Accessing Textual Information in Large Scale Collections

ESSCaSS 2016

Eric Gaussier

Univ. Grenoble Alpes - CNRS, INRIA - LIG
Eric.Gaussier@imag.fr

August 2016

Course objectives

- Introduce the main concepts, models and algorithms behind textual information access
- We will focus on:
    - Document indexing and representations
    - Standard models for Information Retrieval (IR)
    - Evaluation of IR systems
    - Learning to rank models
        - Machine learning approach
        - How to exploit user clicks?
    - Text classificaiton (in large scale taxonomies)

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

1. Indexing
2. Standard IR models
3. Evaluation

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

→ 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 → sparse representation

Example

$$\text{document } d \begin{cases} \text{int l} & \text{(doc length)} \\ \text{ArrWords int[l]} & \text{(sorted word indices)} \\ \text{ArrWeights float[l]} & \text{(word weights)} \\ \dots & \end{cases}$$

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:

$$\text{word } i \left\{ \begin{array}{ll} \text{int I} & \text{(number of docs)} \\ \text{ArrDocs int[I]} & \text{(sorted doc indices)} \\ \ldots & \end{array} \right.$$

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

→ *Blocked sort-based indexing* (BSBI) algorithm

BSBI (1)

```
$n \leftarrow 0$
while (some docs have not been processed)
do
    $n \leftarrow n + 1$
    block $\leftarrow$ ParseBlock()
    BSBI-Invert(block)
    WriteBlockToDisk(block,$f_n$)
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	$\cdots$
$\ldots$	$\ldots$	$\ldots$	$\ldots$	

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

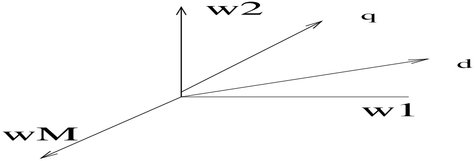
Remark At the basis of many commercial systems

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	→	binary relevance
Vector space model	→	similarity score
Probabilistic model	→	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

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

Language model - QL (1)

Documents are die; a query is a sequence → 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 $P(w|d)$?

→ 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

Content

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

What is the particularity of the web?

→ 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?

→ 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

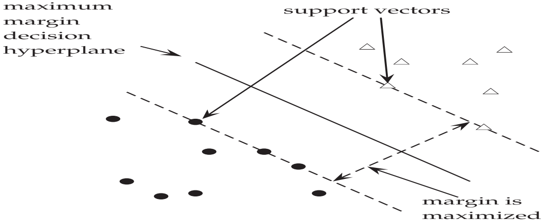
IR and ML: Learning to Rank

Introduction to ML and SVMs (1)

One looks for a decision function that takes the form:

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

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?
→ A (query,doc) pair: $x = (q, d) \in \mathbb{R}^p$
General coordinates (features) $f_i(q, d), i = 1, \cdots, p$, as:

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

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 → *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 → 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).

Part 3: Efficiency and accuracy issues in LSHTC

Content

1. Hierarchical vs flat classification
2. Hierarchy pruning
3. Classification with rare categories

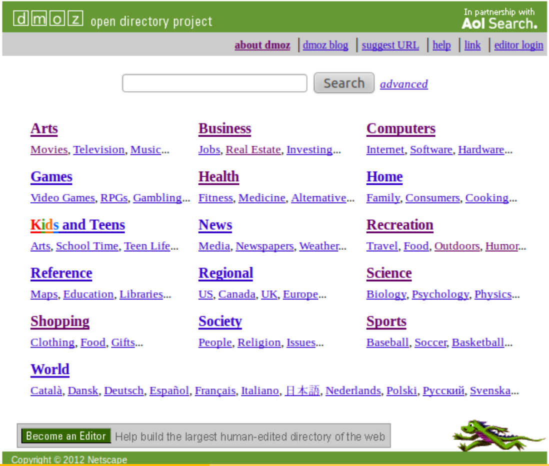
Need for Classification in Big Data

- New data become available:
  - 10,000 articles in **Wikipedia** every day
  - 100 hours every minute in **YouTube**
  - 20,000 scientific articles in **PubMed** every week
- Need for automated methods to categorize these data for:
  - Annotation (enrichment) and archiving purposes
  - Access purposes (ease of retrieval and browsing)

Data Organization

More and more large category systems, with hierarchical structure to organize data

- Directory Mozilla



dmoz open directory project

In partnership with Aol Search.

about dmoz | dmoz blog | suggest URL | help | link | editor login

Search advanced

Arts
Movies, Television, Music...

Business
Jobs, Real Estate, Investing...

Computers
Internet, Software, Hardware...

Games
Video Games, RPGs, Gambling...

Health
Fitness, Medicine, Alternative...

Home
Family, Consumers, Cooking...

Kids and Teens
Arts, School Time, Teen Life...

News
Media, Newspapers, Weather...

Recreation
Travel, Food, Outdoors, Humor...

Reference
Maps, Education, Libraries...

Regional
US, Canada, UK, Europe...

Science
Biology, Psychology, Physics...

Shopping
Clothing, Food, Gifts...

Society
People, Religion, Issues...

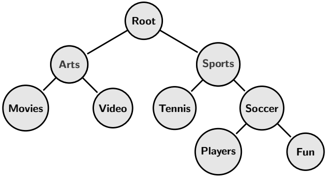
Sports
Baseball, Soccer, Basketball...

World
Català, Dansk, Deutsch, Español, Français, Italiano, 日本語, Nederlands, Polski, Русский, Svenska...

Become an Editor Help build the largest human-edited directory of the web

Copyright © 2012 Netscape

- ca. 5×10^6 sites
- ca. 10^6 categories
- ca. 10^5 editors



Other examples of large scale taxonomies

- Wikipedia: over 600,000 categories organized in a graph (tree backbone)
- Medical Subject Heading[1]: over 27,000 categories organized in a graph (tree backbone)
- International Patent Collection[2]: 60,000 categories in a tree hierarchy
- Amazon (product hierarchy), Yahoo! Directory, ...

[1] https://www.nlm.nih.gov/mesh/

[2] http://www.wipo.int/classifications/ipc/en/

Evaluation challenges for large-scale classification

Recent challenges have been organized to push the state-of-the-art:

- Large Scale Hierarchical Text Classification[3] (2009-2014) : For large-scale text classification in tree and DAG structures
- BioASQ Challenge[4] (2012-2014) : Classification of abstracts of bio-medical data from National Library of Medicine using the Medical Subject Headings Hierarchy
- Large Scale Visual Recognition Challenge [5] (2010-2013) : Classification of Images in Large-scale setup

[3] http://lshtc.iit.demokritos.gr

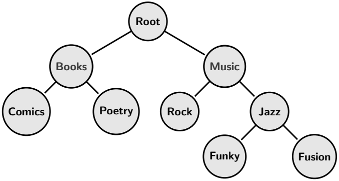
[4] http://www.bioasq.org

[5] http://www.image-net.org/challenges/LSVRC/2013/

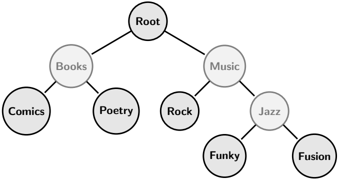
Approaches for classification in large-scale taxonomies

Linear classifiers learned from data

- Hierarchical
  - Top-down - solve individual classification problems at every node (prone to cascading errors) [Liu et al., 2005, Bennett and Nguyen, 2009]
  - Big-bang - solve the problem at once for entire tree (not suitable for large-scale problems) [Cai and Hofmann, 2004, Dekel, 2009, Gopal et al., 2012]



- Flat - ignore the taxonomy *altogether* (higher training and prediction complexities) [Bengio et al., 2010, Gao and Koller, 2011, Perronnin et al., 2012]



- Mildly Hierarchical - use the hierarchical structure *partially* [Malik, 2009]
  - Not clear which layers to remove

Challenges in large-scale classification (1/3)

Which approach to use, flat or hierarchical? Which hierarchy?

Hierarchical vs flat

- *Hierarchical classification*: according to [Liu et al., 2005], outperforms flat classification in terms of classification accuracy and training time on large-scale taxonomies
- *Flat classification*: according to [Bengio et al., 2010], outperforms hierarchical classification on Imagenet dataset
- Build a taxonomy to achieve logarithmic prediction time

Which hierarchy to use?

- Taxonomies are designed by humans for humans - not meant to (fully) optimize classification accuracy
- [Bengio et al., 2010] automatically builds a taxonomy to achieve logarithmic prediction time

Challenges in large-scale classification (2/3)

Scale of datasets

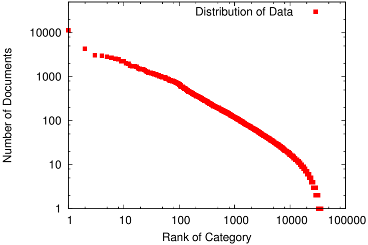
	#Categories	#Features	#Documents	Tree Depth
DMOZ	27,875	594,158	497,992	5
Wiki Small	36,504	346,299	538,148	10
Wiki Large	325,056	1,617,899	2,817,603	14
IPC	451	1,123,497	46,324	3

- 27,875×594,158 = 16,562,154,250 ($\approx$ 16 Billion) parameters when learning One-vs-Rest flat classifier for DMOZ dataset, 525,907,777,344($\approx$ 500 Billion) parameters for Wikipedia dataset (LSHTC)
- 123GB of disk space when using Liblinear to store DMOZ model, 2TB disk space for Wikipedia
- Time complexity for hierarchical lower than for flat ($\mathcal{O}(\log K)$ vs $\mathcal{O}(K)$); What about space complexity?

Challenges in Large-scale classification (3/3)

Rare Category Phenomena

- Distribution of documents among categories in Wikipedia subset exhibits *power-law* phenomena
  - Approximately 15,000 of the 36,000 categories have $\leq$ 5 documents
  - Approximately 4,000 of the 36,000 categories have just 1 document
- Difficulty to learn on such rare categories



Hierarchical vs flat classification

The "hierarchical vs flat" debate (1)

Is it better to use a hierarchical or a flat strategy?

To answer this question

- General framework: flat special case of hierarchical strategy
- Consider linear classifiers (large scale), potentially in feature spaces
- Upper bound on generalization error (concentration inequalities)

Notations (multi-class classification)

- *Training Set:* $\mathcal{S} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{m}, \mathbf{x}^{(i)} \in \mathcal{X} \subseteq \mathbb{R}^d$ and $y^{(i)} \in \mathcal{Y}\}$
- *Linear classifiers ($\mathcal{F}$):* $f(\mathbf{x}, y) = \langle \mathbf{x}, \mathbf{w}_y \rangle$, $\mathbf{w}_y \in \mathbb{R}^d$;
- *Decision function:* $g_f(\mathbf{x}, y) \geq 0$ $(g_f(\mathbf{x}, y) = f(\mathbf{x}, y) - \max_{y' \neq y} f(\mathbf{x}, y'))$
- *Generalization error:* $\mathcal{E}(g_f) = \mathbb{E}_{\sim P(x,y)}[g_f(\mathbf{x}, y) < 0]$
- *Empirical error:* $\mathcal{E}_{emp}(g_f) = \frac{1}{m} \sum_i \mathbf{1}(g_f(\mathbf{x}^{(i)}, y^{(i)}) < 0)$

The "hierarchical vs flat" debate (2)

Some more notations (hierarchical extension)

- *Label Hierarchy:* $\mathcal{G} = (\mathcal{V}, \mathcal{E}); \mathcal{A}(v)$, $\mathcal{N}(v)$, $\mathcal{D}(v)$ (ancestors, sisters, daughters)
- *Target Label Set:* $\mathcal{Y}$=$\{u \in \mathcal{V} : \nexists v \in \mathcal{V}, (u, v) \in \mathcal{E}\} \subseteq \mathcal{V}$
- *Kernel-based classifiers ($\mathcal{F}$):* $f(\mathbf{x}, y) = \langle \phi(\mathbf{x}), \mathbf{w}_y \rangle$
- *Decision function:* $g_f(\mathbf{x}, y) \geq 0$
  $(g_f(\mathbf{x}, y) = \min_{v \in \mathcal{P}(y)} (f(\mathbf{x}, v) - \max_{v' \in \mathcal{N}(v)} f(\mathbf{x}, v'))$
- *Complexity of function class:* Rademacher complexity (McDiarmid concentration inequality)

The "hierarchical vs flat" debate (3)

Theorem: [Babbar et al., 2013] Let $\mathcal{S} = ((\mathbf{x}^{(i)}, y^{(i)}))_{i=1}^{m}$ *be a dataset of* m *examples drawn i.i.d. according to a probability distribution* $\mathcal{D}$ *over* $\mathcal{X} \times \mathcal{Y}$*, and let* $\mathcal{A}$ *be a Lipschitz function with constant* L *dominating the* $0/1$ *loss; further let* $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ *be a PDS kernel and let* $\Phi : \mathcal{X} \rightarrow \mathbb{H}$ *be the associated feature mapping function. Assume that there exists* $R > 0$ *such that* $K(\mathbf{x}, \mathbf{x}) \leq R^2$ *for all* $\mathbf{x} \in \mathcal{X}$*. Then, for all* $1 > \delta > 0$*, with probability at least* $(1 - \delta)$ *the following hierarchical multiclass classification generalization bound holds for all* $g_f \in \mathcal{G}_{\mathcal{F}_B}$ *:*

$$\mathcal{E}(g_f) \leq \frac{1}{m} \sum_{i=1}^{m} \mathcal{A}(g_f(\mathbf{x}^{(i)}, y^{(i)})) + \frac{8BRL}{\sqrt{m}} \sum_{v \in V \setminus \mathcal{Y}} |\mathfrak{D}(v)|(|\mathfrak{D}(v)| - 1) + 3\sqrt{\frac{\ln(2/\delta)}{2m}}$$

where $|\mathfrak{D}(v)|$ *denotes the number of daughters of node* v*.*

Indicates which strategy to adopt on a given taxonomy and suggests ways to improve taxonomies (DMOZ, wikipedia, IPC)

The "hierarchical vs flat" debate (4)

Interpretation

- The complexity term privileges hierarchical structures whereas empirical error term privileges flat structures (error propagation), esp. when classes are well balanced
- Trade-off between empirical error and complexity terms

illustration

Dataset	# Tr.	# Test	# Classes	# Feat.	**CR**	**Emp ER**
LSHTC2-3	38,725	10,102	3,956	145,354	**0.004**	**2.65**
LSHTC2-4	27,924	7,026	2,544	123,953	**0.005**	**1.8**
LSHTC2-5	68,367	17,561	7,212	192,259	**0.002**	**2.12**
IPC	46,324	28,926	451	1,123,497	**0.02**	**12.27**

Table: **CR**: complexity ratio H/F; **Emp ER** emp. error ratio H/F

The "hierarchical vs flat" debate (5)

Validation through classification results

	LSHTC2-3		**LSHTC2-4**		**LSHTC2-5**		**IPC**	
	MLR	SVM	MLR	SVM	MLR	SVM	MLR	SVM
FL	0.528	0.535	0.497	0.501	0.542	0.547	**0.546**†	**0.446**†
RN	0.493	0.517	0.478	0.484	0.532	0.536	0.547	0.458
FH	**0.484**†	**0.498**†	**0.473**†	**0.476**†	**0.526**†	**0.527**†	0.552	0.465
PR-B	**0.481**	**0.495**	**0.466**	**0.465**	**0.522**	**0.522**	0.546	0.450
PR-M	**0.480**	**0.493**	**0.469**	**0.472**	**0.522**	**0.523**	0.544	0.450

Table: Error results across all datasets. Bold typeface is used for the best results. Statistical significance (using micro sign test (s-test) Yang and Liu [1999]) is denoted with † for p-value$<$0.05

Hierarchy pruning

Pruning a hierarchy (speed/accuracy trade-off)

Above result suggests that one can gain in accuracy by *flattening* a hierarchy; however bound not direcly usable. Can we derive more exploitable bounds, for generative and discriminative classifiers?

Theorem: [Babbar et al., 2013] $\forall \epsilon > 0$, $v \in V \setminus \mathcal{Y}$, *one has:*

$$\mathcal{E}(h_m^{\mathfrak{S}'_v}) + \mathcal{E}(h_m^{\mathfrak{D}_v}) \leq \mathcal{E}(h_\infty^{\mathfrak{S}'_v}) + \mathcal{E}(h_\infty^{\mathfrak{D}_v}) + G_{\mathfrak{S}'(v)}(\epsilon) + G_{\mathfrak{D}(v)}(\epsilon)$$

with probability at least $1 - \left(\frac{Rd^2|\mathfrak{S}'(v)|\sigma_0^{\mathfrak{S}'(v)}}{m_{\mathfrak{S}'(v)}\epsilon^2} + \frac{Rd^2|\mathfrak{D}(v)|\sigma_0^{\mathfrak{D}(v)}}{m_{\mathfrak{D}(v)}\epsilon^2} \right)$ *for MLR classifiers and with probability at least* $1 - \left(\delta_{\mathfrak{S}'(v)} + \delta_{\mathfrak{D}(v)}\right)$ *for Naive Bayes classifiers, with:*

$$\delta_{\mathcal{Y}} = 2K \exp\left(\frac{-2\epsilon^2 m}{C(d + n_{max})^2}\right) + 2d \exp\left(\frac{-2\epsilon^2 n_{min}}{C(d + n_{max})^2}\right)$$

Pruning a hierarchy (speed/accuracy trade-off) (2)

1. Above bound suggests ingredients that can be used to determine whether a node should be removed or not:
   - Number of categories, number of examples in different category sets
   - Dimension of feature space in different category sets
   - Confusion between categories
2. Learn a binary (meta-)classifier on some taxonomies (if pruning a node *v* leads to significantly better accuracy, *v* is labelled 1, 0 otherwise)
3. Apply meta-classifier to new taxonomies (number of nodes pruned depends on the policy - conservative or not)

Results on Classification Error

	LSHTC2-3		LSHTC2-4		LSHTC2-5		IPC	
	MLR	SVM	MLR	SVM	MLR	SVM	MLR	SVM
FL	0.528	0.535	0.497	0.501	0.542	0.547	0.546	**0.446**
RN	0.493	0.517	0.478	0.484	0.532	0.536	0.547	0.458
FH	0.484	0.498	0.473	0.476	0.526	0.527	0.552	0.465
PR-M	**0.480**†	**0.493**†	**0.469**†	**0.472**†	**0.522**†	**0.523**	**0.544**	0.450
PR-B	**0.481**	**0.495**	**0.466**	**0.465**	**0.522**	**0.522**	0.546	0.450

Table: Error results across all datasets, bold typeface is used for the best results. Statistical significance (using micro sign test(s-test) [Yang and Liu, 1999]) is denoted with † for p-value$<$0.05

- Pruning the taxonomy using the proposed meta-learning strategy improves classification accuracy
- Another strategy based on directly using the rademacher-based bound can also be applied for pruning

Results on Classification Error

	LSHTC2-3		LSHTC2-4		LSHTC2-5		IPC	
	MLR	SVM	MLR	SVM	MLR	SVM	MLR	SVM
FL	0.528	0.535	0.497	0.501	0.542	0.547	0.546	**0.446**
RN	0.493	0.517	0.478	0.484	0.532	0.536	0.547	0.458
FH	0.484	0.498	0.473	0.476	0.526	0.527	0.552	0.465
PR-M	**0.480**†	**0.493**†	**0.469**†	**0.472**†	**0.522**†	**0.523**	**0.544**	0.450
PR-B	**0.481**	**0.495**	**0.466**	**0.465**	**0.522**	**0.522**	0.546	0.450

Table: Error results across all datasets, bold typeface is used for the best results. Statistical significance (using micro sign test(s-test) [Yang and Liu, 1999]) is denoted with † for p-value$<$0.05

- Pruning the taxonomy using the proposed meta-learning strategy improves classification accuracy
- Another strategy based on directly using the rademacher-based bound can also be applied for pruning

Space complexity of flat and hierarchical methods

For power law distributed category systems (incl. all textual category systems), the space complexity of the hierarchical approaches is lower than the one of the flat approaches

[Babbar et al., 2014a]

Space-complexity for Top-down Classification

- Using Heap's law, it can be shown that distribution of features exhibit a fit to power-law l, i.e., $d_{l,r} \approx d_{l,1} r^{-\beta_l}$
- Size of the top-down model is given by:

$$Size_{hier} = \sum_{l=1}^{L-1} \sum_{r=1}^{B_l} b_{l,r} d_{l,r} \approx \sum_{l=1}^{L-1} \sum_{r=1}^{B_l} b_{l,r} d_{l,1} r^{-\beta_l}$$

where $b_{l,r}$ represent the branching factor for the r-th ranked category, and B_l the total number of categories at level l.

Proposition ([Babbar et al., 2014a])

For a hierarchy of categories of depth L and K leaves, let $\beta = \min_{1 \leq l \leq L} \beta_l$ and $b = \max_{l,r} b_{l,r}$. Then,

$$For\ \beta > 1, if\ \beta > \frac{K}{K - b(L-1)} (> 1),\ then\ \ Size_{hier} < Size_{flat}$$

$$For\ 0 < \beta < 1, if\ \frac{b^{(L-1)(1-\beta)} - 1}{(b^{(1-\beta)} - 1)} < \frac{1-\beta}{b} K, then\ Size_{hier} < Size_{flat}$$

Datasets and Space-complexities

Dataset	Training/Test	Categories	Features	Tree Depth
LSHTC1-large	93,805/34,880	12,294	347,255	6
LSHTC2-a	25,310/6,441	1,789	145,859	6
LSHTC2-b	36,834/9,605	3,672	145,354	6
IPC	46,324/28,926	451	1,123,497	4

Dataset	$Size_{hier}$	$Size_{Flat}$	β	b	$\bigtriangledown$
LSHTC1-large	**2.8**	90.0	1.62	344	1.12
LSHTC2-a	**0.46**	5.4	1.35	55	1.14
LSHTC2-b	**1.1**	11.9	1.53	77	1.09
IPC	**3.6**	10.5	2.03	34	1.17

Table: Model size (in GB) for flat and hierarchical models, $\bigtriangledown$ refers to the quantity $\frac{K}{K-b(L-1)}$

Challenges in Large-scale Classification

- We addressed two challenges
  - Which approach to use, flat or hierarchical? Which hierarchy?
    → Presented theoretical explanation of when flat or hierarchical is to be preferred
    → Taxonomy adaptation through node pruning to select *better* taxonomies
  - Scale of datasets: is hierarchical really faster?
    → Yes (better time and space complexities)
- Third challenge: Dealing with rare categories

Classification with rare categories

Rare category detection challenge

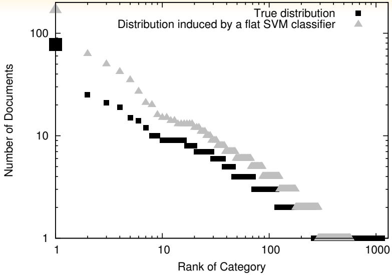


Figure: Comparison of true and induced test-set distributions

- Left part: induced distribution higher
  ⟹ high false positive rate for large categories
- Right part: tail of induced dist. too short
  ⟹ high false negative rate for small categories
- Out of 1139 classes, only 574 are discovered in the test set
  ⟹ low values of Macro and Micro F1

Rare category detection challenge

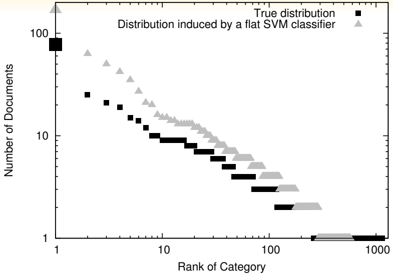


Figure: Comparison of true and induced test-set distributions

- Left part: induced distribution higher $\Longrightarrow$ high false positive rate for large categories
- Right part: tail of induced dist. too short $\Longrightarrow$ high false negative rate for small categories
- Out of 1139 classes, only 574 are discovered in the test set $\Longrightarrow$ low values of Macro and Micro F1

Rare category detection challenge

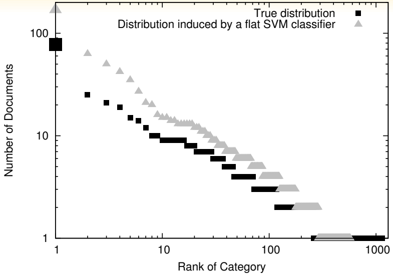


Figure: Comparison of true and induced test-set distributions

- Left part: induced distribution higher
  ⟹ high false positive rate for large categories
- Right part: tail of induced dist. too short
  ⟹ high false negative rate for small categories
- Out of 1139 classes, only 574 are discovered in the test set
  ⟹ low values of Macro and Micro F1

Rare category detection challenge

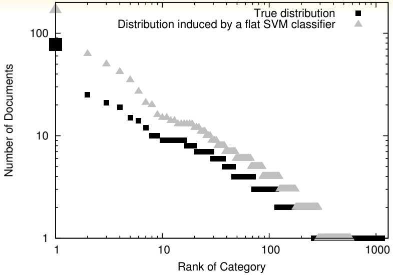


Figure: Comparison of true and induced test-set distributions

- Left part: induced distribution higher
  $\Longrightarrow$ high false positive rate for large categories
- Right part: tail of induced dist. too short
  $\Longrightarrow$ high false negative rate for small categories
- Out of 1139 classes, only 574 are discovered in the test set
  $\Longrightarrow$ low values of Macro and Micro F1

Soft-thresholding for prediction with rare categories

1. Can one quantify the difference in the distribution induced by a classifier and the true distribution on the test set?
2. Can this be related to an upper bound on the test-set accuracy of any classifier C, using the training set only?

Theorem ([Babbar et al., 2014b])

Let $S = \{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{M}$ be the test set generated i.i.d. from $\mathcal{D}$. Let M_ℓ^C be the number of examples in S assigned to category y_ℓ by the classifier C which is trained on S_{train}. Then the following bound on the accuracy of C over S, denoted by $Acc(C)$, holds with high probability :

$$Acc(C) \leq \frac{1}{|S|} \sum_{\ell=1}^{|\mathcal{Y}|} \min\{(\hat{p}_{y_\ell} \times |S|), M_\ell^C\} \triangleq B(Acc(C))$$

where $\hat{p}_{y_\ell}$ denotes the estimate on the prior probability of the category y_ℓ in the training set.

Algorithm to achieve higher bound value

```
Input: Training data $S_{train}$ and Test data $S_{test}$
  Learn Multiclass SVM (Crammer-Singer)
  for each test instance $\mathbf{x} \in S_{test}$ do
      Predict posterior probabilities $(\hat{p}_{y_l}|\mathbf{x}), \forall 1 \leq l \leq |\mathcal{Y}|$
      if $pred(\mathbf{x})$ is true then
          Create instantaneous training set $t$ (odd) times
          To distinguish $\{y_{r1}, y_{r2}\}$, learn $t$ binary classifiers
          Re-predict instance $\mathbf{x}$ with each binary classifier
          Output from $\{y_{r1}, y_{r2}\}$ the one with majority votes
      else
          Output category $\arg\max_{y_l \in \mathcal{Y}} (\hat{p}_{y_l}|\mathbf{x})$
      end if
  end for
  return Labels $\forall \mathbf{x} \in S_{test}$
```

$pred(\mathbf{x})$ is true iff $(\hat{p}_{y_{r1}}|\mathbf{x}) - (\hat{p}_{y_{r2}}|\mathbf{x}) \leq \Delta \quad \&\& \quad N_{y_{r1}}/N_{y_{r2}} \geq R$

Empirical Evaluation

Dataset	Training/Test instances	Categories $|\mathcal{Y}|$	Features d
DMOZ-2010-s	4,463/1858	1,139	51,033
DMOZ-2010-l	128,710/34,880	12,294	381,580
DMOZ-2012	383,408/103,435	11,947	348,548

Table: Datasets and their statistics

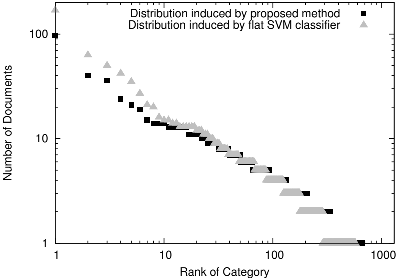
- Datasets are in the form of libSVM format with single label for each training instance
- Feature set size equals the size of the size of the vocabulary

Empirical Results

Dataset	Proposed Algorithm	HR-SVM	CS-SVM
DMOZ-2010-s			
Micro-F1	**47.36**†	45.31	45.15
Macro-F1	**32.91**$^{\downarrow}$	28.94	29.40
B(Acc(C))	**0.71**	0.63	0.64
Categories detected	**658**	570	574
Training Time	**1.1x**	1.7x	1x
DMOZ-2010-l			
Micro-F1	**46.67**†	46.02	45.82
Macro-F1	**34.65**$^{\downarrow}$	33.12	32.63
B(Acc(C))	**0.77**	0.73	0.72
Categories detected	**8523**	8102	8039
Training Time	**1.1x**	1.6x	1x
DMOZ-2012			
Micro-F1	**57.78**†	57.17	56.44
Macro-F1	**34.15**$^{\downarrow}$	33.05	31.59
B(Acc(C))	**0.76**	0.72	0.70
Categories detected	**8220**	7965	7882
Training Time	**1.1x**	1.6x	1x

Table: The significance-test results (using micro sign test (s-test) and macro t-test) are denoted for a p-value less than 1%.

Comparison of induced distributions



- $B(Acc(C)) = 0.71 > 0.64$ (for Flat Classifier)
- Out of 1139, # detected classes $= 658 > 574$ (for Flat Classifier)
- Re-prediction required for approx. 10% of test instances, hence does not impact it adversely

Conclusion

- Flat versus Top-down classification
  - Generalization error bounds to theoretically explain the performance of various methods
  - Hierarchy pruning strategy for improvement in classification accuracy
  - Proof that hierarchical approaches *really* faster than flat ones
- Classification with rare categories
  - Soft-thresholding based algorithm for classification with rare categories
- Datasets available from LSHTC challenges (http://lshtc.iit.demokritos.gr/) and BioASQ (http://bioasq.org/)

Rohit Babbar, Ioannis Partalas, Eric Gaussier, and Massih-Reza Amini. On flat versus hierarchical classification in large-scale taxonomies. In **Advances in Neural Information Processing Systems**, pages 1824–1832, 2013.

Rohit Babbar, Cornelia Metzig, Ioannis Partalas, Eric Gaussier, and Massih-Reza Amini. On power law distributions in large-scale taxonomies. **ACM SIGKDD Explorations Newsletter**, 16(1):47–56, 2014a.

Rohit Babbar, Ioannis Partalas, Eric Gaussier, and Massih-reza Amini. Re-ranking approach to classification in large-scale power-law distributed category systems. In **Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval**, pages 1059–1062. ACM, 2014b.

Samy Bengio, Jason Weston, and David Grangier. Label embedding trees for large multi-class tasks. In **Neural Information Processing Systems**, pages 163–171, 2010.

Paul N. Bennett and Nam Nguyen. Refined experts: improving classification in large taxonomies. In **Proceedings of the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval**, pages 11–18, 2009.

Lijuan Cai and Thomas Hofmann. Hierarchical document categorization with support vector machines. In **Proceedings of the thirteenth ACM international conference on Information and knowledge management**, pages 78–87, 2004.

Ofer Dekel. Distribution-calibrated hierarchical classification. In **Advances in Neural Information Processing Systems 22**, pages 450–458. 2009.

Tianshi Gao and Daphne Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In **IEEE International Conference on Computer Vision (ICCV)**, pages 2072–2079, 2011.

Siddharth Gopal, Yiming Yang, Bing Bai, and Alexandru Niculescu-Mizil. Bayesian models for large-scale hierarchical classification. In **Neural Information Processing Systems**, 2012.

Tie-Yan Liu, Yiming Yang, Hao Wan, Hua-Jun Zeng, Zheng Chen, and Wei-Ying Ma. Support vector machines classification with a very large-scale taxonomy. **SIGKDD**, 2005.

Hassan Malik. Improving hierarchical svms by hierarchy flattening and lazy classification. In **1st Pascal Workshop on Large Scale Hierarchical Classification**, 2009.

Florent Perronnin, Zeynep Akata, Zaïd Harchaoui, and Cordelia Schmid. Towards good practice in large-scale learning for image classification. In **Computer Vision and Pattern Recognition**, pages 3482–3489, 2012.

Yiming Yang and Xin Liu. A re-examination of text categorization methods. In **Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval**, pages 42–49. ACM, 1999.

Thank you!