Ingo Frommholz

Automatic Categorization of Web Documents

January 16, 2001

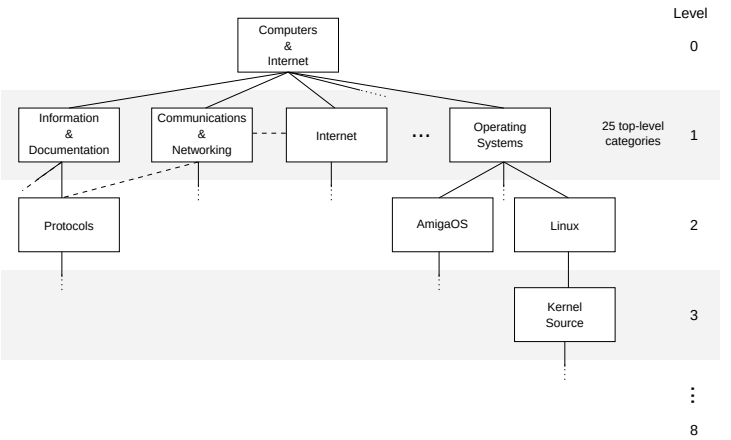
University of Dortmund
Computer Science Department

Table of contents

1 Introduction

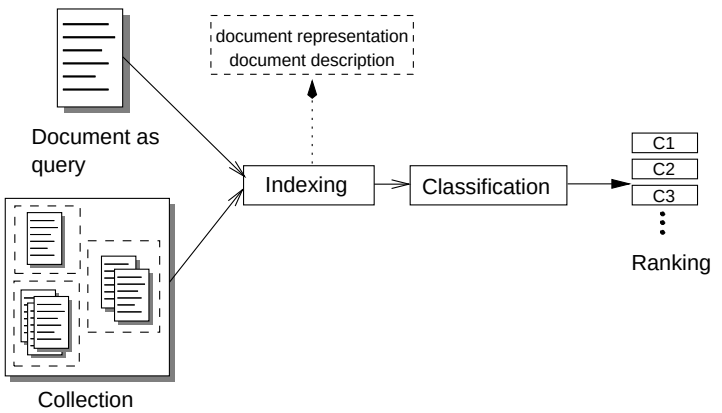
- The internet, especially the World Wide Web and the Usenet, offer a lot of information which can be accessed by the user with search engines like *Yahoo!*, AltaVista, etc.
- The search for a document can be performed with key words or by browsing through a catalogue (like e.g. *Yahoo!*) where documents are categorized into categories.
- *Categorization* of web documents (e.g. HTML documents) describes the task to find relevant categories for a new document which shall be inserted into such a web catalogue so that this document will be assigned to the categories it belongs to.

- Mainly, this categorization is done manually. But the growing number of documents which need to be categorized yields the question if and how this task can be done automatically.
- *Yahoo! Computers & Internet*:



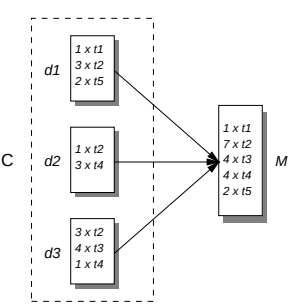
2 Automatic categorization of documents

- Goal: To sort documents into one or more appropriate categories.
- Since there is a closeness of the categorization to information retrieval, classical IR methods can be used for the categorization task.



3 The megadocument approach

- Basic idea: Documents of a category are merged into one big document, the so-called *megadocument*.



- The document to categorize is seen as a query to the collection of megadocuments. The top-ranked megadocuments of the resulting ranking indicate the categories the document might belong to.

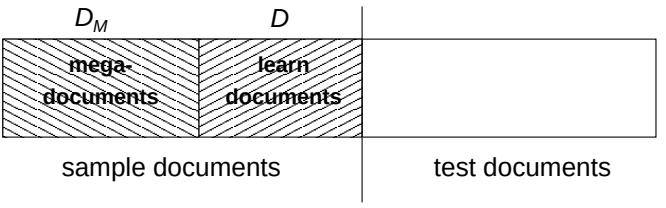
4 Indexing of megadocuments

- One possible approach: $tf \times idf$.
- Another approach: Probabilistic, description-oriented indexing.
- Description of megadocuments as a vector of term weights.
- Extract *feature vector* $\vec{x}(t, M)$ */relevance description* for every term-megadocument pair in the megadocument set.
- Example:

$$x_1(t, d) = \begin{cases} 1 & \text{if } t \text{ appears emphasized in } M \\ 0 & \text{otherwise} \end{cases}$$

- Good way to take the specifics of a markup-document into account.

- Definition: A document d is *relevant* (R) to a megadocument M if d belongs to the category C represented by M, and *not-relevant* ($\overline{R}$) otherwise. $r(d, d') \in \{R, \overline{R}\}$ is a relevance judgement.
- Term weighting by estimating the probability $P(R|\vec{x})$ that there is relevance if we have the feature vector $\vec{x}$.
- Event space: (megadocument, document, term).
- Estimation is done using a learning sample. To achieve this and to build the megadocuments, we have to split the collection:



- Example:

M	d	$t \in M^T \cap d^T$	$\vec{x}(t, M)$	$\vec{x}(t, d)$	r
M_1	d_2	t_2	$(1, 0)$	$(1, 1)$	$\overline{R}$
M_1	d_1	t_3	$(1, 0)$	$(1, 0)$	$\overline{R}$
M_1	d_3	t_1	$(2, 0)$	$(2, 0)$	R
M_2	d_1	t_1	$(1, 0)$	$(1, 1)$	R
M_2	d_3	t_4	$(2, 0)$	$(2, 0)$	$\overline{R}$
M_2	d_2	t_5	$(2, 0)$	$(1, 0)$	$\overline{R}$

- From this, we could build one common sample (ONE) to index both query documents and megadocuments, or two samples, one for indexing the megadocuments (MD) and one for indexing the query documents (QD).

	$P(R\|\vec{x})$		
$\vec{x}$	ONE	MD	QD
$(1, 0)$	$1/5$	$1/3$	0
$(1, 1)$	$1/2$	0	$1/2$
$(2, 0)$	$2/5$	$1/3$	$1/2$

- Regression is used for getting an *indexing function* e:
  - Linear regression: e is a polynomial (*least square polynomial*)

$$P(R|\vec{x}) \approx e_{LSP}(\vec{x}) = \vec{a}^T \cdot (1, \vec{x}).$$

  - Logistic Regression:

$$P(R|\vec{x}) \approx e_{LOG}(\vec{x}) = \frac{e^{\vec{a}^T \cdot (1,\vec{x})}}{1 + e^{\vec{a}^T \cdot (1,\vec{x})}}$$

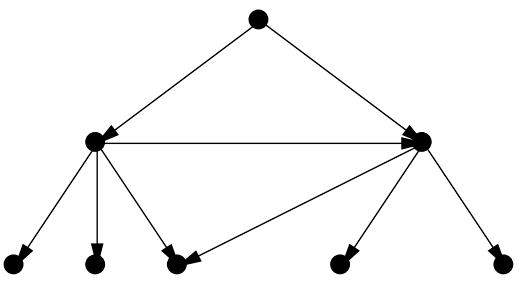
  - Logistic function only yields values between 0 and 1 (in contrast to the linear function).

  Regression would now calculate coefficient vector $\vec{a}$ according to a specific regression criterion (minimum squared errors or maximum likelihood).

- Term weight $w_{t,d} = e(\vec{x}(t, d))$.

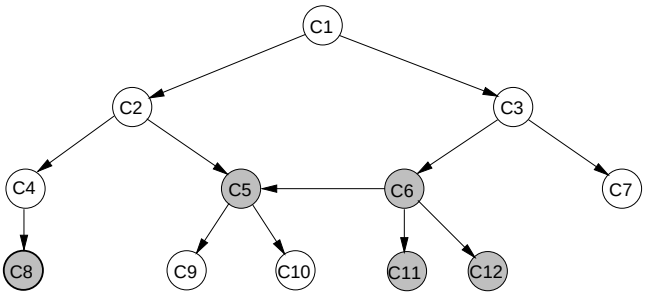
5 Using hierarchy for categorization

- Many approaches don't consider the hierarchical structure of the category scheme.
- *Hierarchy graph*:



- Motivation: Can there be a better classification decision if the knowledge about the hierarchy is considered?

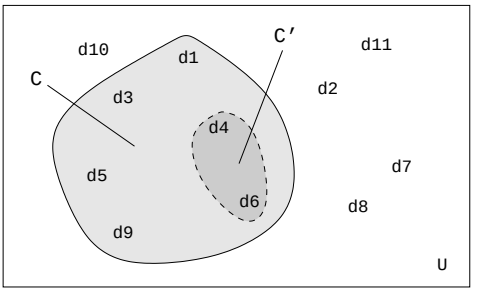
- Example: Ranking of categories according to a query document:



  $\rightarrow$ Category in the area around C_6 might be more adequate for our document.

- Basic idea: If there are categories with higher weights around a category node, this is a positive statement about the relevance of this category. Vice versa for categories with lower weights around the category node.

- Probabilistic interpretation of the hierarchy: Probability of implication $P(C \rightarrow C')$, based on users decision.



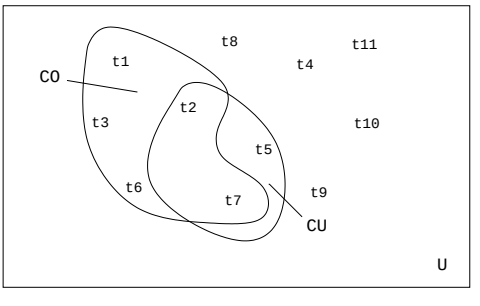
  van Rijsbergen: $P(C \rightarrow C') = P(C'|C) = \frac{|C \cap C'|}{|C|}$

- In most cases, it is impossible to get this probability for all possible pairs (C, C').

- Estimation of $P(C \rightarrow C')$ using *local* and *global* implication probability.

- *Local implication probability* $P(C \dot{\rightarrow} C')$: Probability that C implies C', whereat C and C' must be neighbours in the hierarchy graph.
- Estimation of $P(C \dot{\rightarrow} C')$ is possible and can be done in two ways:
  1. Intellectual estimation: For example with global values for $P(C_O \dot{\rightarrow} C_U)$ and $P(C_U \dot{\rightarrow} C_O)$ (C_O is direct supercategory (i.e. father in the hierarchy graph) of C_U), so that $P(C_O \dot{\rightarrow} C_U) < P(C_U \dot{\rightarrow} C_O)$. The fan-out and fan-in of a category node might be considered as well.

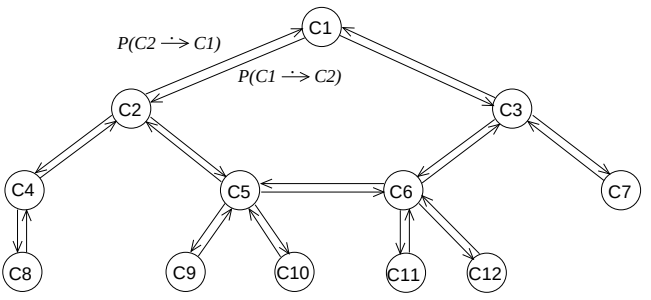
  2. Calculation on term basis: Motivated by the megadocument approach, categories can be seen as (big) documents.



$$P(C_O \rightarrow C_U) = P(C_U|C_O) = \frac{|C_O \cap C_U|}{|C_O|}$$
$$P(C_U \rightarrow C_O) = P(C_O|C_U) = \frac{|C_U \cap C_O|}{|C_U|}$$

(–) $P(C_O \dot{\rightarrow} C_U) < P(C_U \dot{\rightarrow} C_O)$ not guaranteed.

- *Probabilistic hierarchy graph* built from the hierarchy graph using the local implication probabilities as edge weights.



- Every local implication is now seen as an event.

- *Global implication probability* $P(C \stackrel{*}{\rightarrow} C')$.
- $P(C \rightarrow C') \approx P(C \stackrel{*}{\rightarrow} C')$.
- Calculation of $P(C \stackrel{*}{\rightarrow} C')$ with the sieve formula using the local implication probabilitites and the hierarchy graph. The calculation can be done with probabilistic Datalog:

```
0.3 localimply_o(c1,c2).          0.5 localimply_u(c2,c1).
0.3 localimply_o(c2,c5).          0.5 localimply_u(c5,c2).
0.3 localimply_o(c6,c5).          0.5 localimply_u(c5,c6).
0.3 localimply_o(c1,c3).          0.5 localimply_u(c3,c1).
0.3 localimply_o(c3,c6).          0.5 localimply_u(c6,c3).
...

globalimply(C1,C2) :- localimply_o(C1,C2).
globalimply(C1,C2) :- localimply_u(C1,C2).
globalimply(C1,C2) :- globalimply(C1,C) & localimply_o(C,C2).
globalimply(C1,C2) :- globalimply(C1,C) & localimply_u(C,C2).
```

- Using $P(C \rightarrow C')$ for estimating the hierarchical retrieval status value (RSV) (C is the set of categories):

$$P_H(d \rightarrow C) = \sum_{C' \in \mathcal{C}} P(d \rightarrow C') \cdot P(C') \cdot P(C' \rightarrow C).$$

- $P(d \rightarrow C)$ is the RSV calculated by the non-hierarchical classifier.
- $P(C')$ is a normalization factor and can be set to $1/|\mathcal{C}|$.

6 Experiments

- Exact match and Top25 categories.
- Selection of the Top25 categories from the ranking:
  1. Take the Top25 document of the first ranked categories.
  2. kNN variant (ϱ is the retrieval function, d^D and C^D are document descriptions, C^{25} is a Top25 category):

$$\varrho_{25}(d^D, {C^{25}}^D) = \sum_{C \in NN} \frac{\varrho(d^D, C^D) \cdot P(C \rightarrow C^{25})}{|NN|}$$

- Two kinds of experiments: megadocuments with features and megadocuments using hierarchy.

6.1 Megadocuments with features

- The feature vector $\vec{x}(t, d)$ consisted of ten features:
  - $x_1(t, d) = 1$ if t is the most frequent term, 0 else,
  - $x_2(t, d)$ is the number of terms in the document,
  - $x_3(t, d)$ is the number of distinct terms in the document,
  - $x_4(t, d)$ is the term frequency of t in the document,
  - $x_5(t, d) = 1$ if t appears in the title of d, 0 else,
  - $x_6(t, d) = 1$ if t appears emphasized in d, 0 else,
  - $x_7(t, d) = 1$ if t appears in the heading of the document, 0 else,
  - $x_8(t, d) = 1$ if t appears in the first paragraph d, 0 else,
  - $x_9(t, d) = 1$ if t appears in the document d itself, 0 if t appears in a document referenced by d (*radius1 document*),
  - $x_{10}(t, d)$ is the inverse document frequency of t with respect to the megadocuments.

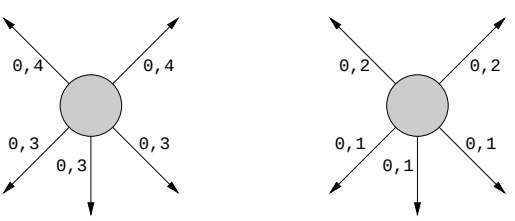
- Baseline (TFDIF): megadocuments indexed by $tf \times idf$; considered the top 50 terms by idf for every query. So query documents are described as a list of their terms sorted by idf.
- Experiments (the digit indicates the number of indexing functions used):
  - Linear regression with least square polynomials: LSP1 and LSP2.
  - Logistic regression with maximum likelihood criterion: LOGLLH1 and LOGLLH2.
  - Logistic regression with minimum squared errors criterion: LOGLSP1 and LOGLSP2.
- Considered the top 50 terms by weight for every query.
- Radius1 strategy.

Experiment	TR	∅	T25 TR	T25 ∅	T25k TR	T25k ∅
TFIDF	11,21%	14,87%	50,58%	21,36%	49,44%	59,39%
LSP1	8,04%	11,44%	43,30%	20,94%	45,33%	56,14%
LSP2	7,56%	10,9%	41,57%	20,88%	44,45%	55,39%
LOGLLH1	8,30%	12,26%	40,31%	20,77%	45,01%	55,54%
LOGLLH2	8,09%	12,18%	40,11%	20,77%	44,83%	55,47%
LOGLSP1	10,32%	14,27%	45,01%	20,86%	**50,84%**	**59,97%**
LOGLSP2	10,11%	14,02%	44,78%	20,84%	**50,41%**	**59,62%**

- Probabilistic, description-oriented indexing can increase effectivity.
- Logistic regression with minimum squared errors performed best (apart from the baseline).
- One indexing function is enough.
- Surprisingly, most regression indexing methods didn't perform better than the baseline.

6.2 Megadocuments using hierarchy

- *Adaptation*:



without adaptation with adaptation

- Baseline (OH): Experiment performed by Claus-Peter Klas (using the top 10 terms in the query). The results of this experiment were the basis of the post-processing step.

- Intellectual estimation: First two digits describe global $P(C_O \dot{\rightarrow} C_U)$, last two digits describe global $P(C_U \dot{\rightarrow} C_O)$, a means adaptation:
  `0102`, `0102a`, `0208`, `0208a`, `0708` and `0708a`.
- Calculation of $P(C \dot{\rightarrow} C')$ on a term basis using the intersection of the top 50 terms by idf of each megadocument: `IDF50` and `IDF50a`.

Experiment	TR	∅	T25 TR	T25 ∅	T25k TR	T25k ∅
OH	14,45%	18,13%	52,74%	18,93%	54,12%	62,73%
IDF50	**14,47%**	**18,2%**	52,12%	**18,97%**	**54,36%**	**62,92%**
IDF50a	**14,54%**	**18,24%**	52,14%	**18,95%**	**54,34%**	**62,9%**
0102	13,58%	17,41%	51,83%	**19,04%**	**55,03%**	**63,19%**
0102a	14,45%	18,13%	52,74%	18,93%	54,12%	62,72%
0208	8,23%	11,31%	39,3%	18,54%	53,15%	60,34%
0208a	9,74%	13,58%	49,51%	18,68%	53,21%	61,4%
0708	6,75%	9,45%	34,22%	18,46%	49,39%	57,22%
0708a	9,71%	13,57%	48,54%	18,79%	53,23%	61,22%

- Using the hierarchy information can increase effectivity.
- $P(C \dot{\rightarrow} C')$ should be chosen carefully.
- Calculation of $P(C \dot{\rightarrow} C')$ considers the local conditions between two neighboured categories. This seems to be the right way for achieving the local implication probabilities.