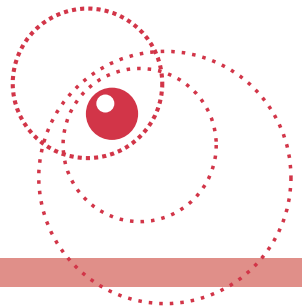


Web mining

Information Retrieval

28 September 2011

Pierre Senellart

TELECOM ParisTech

Problem

How to index Web content so as to answer (keyword-based) queries efficiently?

Context: set of text documents

- d_1 The jaguar is a New World mammal of the Felidae family.
- d_2 Jaguar has designed four new engines.
- d_3 For Jaguar, Atari was keen to use a 68K family device.
- d_4 The Jacksonville Jaguars are a professional US football team.
- d_5 Mac OS X Jaguar is available at a price of US $199 for Apple's new "family pack".
- d_6 One such ruling family to incorporate the jaguar into their name is Jaguar Paw.
- d_7 It is a big cat.

Initial text preprocessing steps

- Number of optional steps
- Highly depends on the application
- Highly depends on the document language (illustrated with English)

How to find the language used in a document?

- Meta-information about the document: often not reliable!
- Unambiguous scripts or letters: not very common!

한글

カタカナ

ⵜⵉⴼⵉⵏⴰⵖ

Għarbi

þorn

How to find the language used in a document?

- Meta-information about the document: often not reliable!
- Unambiguous scripts or letters: not very common!

한글
カタカナ
ދިވެހި
Għarbi
Þorn

Respectively: Korean Hangul, Japanese Katakana, Maldivian Dhivehi, Maltese, Icelandic

- Extension of this: frequent characters, or, better, frequent *k*-grams
- Use standard machine learning techniques (classifiers)

Separate text into tokens (words)

Not so easy!

- In some languages (Chinese, Japanese), words not separated by whitespace
- Deal consistently with acronyms, elisions, numbers, units, URLs, emails, etc.
- Compound words: *hostname*, *host-name* and *host name*. Break into two tokens or regroup them as one token? In any case, lexicon and linguistic analysis needed! Even more so in other languages as German.

Usually, remove punctuation and normalize case at this point

d_1 the$_1$ jaguar$_2$ is$_3$ a$_4$ new$_5$ world$_6$ mammal$_7$ of$_8$ the$_9$ felidae$_{10}$ family$_{11}$

d_2 jaguar$_1$ has$_2$ designed$_3$ four$_4$ new$_5$ engines$_6$

d_3 for$_1$ jaguar$_2$ atari$_3$ was$_4$ keen$_5$ to$_6$ use$_7$ a$_8$ 68k$_9$ family$_{10}$ device$_{11}$

d_4 the$_1$ jacksonville$_2$ jaguars$_3$ are$_4$ a$_5$ professional$_6$ us$_7$ football$_8$ team$_9$

d_5 mac$_1$ os$_2$ x$_3$ jaguar$_4$ is$_5$ available$_6$ at$_7$ a$_8$ price$_9$ of$_{10}$ us$_{11}$ \$199$_{12}$ for$_{13}$ apple's$_{14}$ new$_{15}$ family$_{16}$ pack$_{17}$

d_6 one$_1$ such$_2$ ruling$_3$ family$_4$ to$_5$ incorporate$_6$ the$_7$ jaguar$_8$ into$_9$ their$_{10}$ name$_{11}$ is$_{12}$ jaguar$_{13}$ paw$_{14}$

d_7 it$_1$ is$_2$ a$_3$ big$_4$ cat$_5$

Need to “normalize” terms in indexed text as well as query terms into the same form.

- Example: We want to match *U.S.A.* and *USA*
- We most commonly implicitly define equivalence classes of terms.
- Alternatively: do asymmetric expansion
  - window $\rightarrow$ window, windows
  - windows $\rightarrow$ Windows, windows
  - Windows (no expansion)
- More powerful, but less efficient

Exercise

Why don’t you want to put *window*, *Window*, *windows*, and *Windows* in the same equivalence class?

- Accents: résumé vs. resume (simple omission of accent)
- Umlauts: Universität vs. Universitaet (substitution with special letter sequence “ae”)
- Most important criterion: How are users likely to write their queries for these words?
- Even in languages that standardly have accents, users often do not type them. (Polish?)
- Normalization and language detection interact.
- *PETER WILL NICHT MIT.* $\rightarrow$ MIT = mit
- *He got his PhD from MIT.* $\rightarrow$ MIT $\neq$ mit

Principle

Merge different forms of the same word, or of closely related words, into a single stem

- Not in all applications!
- Useful for retrieving documents containing *geese* when searching for *goose*
- Various degrees of stemming
- Possibility of building different indexes, with different stemming

Morphological stemming (lemmatization).

- Remove bound morphemes from words:
  - plural markers
  - gender markers
  - tense or mood inflections
  - etc.
- Can be linguistically very complex, cf:

  Les poules du couvent couvent.

  [The hens of the monastery brood.]
- In English, somewhat easy:
  - Remove final -s, -'s, -ed, -ing, -er, -est
  - Take care of semiregular forms (e.g., -y/-ies)
  - Take care of irregular forms (mouse/mice)
- But still some ambiguities: cf rose

Lemmatisation.

- Merge lexically related terms of various parts of speech, such as *policy*, *politics*, *political* or *politician*
- For English, Porter's stemming [Porter, 1980]; stem *university* and *universal* to *univers*: not perfect!
- Possibility of coupling this with lexicons to merge (near-)synonyms

Phonetic stemming.

- Merge phonetically related words: search proper names with different spellings!
- For English, Soundex [US National Archives and Records Administration, 2007] stems *Robert* and *Rupert* to *R163*. Very coarse!

d_1 the$_1$ jaguar$_2$ be$_3$ a$_4$ new$_5$ world$_6$ mammal$_7$ of$_8$ the$_9$ felidae$_{10}$ family$_1$

d_2 jaguar$_1$ have$_2$ design$_3$ four$_4$ new$_5$ engine$_6$

d_3 for$_1$ jaguar$_2$ atari$_3$ be$_4$ keen$_5$ to$_6$ use$_7$ a$_8$ 68k$_9$ family$_{10}$ device$_{11}$

d_4 the$_1$ jacksonville$_2$ jaguar$_3$ be$_4$ a$_5$ professional$_6$ us$_7$ football$_8$ team$_9$

d_5 mac$_1$ os$_2$ x$_3$ jaguar$_4$ be$_5$ available$_6$ at$_7$ a$_8$ price$_9$ of$_{10}$ us$_{11}$ \$199$_{12}$ for$_{13}$ apple$_{14}$ new$_{15}$ family$_{16}$ pack$_{17}$

d_6 one$_1$ such$_2$ rule$_3$ family$_4$ to$_5$ incorporate$_6$ the$_7$ jaguar$_8$ into$_9$ their$_{10}$ name$_{11}$ be$_{12}$ jaguar$_{13}$ paw$_{14}$

d_7 it$_1$ be$_2$ a$_3$ big$_4$ cat$_5$

Principle

Remove uninformative words from documents, in particular to lower the cost of storing the index

determiners: *a*, *the*, *this*, etc.

function verbs: *be*, *have*, *make*, etc.

conjunctions: *that*, *and*, etc.

etc.

d_1 jaguar_2 new_5 world_6 mammal_7 felidae_{10} family_{11}

d_2 jaguar_1 design_3 four_4 new_5 engine_6

d_3 jaguar_2 atari_3 keen_5 68k_9 family_{10} device_{11}

d_4 jacksonville_2 jaguar_3 professional_6 us_7 football_8 team_9

d_5 mac_1 os_2 x_3 jaguar_4 available_6 price_9 us_{11} $\$199_{12}$ apple_{14} new_{15} family_{16} pack_{17}

d_6 one_1 such_2 rule_3 family_4 incorporate_6 jaguar_8 their_{10} name_{11} jaguar_{13} paw_{14}

d_7 big_4 cat_5

After all preprocessing, construction of an inverted index:

- Index of all terms, with the list of documents where this term occurs
- Small scale: disk storage, with memory mapping (cf. `mmap`) techniques; secondary index for offset of each term in main index
- Large scale: distributed on a cluster of machines; hashing gives the machine responsible for a given term
- Updating the index costly, so only batch operations (not one-by-one addition of term occurrences)

family	d_1, d_3, d_5, d_6
football	d_4
jaguar	$d_1, d_2, d_3, d_4, d_5, d_6$
new	d_1, d_2, d_5
rule	d_6
us	d_4, d_5
world	d_1
...	

- phrase queries, NEAR operator: need to keep position information in the index
- just add it in the document list!

family	$d_1/11$, $d_3/10$, $d_5/16$, $d_6/4$
football	$d_4/8$
jaguar	$d_1/2$, $d_2/1$, $d_3/2$, $d_4/3$, $d_5/4$, $d_6/8 + 13$
new	$d_1/5$, $d_2/5$, $d_5/15$
rule	$d_6/3$
us	$d_4/7$, $d_5/11$
world	$d_1/6$
...	

- Some term occurrences have more weight than others:
  - Terms occurring frequently in a given document: more relevant
  - Terms occurring rarely in the document collection as a whole: more informative
- Add Term Frequency—Inverse Document Frequency weighting to occurrences;

$$\text{tfidf}(t, d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}} \cdot \log \frac{|D|}{|\{d' \in D \mid n_{t,d'} > 0\}|}$$

$n_{t,d}$	number of occurrences of t in d
D	set of all documents

- Store documents (along with weight) in decreasing weight order in the index

family	$d_1/11/.13$, $d_3/10/.13$, $d_6/4/.08$, $d_5/16/.07$
football	$d_4/8/.47$
jaguar	$d_1/2/.04$, $d_2/1/.04$, $d_3/2/.04$, $d_4/3/.04$, $d_6/8+13/.04$, $d_5/4/.02$
new	$d_2/5/.24$, $d_1/5/.20$, $d_5/15/.10$
rule	$d_6/3/.28$
us	$d_4/7/.30$, $d_5/11/.15$
world	$d_1/6/.47$
...	

- Single keyword query: just consult the index and return the documents in index order.
- Boolean multi-keyword query

(*jaguar* AND *new* AND NOT *family*) OR *cat*

Same way! Retrieve document lists from all keywords and apply adequate set operations:

AND	intersection
OR	union
AND NOT	difference

- Global score: some function of the individual weight (e.g., addition for conjunctive queries)
- Position queries: consult the index, and filter by appropriate condition

$$t_1 \text{ AND } \ldots \text{ AND } t_n$$

$$t_1 \text{ OR } \ldots \text{ OR } t_n$$

Problem

Find the top-k results (for some given k) to the query, without retrieving all documents matching it.

Notations:

$s(t, d)$ weight of t in d (e.g., tfidf)

$g(s_1, \ldots, s_n)$ monotonous function that computes the global score (e.g., addition)

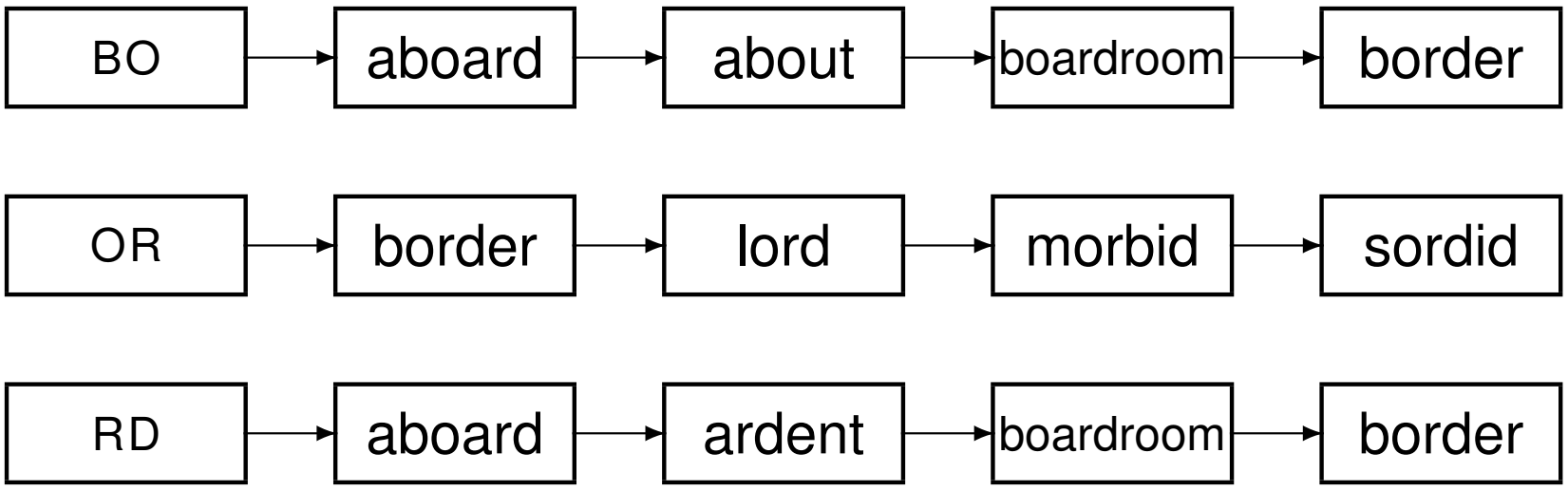
(with an additional direct index giving $s(t, d)$)

1. Let R be the empty list and $m = +\infty$.
2. For each $1 \leq i \leq n$:
   - 2.1 Retrieve the document $d^{(i)}$ containing term t_i that has the next largest $s(t_i, d^{(i)})$.
   - 2.2 Compute its global score $g_{d^{(i)}} = g(s(t_1, d^{(i)}), \ldots, s(t_n, d^{(i)}))$ by retrieving all $s(t_j, d^{(i)})$ with $j \neq i$.
   - 2.3 If R contains less than k documents, or if $g_{d^{(i)}}$ is greater than the minimum of the score of documents in R, add $d^{(i)}$ to R (and remove the worst element in R if it is full).
3. Let $m = g(s(t_1, d^{(1)}), s(t_2, d^{(2)}), \ldots, s(t_n, d^{(n)}))$.
4. If R contains k documents, and the minimum of the score of the documents in R is greater than or equal to m, return R.
5. Redo step 2.

(no additional direct index needed)

1. Let R be the empty list and $m = +\infty$.
2. For each document d, maintain $W(d)$ as its worst possible score, and $B(d)$ as its best possible score.
3. At the beginning, $W(d) = 0$ and $B(d) = g(s(t_1, d(1)) \dots s(t_n, d(n))$.
4. Then, access the next best document for each token, in a round-robin way (t_1, $t_2 \dots t_n$, then t_1 again, etc.)
5. Update the $W(d)$ and $B(d)$ lists each time, and maintain R as the list of k documents with best $W(d)$ scores (solve ties with $B(d)$), and m as the minimum value for $W(d)$ in R.
6. Stop when R contains at least k documents, and all documents outside of R verify $B(d) \leq m$.

- Problem: able to deal with incorrectly spelled terms in documents, or variants in spelling
- Enumerate all *k*-grams in the query term
- Use the *k*-gram index to retrieve “correct” words that match query term *k*-grams
- Threshold by number of matching *k*-grams
- E.g., only vocabulary terms that differ by at most 3 *k*-grams
- Example: bigram index, misspelled word *bordroom*
- Bigrams: *bo, or, rd, dr, ro, oo, om*



- Issue: Fixed number of k-grams that differ does not work for words of differing length.
- Suppose the correct word is NOVEMBER
- Trigrams: *nov*, *ove*, *vem*, *emb*, *mbe*, *ber*
- And the query term is DECEMBER
- Trigrams: *dec*, *ece*, *cem*, *emb*, *mbe*, *ber*
- So 3 trigrams overlap (out of 6 in each term)
- How can we turn this into a normalized measure of overlap?

- Issue: Fixed number of k-grams that differ does not work for words of differing length.
- Suppose the correct word is NOVEMBER
- Trigrams: *nov*, *ove*, *vem*, *emb*, *mbe*, *ber*
- And the query term is DECEMBER
- Trigrams: *dec*, *ece*, *cem*, *emb*, *mbe*, *ber*
- So 3 trigrams overlap (out of 6 in each term)
- How can we turn this into a normalized measure of overlap?
  $\rightarrow$ Use Jaccard coefficent!

- *asteroid that fell form the sky*
- How can we correct *form* here?
- One idea: hit-based spelling correction
  - Retrieve “correct” terms close to each query term
  - for *flew form munich*: *flea* for *flew*, *from* for *form*, *munch* for *munich*
  - Now try all possible resulting phrases as queries with one word “fixed” at a time
  - Try query *“flea form munich”*
  - Try query *“flew from munich”*
  - Try query *“flew form munch”*
  - The correct query *“flew from munich”* has the most hits.
- The “hit-based” algorithm we just outlined is not very efficient.
- More efficient alternative: look at “collection” of queries, not documents

web news images wikipedia blogs jobs more »

jaguar Search

clusters sources sites

All Results (232)

- Jaguar Cars (33)
- Parts (39)
- Club (33)
- Photos (28)
- Panthera onca (15)
- Land Rover (16)
- Jacksonville Jaguars (12)
- Defensive, Falcons (7)
- Atari, Game (10)
- Classic Jaguar (6)

Top **232** results of at least **13,030,000** retrieved for the query **jaguar** (definition) (details)

Search Results

1. **jaguars**.com -- The official web site of the NFL's Jacksonville **Jaguars**
   The official team site with scores, news items, game schedule, and roster.
   www.jaguars.com - [cache] - Live, Open Directory, Ask

2. **Jaguar**

   The **jaguar** (*Panthera onca*) is a large member of the cat family native to warm regions of the Americas. It is closely related to the lion, tiger, and leopard of the Old World, and is the largest species of the cat family found in the Americas.
   en.wikipedia.org/wiki/Jaguar - [cache] - Wikipedia, Ask, Live

3. **Jaguar** Enthusiasts' Club
   World's largest **Jaguar** / Daimler Club ... Largest **Jaguar** Club in the World, serving over 20,000 members ...
   www.jec.org.uk - [cache] - Ask, Open Directory

4. US abandons bid for **jaguar** recovery plan
   Jan 18, 2008 - The Interior Department has abandoned attempts to craft a recovery plan for the endangered **jaguar** because too few of the rare cats have been spotted along the Southwest region of New Mexico and Arizona to warrant such action. Some critics of the decision said Thursday the **jaguar** is being sacrificed for the government's new border fence, which is going up along many of the same areas where the ... has crossed into the United States from Mexico. If the U.S. border areas

- Document Vector Space model:

  terms dimensions

  documents vectors

  coordinates weights

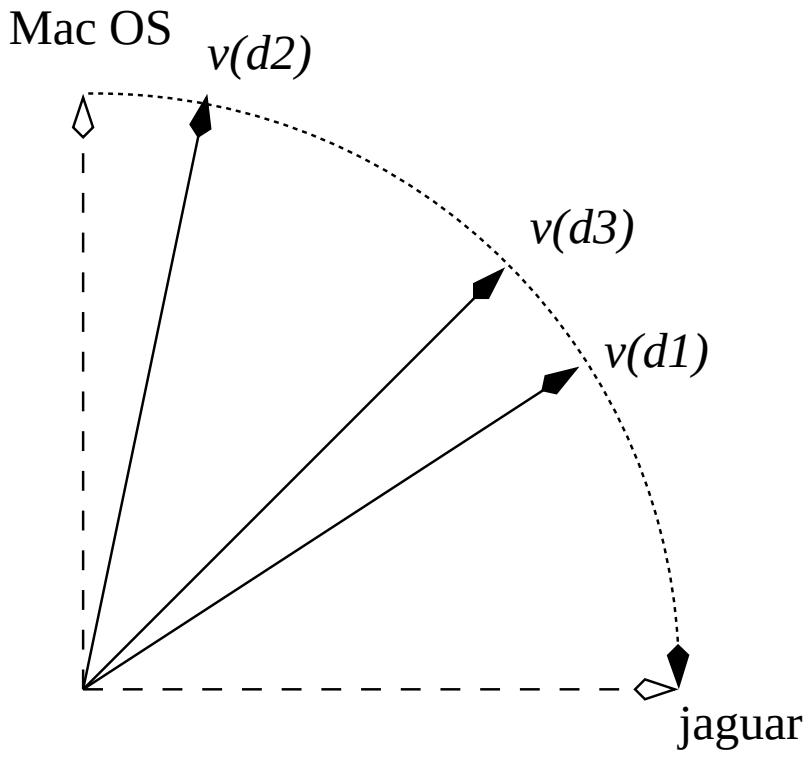
  (The projection of document d along coordinate t is the weight of t in d, say tfidf(t, d))
- Similarity between documents d and d': cosine of these two vectors

$$\cos(d, d') = \frac{d \cdot d'}{\|d\| \times \|d'\|}$$

$d \cdot d'$ scalar product of d and d'

$\|d\|$ norm of vector d

- $\cos(d, d) = 1$
- $\cos(d, d') = 0$ if d and d' are orthogonal (do not share any common term)



1. Initially, each document forms its own cluster.
2. The similarity between two clusters is defined as the maximal similarity between elements of each cluster.
3. Find the two clusters whose mutual similarity is highest. If it is lower than a given threshold, end the clustering. Otherwise, regroup these clusters. Repeat.

Remark

Many other more refined algorithms for clustering exist.

The Inverted Index Model

Clustering

Indexing Other Media

Measuring the quality of results

Conclusion

- HTML: text + meta-information + structure
- Possibly: separate index for meta-information (title, keywords)
- Increase weight of structurally emphasized content in index
- Tree structure can also be queried with XPath or XQuery, but not very useful on the Web as a whole, because of tag soup and lack of consistency.

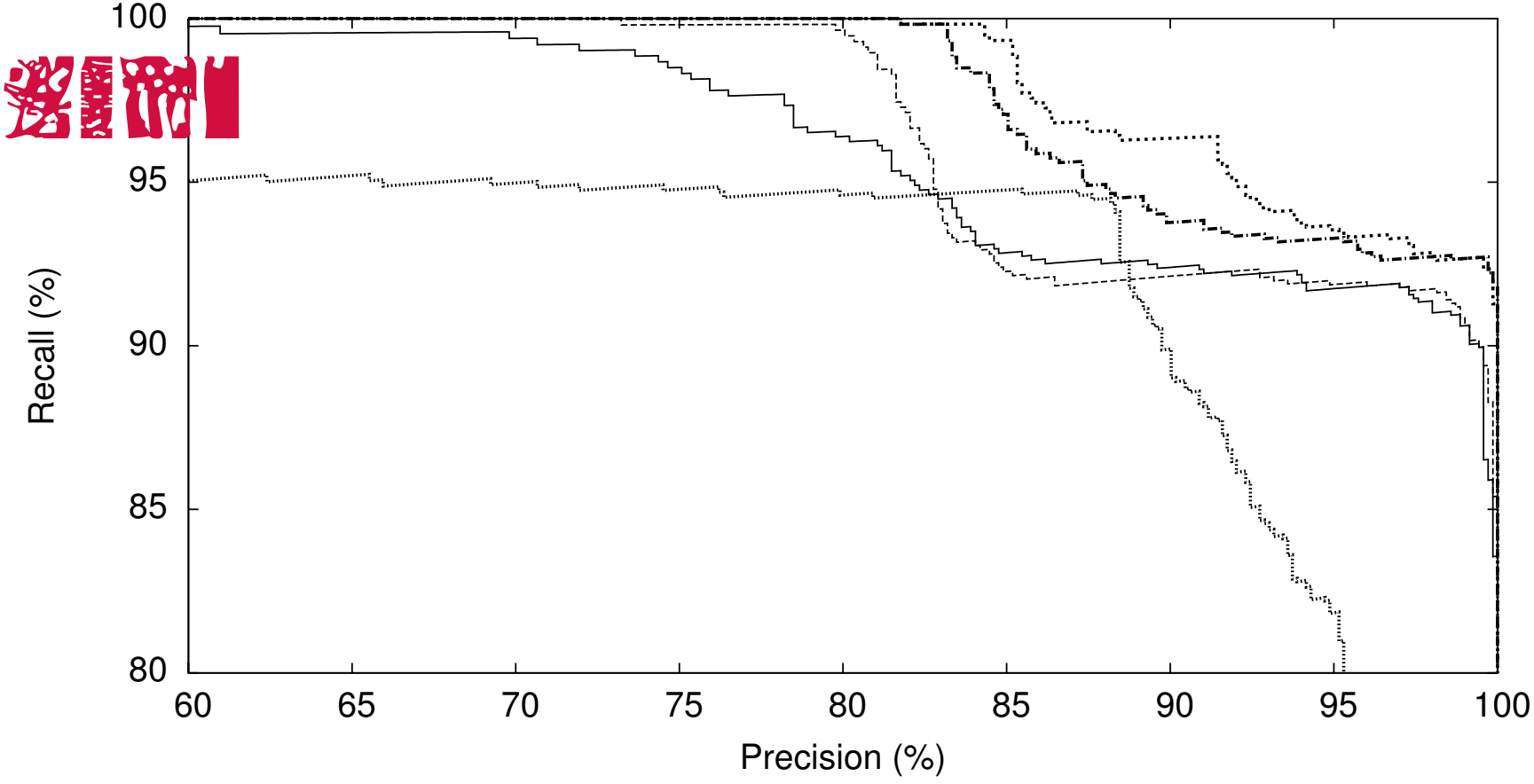
- Basic approach: index text from context of the media
  - surrounding text
  - text in or around the links pointing to the content
  - filenames
  - associated subtitles (hearing-impaired track on TV)
- Elaborate approach: index and search the media itself, with the help of speech recognition and sound, image, and video analysis. Very recent, and already some applications at Web scale, but still experimental!
  - TrackID, Shazam: identify a song played on the radio.
  - Musipedia: look for a partition by whistling a tune, `http://www.musipedia.org/`
  - Image search from a similar image, Google Images
  - Voxalead, `http://voxaleadnews.labs.exalead.com/`

Quality of search engines results evaluated with precision and recall

$$\text{precision} = \frac{\text{nb of correct results returned}}{\text{total nb of results}}$$

$$\text{recall} = \frac{\text{nb of correct results returned}}{\text{total nb of correct results}}$$

- "Correctness" usually given by human assessment
- Precision can be evaluated relatively reliably, much more difficult for recall! (Why?)
- Human judgment quite subjective! Agreement between human evaluators rarely go over 80%



- Computed with the precision-at-k, recall-at-k for the k top results
- Area under the curve: quality of a method
- Usually, interpolate to force the decreasing of the curve

What you should remember

- The inverted index model, associated tools and techniques
  - Main ideas behind Fagin's TA and NRA
  - The document vector space model

Software

- Most DBMS have text indexing capabilities (e.g., MySQL's `FULLTEXT` indexes)
- Apache Lucene, free software library for information retrieval

To go further

- A good textbook [Manning et al., 2008]. Available online, along with slides!
- A very influential paper on top-k algorithms: [Fagin et al., 2001]

Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *Proc. PODS*, Santa Barbara, USA, May 2001.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, United Kingdom, 2008. Available online at http://informationretrieval.org.

Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3): 130–137, July 1980.

US National Archives and Records Administration. The Soundex indexing system. `http://www.archives.gov/genealogy/census/soundex.html`, May 2007.