

Data acquisition, extraction, and storage

Inverted Index Construction

Pierre Senellart



5 December 2023

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)

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 the₁ jaguar₂ is₃ a₄ new₅ world₆ mammal₇ of₈ the₉ felidae₁₀
family₁₁
- d_2 jaguar₁ has₂ designed₃ four₄ new₅ engines₆
- d_3 for₁ jaguar₂ atari₃ was₄ keen₅ to₆ use₇ a₈ 68k₉ family₁₀ device₁₁
- d_4 the₁ jacksonville₂ jaguars₃ are₄ a₅ professional₆ us₇ football₈
team₉
- d_5 mac₁ os₂ x₃ jaguar₄ is₅ available₆ at₇ a₈ price₉ of₁₀ us₁₁ \$199₁₂
for₁₃ apple's₁₄ new₁₅ family₁₆ pack₁₇
- d_6 one₁ such₂ ruling₃ family₄ to₅ incorporate₆ the₇ jaguar₈ into₉
their₁₀ name₁₁ is₁₂ jaguar₁₃ paw₁₄
- d_7 it₁ is₂ a₃ big₄ cat₅

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 the₁ jaguar₂ be₃ a₄ new₅ world₆ mammal₇ of₈ the₉ felidae₁₀
family₁₁
- d_2 jaguar₁ have₂ design₃ four₄ new₅ engine₆
- d_3 for₁ jaguar₂ atari₃ be₄ keen₅ to₆ use₇ a₈ 68k₉ family₁₀ device₁₁
- d_4 the₁ jacksonville₂ jaguar₃ be₄ a₅ professional₆ us₇ football₈ team₉
- d_5 mac₁ os₂ x₃ jaguar₄ be₅ available₆ at₇ a₈ price₉ of₁₀ us₁₁ \$199₁₂
for₁₃ apple₁₄ new₁₅ family₁₆ pack₁₇
- d_6 one₁ such₂ rule₃ family₄ to₅ incorporate₆ the₇ jaguar₈ into₉
their₁₀ name₁₁ be₁₂ jaguar₁₃ paw₁₄
- d_7 it₁ be₂ a₃ big₄ cat₅

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 jaguar₂ new₅ world₆ mammal₇ felidae₁₀ family₁₁

d_2 jaguar₁ design₃ four₄ new₅ engine₆

d_3 jaguar₂ atari₃ keen₅ 68k₉ family₁₀ device₁₁

d_4 jacksonville₂ jaguar₃ professional₆ us₇ football₈ team₉

d_5 mac₁ os₂ x₃ jaguar₄ available₆ price₉ us₁₁ \$199₁₂ apple₁₄
new₁₅ family₁₆ pack₁₇

d_6 one₁ such₂ rule₃ family₄ incorporate₆ jaguar₈ their₁₀ name₁₁
jaguar₁₃ paw₁₄

d_7 big₄ cat₅

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.

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 :

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

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:

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

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;

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

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

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

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	$d_1/6/.47$
...	

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.