

# Full-Text Indexing and Search

Pierre Senellart

Max-Planck-Institut für Informatik & Université Paris-Sud

May 16, 2008

# Outline

- 1 Introduction
- 2 Text Preprocessing
- 3 Inverted Index
- 4 Answering Keyword Queries
- 5 Clustering
- 6 Other Media
- 7 Conclusion

# Information Retrieval, Search

## Problem

How to *index* textual content so as to answer (keyword-based) queries *effectively* and *efficiently*? Example of contexts: Web search, text fields in databases.

# Information Retrieval, Search

## Problem

How to *index* textual content so as to answer (keyword-based) queries *effectively* and *efficiently*? Example of contexts: Web search, text fields in databases.

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.

Usually, remove punctuation and normalize case at this point

# Tokenization: Example

- $d_1$  the<sub>1</sub> jaguar<sub>2</sub> is<sub>3</sub> a<sub>4</sub> new<sub>5</sub> world<sub>6</sub> mammal<sub>7</sub> of<sub>8</sub> the<sub>9</sub> felidae<sub>10</sub> family<sub>11</sub>
- $d_2$  jaguar<sub>1</sub> has<sub>2</sub> designed<sub>3</sub> four<sub>4</sub> new<sub>5</sub> engines<sub>6</sub>
- $d_3$  for<sub>1</sub> jaguar<sub>2</sub> atari<sub>3</sub> was<sub>4</sub> keen<sub>5</sub> to<sub>6</sub> use<sub>7</sub> a<sub>8</sub> 68k<sub>9</sub> family<sub>10</sub> device<sub>11</sub>
- $d_4$  the<sub>1</sub> jacksonville<sub>2</sub> jaguars<sub>3</sub> are<sub>4</sub> a<sub>5</sub> professional<sub>6</sub> us<sub>7</sub> football<sub>8</sub> team<sub>9</sub>
- $d_5$  mac<sub>1</sub> os<sub>2</sub> x<sub>3</sub> jaguar<sub>4</sub> is<sub>5</sub> available<sub>6</sub> at<sub>7</sub> a<sub>8</sub> price<sub>9</sub> of<sub>10</sub> us<sub>11</sub> \$199<sub>12</sub>  
for<sub>13</sub> apple's<sub>14</sub> new<sub>15</sub> family<sub>16</sub> pack<sub>17</sub>
- $d_6$  one<sub>1</sub> such<sub>2</sub> ruling<sub>3</sub> family<sub>4</sub> to<sub>5</sub> incorporate<sub>6</sub> the<sub>7</sub> jaguar<sub>8</sub> into<sub>9</sub>  
their<sub>10</sub> name<sub>11</sub> is<sub>12</sub> jaguar<sub>13</sub> paw<sub>14</sub>
- $d_7$  it<sub>1</sub> is<sub>2</sub> a<sub>3</sub> big<sub>4</sub> cat<sub>5</sub>

# 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.

- 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 stocking, rose

# Stemming schemes (1/2)

## Lexical stemming.

- Merge **lexically related** terms of various parts of speech, such as *policy*, *politics*, *political* or *politician*
- For English, **Porter's stemming** [Por80]; 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 despite spelling errors!
- For English, **Soundex** [US 07] stems *Robert* and *Rupert* to *R163*. Very **coarse**!

# Stemming Example

- $d_1$  the<sub>1</sub> jaguar<sub>2</sub> be<sub>3</sub> a<sub>4</sub> new<sub>5</sub> world<sub>6</sub> mammal<sub>7</sub> of<sub>8</sub> the<sub>9</sub> felidae<sub>10</sub> family<sub>11</sub>
- $d_2$  jaguar<sub>1</sub> have<sub>2</sub> design<sub>3</sub> four<sub>4</sub> new<sub>5</sub> engine<sub>6</sub>
- $d_3$  for<sub>1</sub> jaguar<sub>2</sub> atari<sub>3</sub> be<sub>4</sub> keen<sub>5</sub> to<sub>6</sub> use<sub>7</sub> a<sub>8</sub> 68k<sub>9</sub> family<sub>10</sub> device<sub>11</sub>
- $d_4$  the<sub>1</sub> jacksonville<sub>2</sub> jaguar<sub>3</sub> be<sub>4</sub> a<sub>5</sub> professional<sub>6</sub> us<sub>7</sub> football<sub>8</sub> team<sub>9</sub>
- $d_5$  mac<sub>1</sub> os<sub>2</sub> x<sub>3</sub> jaguar<sub>4</sub> be<sub>5</sub> available<sub>6</sub> at<sub>7</sub> a<sub>8</sub> price<sub>9</sub> of<sub>10</sub> us<sub>11</sub> \$199<sub>12</sub>  
for<sub>13</sub> apple<sub>14</sub> new<sub>15</sub> family<sub>16</sub> pack<sub>17</sub>
- $d_6$  one<sub>1</sub> such<sub>2</sub> rule<sub>3</sub> family<sub>4</sub> to<sub>5</sub> incorporate<sub>6</sub> the<sub>7</sub> jaguar<sub>8</sub> into<sub>9</sub>  
their<sub>10</sub> name<sub>11</sub> be<sub>12</sub> jaguar<sub>13</sub> paw<sub>14</sub>
- $d_7$  it<sub>1</sub> be<sub>2</sub> a<sub>3</sub> big<sub>4</sub> cat<sub>5</sub>

# 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<sub>2</sub> new<sub>5</sub> world<sub>6</sub> mammal<sub>7</sub> felidae<sub>10</sub> family<sub>11</sub>
- $d_2$  jaguar<sub>1</sub> design<sub>3</sub> four<sub>4</sub> new<sub>5</sub> engine<sub>6</sub>
- $d_3$  jaguar<sub>2</sub> atari<sub>3</sub> keen<sub>5</sub> 68k<sub>9</sub> family<sub>10</sub> device<sub>11</sub>
- $d_4$  jacksonville<sub>2</sub> jaguar<sub>3</sub> professional<sub>6</sub> us<sub>7</sub> football<sub>8</sub> team<sub>9</sub>
- $d_5$  mac<sub>1</sub> os<sub>2</sub> x<sub>3</sub> jaguar<sub>4</sub> available<sub>6</sub> price<sub>9</sub> us<sub>11</sub> \$199<sub>12</sub> apple<sub>14</sub>  
new<sub>15</sub> family<sub>16</sub> pack<sub>17</sub>
- $d_6$  one<sub>1</sub> such<sub>2</sub> rule<sub>3</sub> family<sub>4</sub> incorporate<sub>6</sub> jaguar<sub>8</sub> their<sub>10</sub> name<sub>11</sub>  
jaguar<sub>13</sub> paw<sub>14</sub>
- $d_7$  big<sub>4</sub> cat<sub>5</sub>

# 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	$d_1$
...	

## 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	$d_1/6$
...	

# 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 (other weighting functions exist!);

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

$n_{t,d}$  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	$d_1/6/.47$
...	

## 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

# Answering Top- $k$ Queries

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

## 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)

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

# Fagin's Threshold Algorithm [FLN01]

- 1 Let  $R$  be the empty list, and  $m = +\infty$ .
- 2 For each  $1 \leq i \leq n$ :
  - 1 Retrieve the document  $d^{(i)}$  containing term  $t_i$  that has the **next largest**  $s(t_i, d^{(i)})$ .
  - 2 Compute its global score  $g_{d^{(i)}} = g(s(t_1, d^{(i)}), \dots, s(t_n, d^{(i)}))$  by retrieving all  $s(t_j, d^{(i)})$  with  $j \neq i$ .
  - 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  $R$ .
- 3 Let  $m = g(s(t_1, d^{(1)}), s(t_2, d^{(2)}), \dots, s(t_n, d^{(n)}))$ .
- 4 If  $R$  contains more than  $k$  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.

**Assumption:** additional index for efficient computation of  $s(t_i, d_j)$  given  $t_i$  and  $d_j$ .

# Threshold Algorithm Example

## Exercise

- 8 documents
- Query  $t_1$  AND  $t_2$
- Scores with respect to  $t_1$ :  $d_1/.8$ ,  $d_6/.7$ ,  $d_8/.6$ ,  $d_3/.5$ ,  $d_4/.1$ ,  $d_2/.1$ ,  $d_7/.05$ ,  $d_9/.05$
- Scores with respect to  $t_2$ :  $d_2/.8$ ,  $d_7/.8$ ,  $d_9/.7$ ,  $d_4/.6$ ,  $d_3/.5$ ,  $d_8/.1$ ,  $d_1/.1$ ,  $d_6/.05$
- Aggregation function: addition
- Top-3?

## Clustering Example

web [news](#) [images](#) [wikipedia](#) [blogs](#) [jobs](#) [more »](#)

jaguar

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

# Cosine Similarity of Documents

- **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  $\text{tfidf}(t, d)$ )

- 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)

# Agglomerative Clustering of Documents

- 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.*

# Indexing HTML

- 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.




# Indexing Multimedia Content

- 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**
  - ▶ Musipedia: look for a partition by whistling a tune
  - ▶ Image search from a similar image

# What you should remember

- The **inverted index** model for efficient answers of keyword-based queries.
- The **threshold algorithm** for retrieving top- $k$  results.

# Bibliography

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