with prob. λ, or teleport to any page of the graph, with prob. $(1-\lambda)$
- It's as if all web pages were connected (completely connected graph)
- The random walk thus defines a Markov chain with probability matrix:

$$P_{ij} = \begin{cases} \lambda \frac{A_{ij}}{\sum_{j=1}^{N} A_{ij}} + (1-\lambda)\frac{1}{N} & \text{si } \sum_{j=1}^{N} A_{ij} \neq 0 \\ \frac{1}{N} & \text{sinon} \end{cases}$$

where $A_{ij} = 1$ if there is a link from i to j and 0 otherwise

Definitions and notations

Definition 1 A sequence of random variables $X_0, ..., X_n$ is said to be a *(finite state) Markov chain* for some state space S if for any $x_{n+1}, x_n, ..., x_0 \in S$:

$$P(X_{n+1} = x_{n+1} | X_0 = x_0, ..., X_n = x_n) = P(X_{n+1} = x_{n+1} | X_n = x_n)$$

X_0 is called the initial state and its distribution the initial distribution

Definition 2 A Markov chain is called homogeneous or stationary if $P(X_{n+1} = y | X_n = x)$ is independent of n for any x, y

Definition 3 Let $\{X_n\}$ be a stationary Markov chain. The probabilities $P_{ij} = P(X_{n+1} = j | X_n = i)$ are called the *one-step transition probabilities*. The associated matrix P is called the *transition probability matrix*

Definitions and notations (cont'd)

Definition 4 Let $\{X_n\}$ be a stationary Markov chain. The probabilities $P_{ij}^{(n)} = P(X_{n+m} = j | X_m = i)$ are called the *n-step transition probabilities*. The associated matrix $P^{(n)}$ is called the *transition probability matrix*

Remark: P is a stochastic matrix

Theorem (Chapman-Kolgomorov equation) Let $\{X_n\}$ be a stationary Markov chain and $n, m \geq 1$. Then:

$$P_{ij}^{m+n} = P(X_{m+n} = j | X_0 = i) = \sum_{k \in S} P_{ik}^m P_{kj}^n$$

Regularity (ergodicity)

Definition 5 Let $\{X_n\}$ be a stationary Markov chain with transition probability matrix P. It is called *regular* if there exists $n_0 > 0$ such that $p_{ij}^{(n_0)} > 0 \; \forall i, j \; \in S$

Theorem (fundamental theorem for finite Markov chains) Let $\{X_n\}$ be a regular, stationary Markov chain on a state space S of t elements. Then, there exists π_j, $j = 1, 2, ..., t$ such that:

(a) For any initial state i,
$P(X_n = j | X_0 = i) \rightarrow \pi_j$, $j = 1, 2, ..., t$

(b) The row vector $\pi = (\pi_1, \pi_2, ..., \pi_t)$ is the unique solution of the equations $\pi P = \pi, \; \pi \mathbf{1} = 1$

(c) Any row of P^r converges towards π when $r \rightarrow \infty$

Remark: π is called the long-run or stationary distribution

Summary (1)

1. Stationary, regular Markov chains admit a stationary (steady-stable) distribution
2. This distribution can be obtained in different ways:
   - Power method: let the chain run for a sufficiently long time! $\pi = \lim_{k \to \infty} P^k$
   - Linear system: solve the linear system associated with $\pi P = \pi, \; \pi \mathbf{1} = 1$ (*e.g.* Gauss-Seidel)
   - π is the left eigenvector associated with the highest eigenvalue (1) of P (eigenvector decomposition, *e.g.* Cholevsky)

The PageRank can be obtained by any of these methods

Summary (2)

Two main innovations at the basis of Web search engines at the end of the 90's:

1. Rely on additional index terms contained in anchor texts
2. Integrate the importance of a web page (PageRank) into the score of a page

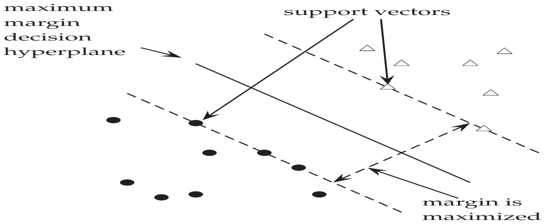
IR and ML: Learning to Rank

Introduction to ML and SVMs (1)

One looks for a decision function that takes the form:

$$f(x) = \text{sgn}(< w, x > +b) = \text{sgn}(w^T x + b) = \text{sgn}(b + \sum_{j=1}^{p} w_j x_j)$$

The equation $< w, x > +b = 0$ defines an hyperplane with *margin* $2/||w||)$



Introduction to ML and SVMs (2)

Finding the *separating* hyperplane with maximal margin amounts to solve the following problem, from a training set $\{(x^{(1)}, y^{(1)}), \cdots (x^{(n)}, y^{(n)})\}$:

$$\begin{cases} \text{Minimize} & \frac{1}{2} w^T w \\ \text{subject to} & y^{(i)}(< w, x^{(i)} > + b) \geq 1, \; i = 1, \cdots, n \end{cases}$$

Non separable case:

$$\begin{cases} \text{Minimize} & \frac{1}{2} w^T w + C \sum_i \xi_i \\ \text{subject to} & \xi_i \geq 0, \; y^{(i)}(< w, x^{(i)} > + b) \geq 1 - \xi_i, \; i = 1, \cdots, n \end{cases}$$

Introduction to ML and SVMs (2)

The decision functions can take two equivalent forms. The "primal" form:

$$f(x) = \text{sgn}(< w, x > +b) = \text{sgn}(< w^*, x^{aug} >)$$

and the "dual" form:

$$f(x) = \text{sgn}(\sum_{i=1}^{n} \alpha_i y^{(i)} < x^{(i)}, x > + b)$$

Modeling IR as a binary classification problem

What is an example? A doc? A query?

$\rightarrow$ A (query,doc) pair: $x = (q, d) \in \mathbb{R}^p$

General coordinates (features) $f_i(q, d), i = 1, \cdots, p$, as:

- $f_1(q,d) = \sum_{t \in q \bigcap d} \log(t^d)$, $f_2(q,d) = \sum_{t \in q} \log(1 + \frac{t^d}{|\mathcal{C}|})$
- $f_3(q,d) = \sum_{t \in q \bigcap d} \log(\text{idf}(t))$, $f_4(q,d) = \sum_{t \in q \bigcap d} \log(\frac{|\mathcal{C}|}{t^{\mathcal{C}}})$
- $f_5(q,d) = \sum_{t \in q} \log(1 + \frac{t^d}{|\mathcal{C}|}\text{idf}(t))$, $f_6(q,d) = \sum_{t \in q} \log(1 + \frac{t^d}{|\mathcal{C}|}\frac{|\mathcal{C}|}{t^{\mathcal{C}}})$
- $f_7(q,d) = \text{RSV}_{\text{vect}}(q,d)$
- $f_8(q,d) = \text{PageRank}(d)$
- $f_8(q,d) = \text{RSV}_{LM}(q,d)$
- ...

Application

Each pair $x(= (q, d))$ containing a relevant (resp. non relevant) doc for the query in the pair is associated to the positive class $+1$ (resp. to the negative class -1)

Remarks

1. One uses the value of the decision function (not its sign) to obtain an order on documents
2. Method that assigns a score for a (query,doc) pair independently of other documents $\rightarrow$ *pointwise method*
3. Main advantage over previous models: possibility to easily integrate new (useful) features
4. Main disadvantage: need for many more annotations
5. Another drawback: objective function different from evaluation function (true objective)

Preference pairs and ranking

1. Relevance is not an absolute notion and it is easier to compare relative relevance of say two documents
2. One is looking for a function f that preserves partial order bet. docs (for a given query): $x_{(i)} \prec x_{(j)} \Longleftrightarrow f(x_{(i)}) < f(x_{(j)})$, with $x_{(i)}$ being again a (query,doc) pair: $x_i = (d_i, q)$

Can we apply the same approach as before? Idea: transform a ranking information into a classification information by forming the difference between pairs

From two documents (d_i, d_j), form:

$$x^{(i,j)} = (x_i - x_j, z = \begin{cases} +1 \text{ if } x_i \prec x_j \\ -1 \text{ if } x_j \prec x_i \end{cases})$$

then apply previous method!

Remarks on ranking SVM

How to use w^* in practice? (

Property: $d \succ_q d'$ iff $\text{sgn}(w^*, \overrightarrow{(d,q)} - \overrightarrow{(d',q)})$ positive

However, a strict application is too costly and one uses the SVM score:

$$RSV(q,d) = (w^* . \overrightarrow{(q,d)})$$

But

- No difference between errors made at the top or at the middle of the list
- Queries with more relevant documents have a stronger impact on w^*

RSVM-IR (1)

Idea: modify the optimization problem so as to take into account the doc ranks ($\tau_{k()}$) and the query type ($\mu_{q()}$)

$$\begin{cases} \text{Minimize} & \frac{1}{2} w^T w + C \sum_l \tau_{k(l)} \mu_{q(l)} \xi_l \\ \text{subject to} & \xi_l \geq 0,\ y^{(l)}(w^*.x^{(l)}) \geq 1 - \xi_l,\ l = 1, \cdots, p \end{cases}$$

where $q(l)$ is the query in the l^{th} example and $k(l)$ is the rank type of the docs in the l^{th} example

RSVM-IR (2)

- Once w^* has been learnt (standard optimization), it is used as in standard RSVM
- The results obtained are state-of-the-art, especially on web-like collections
- *Pairwise* approach, that dispenses with a limited view of relevance (absolute relevance)

General remarks

1. *Listwise* approach: directly treat lists as examples; however no clear gain wrt pairwise approaches
2. Difficulty to rely on an optimal objective function
3. Methods that require *a lot of* annotations

Which training data?

Building training data

- Several annotated collections exist
  - TREC (TREC-vido)
  - CLEF
  - NTCIR

- For new collections, as intranets of companies, such collections do not exist and it may be difficult to build them $\rightarrow$ standard models, with little training

- What about the web?

Training data on the web

- An important source of information; click data from users
  - Use clicks to infer preferences between docs (preference pairs)
  - In addition, and if possible, use eye-tracking data
- What can be deduced from clicks?

Exploiting clicks (1)

Clicks can not be used to infer absolute relevance judgements; they can nevertheless be used to infer relative relevance judgements. Let $(d_1, d_2, d_3, \cdots)$ be an ordered list of documents retrieved for a particular query and let C denote the set of clicked documents. The following strategies can be used to build relative relevance judgements:

1. If $d_i \in C$ and $d_j \notin C$, $d_i \succ_{pert-q} d_j$
2. If d_i is the last clicked doc, $\forall j < i$, $d_j \notin C$, $d_i \succ_{pert-q} d_j$
3. $\forall i \geq 2, d_i \in C, d_{i-1} \notin C, d_i \succ_{pert-q} d_{i-1}$
4. $\forall i, d_i \in C, d_{i+1} \notin C, d_i \succ_{pert-q} d_{i+1}$

Exploiting clicks (2)

- The above strategies yield a partial order between docs
- Leading to a very large training set on which one can deploy learning to rank methods
- IR on the web has been characterized by a "data rush":
  - Index as many pages as possible
  - Get as many click data as possible

Letor

http://research.microsoft.com/en-us/um/beijing/projects/letor/

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. *LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval*, Information Retrieval Journal, 2010

Conclusion on L2R

- Approaches aiming at exploiting all the available information (60 features for the *gov* collection for example - including scores of standard IR models)
- Approaches aiming at "ranking" documents (*pairwise*, *listwise*)
- Many proposals (neural nets, *boosting* and ensemble methods, ...); no clear difference on all collections
- State-of-the-art methods when many features available

References (1)

- Burges et al. *Learning to Rank with Nonsmooth Cost Functions*, NIPS 2006
- Cao et al. *Adapting Ranking SVM to Document Retrieval*, SIGIR 2006
- Cao et al. *Learning to Rank: From Pairwise to Listwise Approach*, ICML 2007
- Goswami et al. *Query-based learning of IR model parameters on unlabelled collections*, ICTIR 2015
- Joachims et al. *Accurately Interpreting Clickthrough Data as Implicit Feedback*, SIGIR 2005
- Liu *Learning to Rank for Information Retrieval*, tutoriel, 2008.
- Manning et al. *Introduction to Information Retrieval*. Cambridge University Press 2008
www-csli.stanford.edu/∼hinrich/information-retrieval-book.html

References (2)

- Nallapati *Discriminative model for Information Retrieval*, SIGIR 2004
- Yue et al. *A Support Vector Method for Optimizing Average Precision*, SIGIR 2007
- Workshop LR4IR, 2007 (Learning to Rank for Information Retrieval).