Abstract

Text Mining is an active area of research and development, which combines and expands techniques found in related areas like information retrieval, computational linguistics, and data mining to perform an analysis of large corpora of digital documents. This paper describes the TaxGen Text Mining project carried out at the IBM Software Development Lab. at Boeblingen, Germany. The goal of TaxGen was the automatic generation of a taxonomy for a collection of previously unstructured documents, namely a set of 73,000 news wire documents spanning one year.

1. Introduction

Data Mining is a strong growing area of research and development, that has seen new algorithms and new applications appearing year by year [2]. Naturally, the most prominent application sectors like banking, insurance, and marketing do provide rich data sets with numerical attributes. The question of whether two records should be considered similar or not can be reduced to a straightforward statistical comparison of feature ranges.

The discipline of Text Mining merges methods of information retrieval, computational linguistics and data mining to achieve means for an automatic analysis of large and unstructured corpora of digital documents. Although this seems to be pretty close to Data Mining-like engineering for large sets of raw transactional records, there are a few but important differences. First, the question how one defines the similarity of two given documents always depends on the intended application and usage within a Text-Mining project. Plain word-by-word comparison, statistical phrase detection or sophisticated names extraction methods may judge the same two documents to be similar or not - the acceptability of this decision depends on, for example, whether one must provide a coarse overview of a company's patent text collection, versus the task of detecting threads of important persons in a news wire text stream. In the first case, the semantic notion of similarity can be reduced to standard text processing techniques (stemming, dictionary building and lookup), whereas the second case requires heuristics and parsing techniques as they are developed in the area of natural language processing (NLP). Fact extraction from texts, and the harvesting of crisp and vague information, as it is done in information discovering projects (e.g. see [1] for a recent research prototype), however, require sophisticated knowledge models, which tend to become domain specific.

Second, Text Mining must deal with a much higher dimensionality of features per entity. For a real-world text database the number of potential features ranges up to tens or hundreds of thousands. This has a major impact not only on the design of the mining algorithms used, but also on the subsequent visualization modules. Well known interactive graphical systems like VIBE [9] are designed to visualize the inter-document relationship of all documents returned by a retrieval system (i.e., with a common topic), and to relate them to the user-defined query, which is typically less than five words long.

A cluster tree that has to be constructed automatically has no query terms, but must relate all documents with all terms and provide means to visualize them. This task is not covered by standard information retrieval graphical user interface (GUI) design, nor does it match typical graph layout algorithms. Further, incremental and interactive browse-and-recluster designs like Scatter/Gather [6] will not scale up to giga-bytes of data. As we have shown in Agosti et.al. [5], navigation and context are appropriate means to visualize large and unknown hyper spaces like an intranet.
We will show how this principle can be applied to the output of clustering of many documents later in this paper.

In this paper we will present the design we developed during the Taxonomy Generation (TaxGen) project. In the context of this project a taxonomy represents the thematic structure inherent in a collection of documents. The goal of the TaxGen project is to create taxonomies automatically by comparing documents on the basis of certain linguistic elements they contain. Depending on the criteria under which these linguistic elements are selected the resulting taxonomies will provide different views on the underlying structure.

We introduce our set of quality criteria, on the basis of which all generated taxonomies were judged. The process of developing a taxonomy was split into three threads. Number one is based upon shallow linguistic preprocessing and number two applies to more detailed phrase-based text processing techniques. During these two threads we expected to see a low (for number one) and a high (for number two) dimensional representation of documents. Thread number three simply used the keywords already provided in the collection and served mainly as a baseline, which showed a low number of dimensions. This is due to the fact that the number of keywords provided was much lower (on average, about twenty terms per document) than the full text portion of the documents.

In the remainder of this paper, we investigate the appropriateness of using the Hierarchical Agglomerative Clustering algorithm (which is part of the IBM Intelligent Miner for Text product suite) to generate taxonomies and we demonstrate the interaction of the Clustering tool with the Topic Categorization tool to route new and unseen documents into the correct cluster automatically. In the two major sections of this paper we present the evaluation of the two resulting taxonomies and the browser front end we developed. The GUI is a HTML-based taxonomy browser, which works top-down from the root of the hierarchy.

1. The design of a taxonomy generator and classifier

In Winter 1997-98, our team at the IBM Software Development Lab. at Boeblingen, Germany undertook a field experiment to demonstrate the feasibility of the Text Mining algorithms that we recently formed into a problem solving suite for digital documents. Our intent was to generate a taxonomy for an independently obtained collection of digital documents. In our case, it consists of more than 70,000 documents covering one year of a news wire stream with medium sized documents. A typical document has between hundreds and a thousand lines of English text. Each document is annotated with a list of keywords. They have been assigned manually to ease the grouping of related news. Keywords also help a lot in pre-selecting documents of interest for a human reader.

SUBJECT: Access -printer software releases
DATE: February 20, 1997
KEYWORDS: communications devices,
telephones, wireless datacom, ad terms, handheld
computers, bundling, test & measurement, PC
manufacturers, computer maintenance, servers,
computers, software, general computer terms,
business, improvements, breakthroughs, prices,
enterprise systems management, print servers,
business alliances, embedded systems, ... 
WORDS: 270
TEXT: IBM Corp. recently unveiled printer
management software based on Java. The Network
Printer Manager is designed to enable PC users to
test the functions of printers in an Intranet
through a Web browser. Initially available for the
Windows NT environment, the IBM-only software
will support non-IBM printers before the end of the
year, according to IBM officials. Ericsson will use
Microware's OS-9 operating system in future
products and services for the mobile market.
AverMedia unveiled the TVGenie, a TV tuner
device that enables most computer monitors to
display television programs. The TVGenie plugs
directly into the back of the computer display and
does not require the use of a computer.

Subset of data taken from a news article

Our goal was to provide a design to automatically
generate a taxonomy, which must fulfill the same purpose as
the keywords and which can be used to detect new and
unforeseen intra-document relationships. As a byproduct we
wanted to investigate the two major problems of Text
Mining in a real-world environment: what are sufficient
criteria to consider two documents similar and how do we
reduce the dimensionality of a full-text based domain.
All the taxonomies generated during the project were judged on the following criteria:

- **Navigational balance**: the taxonomy must be well-balanced for navigation by an end-user. In particular, the fan-out at each level of the hierarchy must be limited (goal: less than 20), the depth must be limited (maximum 6 levels), and there must not be empty nodes. Given this specification, a standard tree browser can be used to browse the taxonomy, since it will fit on a typical screen.

- **Orientation**: nodes in the taxonomy should reflect "concepts" and be adequately labeled so that the labels of the nodes give sufficient orientation for a user traversing the taxonomy.

- **Coherence and selectivity**: the leaf nodes in the taxonomy should be maximally coherent with all assigned documents having the same thematic content. Related documents from different nodes should appear within a short distance in the taxonomy structure. For each non-leaf node, the selectivity of the computed taxonomy should be reasonably high, i.e., one must be able to anticipate what will be the difference(s) between branches originating at this node.

- **Domain-independence of all applied technologies**: we will not use any hand-coded knowledge derived from an analysis of the given document collection to steer the taxonomy generation process.

Considering this, we addressed the following objectives for the TaxGen project:

- **Integrate linguistic methods** to increase the conceptual quality of the taxonomy. We define conceptual quality as:
  - coherence and selectivity of document clusters
  - significance of extracted terms and phrases
  - address the ambiguity problems of natural language

Thus, we split the project into three threads. Within the two development threads, we exploited two techniques implemented in the Intelligent Miner for Text product suite, namely the lexical affinities (LA) preprocessing layer and the Feature Extraction module. Both techniques will be compared against will be compared against manually classified and labeled texts, i.e., the keywords assigned to the document collection by the editors.

- **Scalability**: any document of the collection must be assigned to some leaf node in the taxonomy and the whole process must be applicable to significantly larger databases than the sample provided. Hence, we decided to establish a separate routing task to guarantee scalability for arbitrary collection sizes and processing of forthcoming documents. The routing task is to train a given taxonomy and afterwards sort all documents into the appropriate places - both during the creation phase (i.e. the TaxGen project) and for later use (adding new documents to the set of documents which are already organized into a taxonomy). This requires experimentation and measuring based upon test sets to identify the optimal training conditions.

- **Extensibility**: we anticipated that automatically generated taxonomies might not match 'common sense' expectations of what is a familiar, well-known index for a large corpus with a high diversity of topics. Hence, our solution must show well-defined interfaces to add domain-oriented extra knowledge (like a thesaurus) to improve the navigational ease of use of the taxonomy tree.

- **Provide a user interface to ease exploration and evaluation**: once a taxonomy is generated, the user of the system must be able to navigate using contextual browsing information. There is a semantic notion of zooming-into a taxonomy, which we implemented as the process of top-down browsing from high-level concepts to individual documents (news article).
Figure 1 gives an overview of the process architecture used to perform the taxonomy generation. The generation is based on a 10% sample taken from the large document database (73,000 documents). Our first inspection of the collection showed that we needed to normalize the sample by the size of the documents and to select a well-balanced distribution of all news articles which span several months of worldwide news in the timeline. At this point we could affirm our a priori decision to use an order-independent cluster algorithm (a Hierarchical Agglomerative Cluster algorithm - for an overview on clustering see [4]), since we could observe news topics come and go over time. Members of the sample were then selected based on a constant interval over the complete period of time which was spanned by the news collection.

The sample is then piped into the corresponding Feature Extraction tool, which produces a concise representation of each document, and sends it to the Hierarchical Clustering module. When clustering is finished the data and the processing forks into label generation and training/routing.

The label generation works on the same subset of the document database to force consistency between the taxonomy structure and the assigned labels. The training of the categorizer is performed on the same subset for the same reasons.

Afterwards, a final batch process merges the forked data and processing modules: the routing module takes each unprocessed document and assigns it to its appropriate cluster. The clusters are presented to the user together with the precomputed labels.

2. Preprocessing digital documents

As was mentioned above, we applied two preprocessing functions of the Intelligent Miner for Text suite to perform a mapping from raw text into countable and comparable discrete variable values.

Lexical affinities (LAs) are a statistical means to identify and capture the frequent co-occurrence of two words. Typically words co-occurring in phrases like ‘service provider’ or ‘computer software’ will be identified as a lexical affinity. But a LA are more than well formed phrase. Words that frequently co-occur within a text window of a few words length have shown to a have a much higher semantic meaning compared to single terms like ‘service’,

![Image of TaxGen architecture](image-url)
meaningful cluster descriptors. In this case we ended up
decreases the numbers of clusters at the expense of
(i.e., all (single) words that are not part of the stop word list)
balance of the complete taxonomy. Using a loose measure
very coherent clusters, but decreases the navigational
(like e.g. using just person and company names) leads to
companies and their products. Using a very sharp measure
business database tends to build the taxonomy around
and multi-word terms as input features for clustering in a
loose overall quality. For example, selecting only names
taxonomy towards a well balanced view of the data without
parameters can be changed in the process to tune the
statistical or content-oriented filtering of term output. This
the selection of terms for the clustering step can be based on
representation of documents and clusters. To overcome this,
with relatively meaningless terms like „work”, „system”,
„information” etc.

For the final version, we decided to use names, terms and
general words, but to apply filtering to remove high-frequency terms and very low-frequency terms. Ignoring high-frequency terms basically is close to automatically extending a stop word list. Filtering low-frequency terms eliminates noisy words from the
taxonomy.

3. Hierarchical clustering for news-wire
data

Besides its well-known use in typical Data Mining
scenarios, clustering has a long tradition in Information
Retrieval as well. One origin is the cluster hypothesis as it
was formulated by van Rijsbergen in 1979 [11] (here,
comparing keyword based and document based clustering,
e.g. the cluster-based search by Salton [12]). Based upon his
SMART retrieval system, Salton sorted incoming
documents into several clusters, re - partitioning the cluster
structure if appropriate. Unseen documents are routed with
respect to the cluster representatives only, because it was
computationally too expensive to compare each unseen
document with all training data. Hence, the global design
showed several weaknesses like the dependency of the
results on the ordering and the number of the incoming
documents. Meanwhile, computation power is much
cheaper, but the computational complexity of always
‘comparing-each-by-each’ is still prohibitive in the general
case.

The hierarchical clustering algorithm (HCA) we used in
the TaxGen project is a bottom-up algorithm as described in
[8]. Clustering starts with the individual documents and
builds the bottom clusters first. It then works its way upward
in the hierarchy forming higher-level clusters by grouping
together related clusters. A good description of different
agglomerative clustering algorithms can be found in Willet
[4]. Alternative clustering designs, which have been
optimized to combine clustering and routing tasks, can be
found e.g. in Iwayama et.al. [3].

The advantage of our HCA approach is that it achieves
very good coherence at the lower levels of the resulting
taxonomy. Usually, the first (i.e. the top) level of the
generated hierarchy is too detailed (with the exception of
using the supplied keywords as input) so this remains an
issue for application programming when building a real
solution based on this technology. Scalability is achieved in
this approach by building the taxonomy only on a subset of
the data and then routing the remaining documents into their respective categories by a separate routing step.

Like most agglomerative cluster algorithms, we internally build up a complete cluster hierarchy (a dendogram). This hierarchy relates each pair of documents and annotates this relationship with the corresponding degree of similarity. Slicing a dendogram into layers with certain ranges of similarity reveals high-level concepts (with a lower intra-cluster similarity) and more concrete topics (closer to the leafs of the dendogram, with high similarities). Since we assigned the routing task to a subsequent processing module, we could identify the optimal range specification to ease the process of browsing the final taxonomy, which essentially is a dendogram sliced into (a maximum of) six levels.

Our experience shows that working on a subset does not impact quality. Cluster runs with different subset sizes and otherwise unchanged parameters showed that the taxonomy becomes ‘stable’ at about 5000 documents, i.e., the resulting structure did not change significantly when we increased the sample size. The point of saturation changes slightly depending on which input preprocessing technique we applied. For input data having a lower dimensionality of features (supplied keywords, LA-based preprocessing), we achieved no further significant changes above ~4000 documents (if sampled properly). The higher dimensional input of thread two forced us to increase the size of the sample by some extra 50% of data (i.e. documents).

The quality we could achieve for the routing task with this size of training data is very high (see the next section on the accuracy of the router). Clustering of a 5000 document sample takes about 1.5 hours on a RS/6000 43P, running under AIX 4.2.1, with 248MB of main memory. This relatively short elapsed time for taxonomy generation allows for short turnaround cycles while experimenting with various process setups.

4. Routing new documents: The Topic Categorizer

Since the purpose of the generated taxonomy was not only to organize the given document collection, but also to be able to classify new incoming documents, we decided to use a dedicated tool for this task, the Topic Categorization tool of the IM for Text suite.

The Topic Categorization tool in general assigns documents to predefined categories, which in our case are the leaf nodes of the taxonomy derived by the hierarchical clustering. In a training phase the tool extracts relevant vocabulary statistics from sample documents for each category and stores these in a dictionary, called a category scheme. This training uses the same feature extraction tool as mentioned above, and the vocabulary items actually are the extracted features. The trained scheme can be thought of as a set of feature vectors, one per category, in the multidimensional vector space where each feature corresponds to a dimension. Each vector encodes frequency information for each feature.

For the actual categorization of an incoming document, called a query document, the tool computes its feature vector using the Feature Extraction tool in the most efficient mode. It then returns a ranked list of categories, sorted according to the similarity between the query document vector and the respective category vector.

The accuracy of this tool is comparable to other leading-edge technology for this task. For a comparison of statistical evaluation measures, we would like to refer to [10]. We measured the micro-averaged 11-point average precision to be 0.88 on the Reuters-21578 collection (with Apte split), that is commonly used for evaluating text categorizers. However, we would like to point out that with this collection there are several problems (special jargon and abbreviations, ad-hoc categories, etc.) that make its suitability as a categorization benchmark highly questionable.

Training the categorizer

For both versions of the taxonomy, based upon full linguistic preprocessing and upon lexical affinities, a category scheme was trained as follows. The categories were taken to be the leaves of the taxonomy with the documents assigned to them as training material.

<table>
<thead>
<tr>
<th>Taxonomy</th>
<th>Full linguistic preprocessing</th>
<th>statistical phrase - analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td># Categories</td>
<td>686</td>
<td>1.052</td>
</tr>
<tr>
<td># Training documents</td>
<td>3.729</td>
<td>6.534</td>
</tr>
<tr>
<td>Training set size</td>
<td>18.8MB</td>
<td>31.1MB</td>
</tr>
<tr>
<td>Scheme file size</td>
<td>35MB</td>
<td>41MB</td>
</tr>
<tr>
<td>Training time</td>
<td>20min</td>
<td>40min</td>
</tr>
</tbody>
</table>

The result table shows details about the category schemes resulting from the two different approaches to linguistic
preprocessing. All times were measured on Pentium-II, 266MHz, with 248MB of main memory.

**Categorizing**

The routing task was performed on all of the 73031 documents using both of the category schemes. The complete routing into one taxonomy took around 100 minutes, i.e., the routing speed is above 200MB/h. Each document was simply assigned to the taxonomy category that received the highest rank value from the Topic Categorizer.

Due to the sparseness of the training data for the categorizer, a formal evaluation of its accuracy by means of held-out examples was not meaningful.

On the other hand, by analyzing the ability of the tool to adapt to the partitioning that the cluster engine computed we could draw two conclusions. Firstly, we could verify that documents inside a cluster were actually more similar to each other (according to the categorizer's similarity measure) than to other documents. Running the classifier on the training set again returned the correct category in 99.95% of the cases for the LF thread. In other words, the cluster engine did what it is supposed to do. Secondly, the difference in the similarity measures between LF and LA does not seem to have a big impact on the coherence of the clusters. Running the LA-trained classifier (which always uses an LF-based similarity measure) on its training set again revealed the correct category in 99.6% of the cases. I.e., document clusters formed under the LA-based measure look very coherent also under the LF-based one.

### 5. Evaluation methods and results

By applying lexical-affinity extraction as a preprocessor, we achieved a taxonomy well suited for navigation, using up to six levels if required to visualize different aspects of concepts contained in the clusters. It shows an average branching factor of 3 with no more than 12 branches in the inner parts, a reasonably sized second-level structure (252 nodes), and a balanced distribution of documents among the leaf nodes. A weakness is the fan-out at the root of the taxonomy. It is 56, compared to a fan-out of 19 which we achieved when clustering was based upon the supplied keywords.

<table>
<thead>
<tr>
<th>version</th>
<th>corp,microsoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>nt,window</td>
<td>nt,window,</td>
</tr>
<tr>
<td>applic,develop</td>
<td>managed,network / nt,server</td>
</tr>
<tr>
<td>nt,window</td>
<td>site,web</td>
</tr>
<tr>
<td>nt,server</td>
<td>anti,virus / ip,switch</td>
</tr>
<tr>
<td>applic,develop</td>
<td>ce,window</td>
</tr>
<tr>
<td>nt,window</td>
<td>managed,network / nt,server</td>
</tr>
<tr>
<td>nt,window</td>
<td>unix,workstation / compute,</td>
</tr>
</tbody>
</table>

**Figure 2: node within LA-based Taxonomy**

Figure 2 shows an example taken from the LA-based taxonomy. It is rendered as an HTML table, where each page corresponds to exactly one node in the resulting hierarchy. A node represents a cluster, printed on the left hand side, and the outgoing links. It either links from level x down to level x+1 or (in the case of a leaf node) or it shows a terminal cluster which lists the titles of all documents belonging to this cluster. The example node is taken from the second layer of the hierarchy, i.e., it links from level 2 to level 3 (where level 1 is the root of the tree). In cases like the one shown here, where the sub-clusters have similar top-frequent terms (NT, window), the output is augmented by disambiguating terms, taken from the cluster representation. Thus, user will always navigate within a semantic context, showing the alternatives for the actual browsing decision. Browsing the hierarchy by using a standard HTML browser will map the notion of 'up' (i.e., navigate to a less specific cluster, or withdraw a browsing decision) with the browsers 'back' button.

The taxonomy shows very good coherence, i.e., related documents are grouped closely together. We selected the 5 most frequent LAs within each cluster as the cluster label. Since LAs do not directly correspond to phrases in the text (order, omissions), the readability of the labels is not optimal. Overall, the labels represent the cluster content well.
and allow for easy orientation towards interesting sub clusters.

**Judging the linguistic feature (LF)-based taxonomy:**

The HTML snapshot displayed in figure 3 is an example taken from the taxonomy which was generated based upon names, terms and general words (with some frequency-based filtering applied).

<table>
<thead>
<tr>
<th>bank</th>
<th>bank, banking, Banc One Corp</th>
</tr>
</thead>
<tbody>
<tr>
<td>banking</td>
<td>bank, fund, pay</td>
</tr>
<tr>
<td>Federal Reserve Bank</td>
<td>bank, Federal Reserve Ban, International Bank</td>
</tr>
<tr>
<td>fHome Banking</td>
<td>bank, banking, Home Banking</td>
</tr>
<tr>
<td></td>
<td>bank, NatWest Securities, comment</td>
</tr>
</tbody>
</table>

**Figure 3: node within LF-based taxonomy**

Overall, the taxonomy is broader than the LA-based taxonomy at the first and second level (73 and 283 nodes, respectively). The average branching factor is 3 with a maximum of 7 in the inner parts. Coherence is very good throughout the taxonomy. We selected the 5 most frequent LFs within each cluster as the cluster label. Since the LFs are a mixture of names, phrases, and single words, orientation is slightly worse than for LAs in some cases. Readability, though, is pretty good.

The final solution has a very high selectivity. The overall shape of the taxonomy now can be adjusted by monitoring and adjusting the importance of individual items with respect to the whole collection, which is called Information Quotient (IQ). We achieved an even more thorough collection analysis by feeding back the IQ analysis into the feature extraction methods to boost the number of relevant features.

**Summary of Analysis:**

In general, we observed that the taxonomies we generate are well-balanced and coherent down from level 2 of the hierarchy. The root level of the hierarchy seems to be more problematic and some application programming effort will be necessary to change this. The leaf nodes of the taxonomies are very coherent and related documents are within short distance in the taxonomy. Selecting the n most frequent terms in a cluster for labeling gives good overall orientation in the structure.

**6. Future Work**

The computation of lexical affinities identifies not just well-formed noun phrases but also statistical collocations (syntactic formations with no well-defined meanings). Additionally, the application of a stemming algorithm makes LAs bear less information than full forms. The labels we assigned to clusters were taken from this raw material, and thus are sometimes hard to read.

We have developed a method which allows us to re-map the vector terms to the noun phrases, proper names (most commonly) or verb phrases (rarely) from which they originated from. The method uses a thesaurus over the entire corpus, which is generated automatically. Labeling then is a scored fuzzy matching procedure over the thesaurus. Our experiments showed that for the document corpus, thesaurus generation and label forming adds a negligible amount of time to the complete process. Labels generated are e.g., „United Nations“, „Local government bodies“, or „Business patents“. We reached an accuracy of over 84% in finding meaningful phrases mentioned in the original documents. Some further work is required to enhance the quality and evaluate the re-labeled taxonomy.

Overall, we observed that many parameters of this solution still can be adjusted to personal needs, i.e. there is an administrative task to generate different “feelings” or views of the taxonomy, without loosing the overall quality or missing the requirements.

**7. Conclusions and Status**

The automatic generation of a taxonomy for a large document collection could be achieved in reasonable time by means of the Intelligent Miner For Text product suite and a few supplementary modules like a graphical tree browser. At the start of the project we established a list of minimum quality criteria which were met or even exceeded. The overall coherence of the generated taxonomy was very high,
for both (linguistic) preprocessing techniques. There are no serious scalability problems for the TaxGen design. Even a database that is an order of magnitude larger will not cause a problem - the training of the categories showed a robust behavior if the feature extraction, the clustering and the categorization tool work on the same excerpt and representation of the documents.

Hierarchical agglomerative clustering is an appropriate algorithm to group rich text documents into coherent clusters. The levels of abstraction, which where drawn by the HCA algorithm, identified mostly convincing neighborhood relationships.

However, clustering, routing and visualizing of document relationships must be accompanied by an intuitive representation of the contents of a document or a cluster of documents. The label generation for resulting clusters becomes more and more complicated as clusters grow larger. Since the generated labels showed only the most important features per cluster, the minor topics - although clustered and routed under the right sub-tree of the hierarchy - could not be represented at the top layer of the resulting hierarchy. For this special case, we imagine a thesaurus or other manually edited meta data structure (e.g. the keywords used in a news wire stream) to provide the appropriate domain specific knowledge. This information can be phased into the clustering algorithm and provide a convincing, intuitive and well-balanced root structure of the generated taxonomy.

Demonstrations: A demonstration related to the results of the TaxGen project can be found at the IBM web site http://www.software.ibm.com/data/iminer/fortext/.

References

2. A recent list of applications, tools, companies etc. involved in Data Mining can be found at the URL http://www.kdnuggets.com
5. U. Thiel, A. Müller: Why was this item retrieved? New ways to explore retrieval results, in: Information Retrieval and Hypertext, A. Smeaton and M. Agosti (eds.), 1996, Kluwer
7. Y. Maarek, D. Berry, G. Kaiser: An information retrieval approach for automatically constructing software libraries, in: Transactions on Software Engineering, 17(8), 1991