Basics of Information Retrieval

Advanced Databases

Pierre Senellart

ÉCOLE NORMALE SUPÉRIEURE | PSL

9 November 2022

Outline

Information Retrieval, Search

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.

Text Preprocessing

Initial text preprocessing steps

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

Language Identification

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

Language Identification

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)

Tokenization

Principle

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.

Punctuation may be removed and case normalized at this point

Tokenization: Example

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$_5$

Stemming

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

Stemming schemes (1/2)

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

Stemming schemes (2/2)

Lexical stemming.

- Merge lexically related terms of various parts of speech, such as *policy*, *politics*, *political* or *politician*
- For English, Porter's stemming [Porter, 1980]; stems *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!

Stemming Example

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

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$_5$

Stop Word Removal

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.

Stop Word Removal Example

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$_5$

Outline

Text Preprocessing

Inverted Index

Answering Keyword Queries

Inverted Index

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)

Inverted Index Example

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
...	

Note:

- the length of an inverted (posting) list is highly variable – scanning short lists first is an important optimization.
- *entries* are homogeneous: this gives much room for compression.

Storing positions in the index

- 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	$d_1/6$
...	

$\Rightarrow$ so far, ok for Boolean queries: find the documents that contain a set of keywords; reject the other.

Ranked search

Boolean search does not give an accurate result because it does not take account of the relevance of a document to a query.

If the search retrieves dozen or hundreds of documents, the most relevant must be shown in top position!

Weighting terms occurrences

Relevance can be computed by giving a weight to term occurrences.

- Terms occurring frequently in a given document: more relevant. The *term frequency* is the number of occurrences of a term t in a document d, divided by the total number of terms in d:

$$\mathrm{tf}(t,d) = \frac{n_{t,d}}{\sum_{t'} n_{t',d}}$$

  where $n_{t',d}$ is the number of occurrences of t' in d.

- Terms occurring rarely in the document collection as a whole: more informative. The *inverse document frequency* (idf) is obtained from the division of the total number of documents by the number of documents where t occurs, as follows:

$$\mathrm{idf}(t) = \log \frac{|D|}{\left|\left\{d' \in D \mid n_{t,d'} > 0\right\}\right|}.$$

TF-IDF Weighting

- 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;

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

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

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

TF-IDF Weighting Example

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	$d_1/6/.47$
...	

Outline

Text Preprocessing

Inverted Index

Answering Keyword Queries

Answering Boolean Queries

- 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

Exercise

Consider the following documents:

1. $d_1 =$ I like to watch the sun set with my friend.
2. $d_2 =$ The Best Places To Watch The Sunset.
3. $d_3 =$ My friend watches the sun come up.

Construct an inverted index with tf/idf weights for terms 'best' and 'sun'. What would be the ranked result of the query 'best OR sun'?

Bibliography I

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.