Usability study for domain-independent clustering of large document sets

Ingo DEVOOGHT and Melanie GNASA
Institute for Computer Science III, University Bonn
Römerstr. 164, D-53117 Bonn, Germany

and

Karin HARBUSCH
Computer Science Department, University of Koblenz–Landau
Universitätsstr. 1, D-56070 Koblenz, Germany

ABSTRACT

In this paper we present a usability study of clustering a randomly collected large set of documents (nearly 500,000) with respect to time- and space-complexity. An important prerequisite of our study is to use only conventional hardware (normal PCs) instead of high-end computers and free software (such as MYSQL as our underlaying database system) in order to build a cost-efficient solution for a clipping of the internet. Here, a clipping explicitly refers to portals as we can assume an upper bound for the number of documents and furthermore the process does not run necessarily at search time. As our investigation shows the number of documents to be considered remains below 500,000 documents. Clustering of these documents provides an automatically imposed further structuring in individual sub-areas. Beside time- and space-complexity measuring, our study also evaluates whether this structuring provides suitable groupings for search refinement. All in all, our usability study shows that domain-independent clustering provides a real-time behaviour for a fix test area of nearly 500,000 documents and a convincing grouping behaviour although we only impose heuristic clustering methods (single pass and a two step algorithm with a non-hierarchic first step and a hierarchic second phase) on the system.

Keywords: Heuristic clustering, two-step clustering, test suite, evaluation, usability study, search refinement

1. INTRODUCTION

Search engines for the internet are subject of fast progressing technology developments: everyday search engines have to deal with millions of search requests which navigate through several hundred millions documents in order to fulfill the individual information need of the users within a fraction of a second. In order to guarantee an efficient processing highly optimized algorithms are needed. In this paper we investigate clustering as a technique to improve internet search.

Clustering of test documents with different topics results in a thematic grouping. Algorithms for clustering are useful for improving the search results and their visual presentation [2] as the resulting groupings provided to the user allow to suppress uninteresting areas also covered by the search query. Due to the very CPU- and memory-intensive cluster computation common search engines elude this technique. Basically this is a consequence of the exponential growth of the number of internet sites on the one hand and the well-known time complexity of several clustering algorithms (cf. [10] for an overview) on the other.

Our investigation concentrates on portals which are frequently provided in the internet (cf., e.g., in various business areas). Here we illustrate clustering in portal sites as a helpful structuring of the search results. For these sites we assume 500,000 documents as an upper bound for the number of documents to be considered. Clustering of portal sites provides on the one hand a check whether the provided areas are plausible (single element clusters should be checked by hand) and on the other, a structure within the individual areas themselves.

In the following we present a usability study of clustering a randomly collected large set of documents (nearly 500,000) with respect to time- and space-complexity. An important prerequisite of our study is to use only conventional hardware (normal PCs) instead of high-end computers and free software (such as MYSQL as our underlaying database system) in order to build a cost-efficient solution for a clipping of the internet. Here, a clipping explicitly refers to portals as we can assume an upper bound for the number of documents and furthermore the process does not run necessarily at search time. As our investigation shows the number of documents to be considered remains below 500,000 documents. Clustering of these documents provides an automatically imposed further structuring into individual sub-areas. Beside time- and space-complexity measuring, our study also evaluates whether this structuring provides suitable groupings for search refinement. All in all, our usability study shows that domain-independent clustering provides a real-time behaviour for a fix test area of nearly 500,000 documents and a convincing grouping behaviour although we only impose heuristic clustering methods on the system.

The paper is organized as follows. In the next section the test suite and its prerequisites are described. Thereafter, the basic system's design is portrayed. In Section 4 the clustering methods impinging on our system are outlined. In section 5 the time and space complexity on the one hand and the grouping behaviour are evaluated on the other. In section 6 related work is presented. The paper ends addressing open problems and future work.

2. THE TEST SUITE

To simulate a clipping of the internet we first built a fix test suite with 403,450 documents in order to circumvent that the dynamic character of the internet where almost daily documents are updated or deleted influences the evaluation. To guarantee that these documents represent a random snapshot of the internet we collect the documents from the cache servers of different universities and about 55,000 documents were spidered by bots [5]. Actually many more documents were originally collected but only HTMl-documents of one language were actually added to the test suite by

an automatic filter in order to be finally able to evaluate the clustering quality. For instance pictures would probably form a cluster. This fairly simple differentiation does not help to identify weaknesses in grouping.

In order to collect documents we deployed two bots to recursively spider HTML-documents (cf. HT-Track [23] and GNU Wget (cf. http://www.gnu.org/manual/wget)). HT-Track started with the results of the search engine google (cf. http://www.google.com) for the word “doku” (German abbreviation for ’document (=ation’)). GNU Wget cannot deal with links that contain the symbol “?” which is the case for the google page. Therefore this bot started at YAHOO’s home page (cf. http://www.yahoo.com). To give an idea of the time complexity of this endeavor (under LINUX on a Pentium III (500 MHz) and 256 MB memory, GNU Wget collected within one hour 7, 000 and after one week 170, 000 documents).

Compared to the time complexity to spider documents, it is much simpler to use the content of proxy servers, i.e. locally stored documents from the internet. We used this resource (caches of three German universities; cf. Table 1) as it provides a huge collection of documents. They can be assumed to be random due to the fact that documents of thousands of users are stored in proxies.

**Table 1: Origins of collected documents**

<table>
<thead>
<tr>
<th>Method</th>
<th>Origin</th>
<th>Filtered out</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bots</td>
<td>55</td>
<td>652</td>
</tr>
<tr>
<td>Cache of FH Koblenz</td>
<td>90</td>
<td>275</td>
</tr>
<tr>
<td>Cache of Univ. Koblenz</td>
<td>322</td>
<td>965</td>
</tr>
<tr>
<td>Cache of FH Worms</td>
<td>123</td>
<td>737</td>
</tr>
<tr>
<td>Total</td>
<td>592</td>
<td>128,979</td>
</tr>
</tbody>
</table>

The documents collected by the above mentioned techniques are filtered according to the following four criteria:

1. Same document names (e.g. for HTML-document the name index.html rather frequently occurs) are automatically extended by unique numbers (at the cost of \( O(n^2) \)).

2. In order to make sure that only HTML-documents are collected, it does not suffice to check the extension “.html” of the currently considered document. Furthermore the content has to be checked for keywords such as “<html>” or “<body>”.

3. The language of documents cannot only be determined by the domain the documents come from (e.g., in “de” many English documents are yielded). In order to collect only German documents the program TextCat [4] classifies documents of individual languages on the basis of N-grams for letters, i.e. the most frequent patterns characterize the different languages. Only German documents were stored.

4. Only documents of a reasonable size are collected. Empty documents on the one hand and very large documents on the other were ignored. Empty documents or documents with only a few words, respectively, cannot be grouped meaningful. Very large documents take too much space and increase the execution time for any further handling. For our considerations, 10 to 10, 000 words were actually stored per document.

One of the important decisions for our approach is the choice of the data structure to guarantee efficient access to the documents. In order to build a cost-efficient system we employ the database management system (DBMS) MySQL (which is an open source free software product; [19]) for storing the test suite as described in the next Section and the cluster structure (cf. end of Section 3). The lack of transaction handling does not spoil the whole system as no parallel access to the data is required in our application.

**3. SYSTEM DESIGN**

Table 2 specifies the design of our system more formally according to the faceted classification of information retrieval (IR) systems by Frakes & Baeza-Yates (cf. [7]).

**Table 2: Facets and terms characterizing our system’s IR behaviour**

<table>
<thead>
<tr>
<th>Facets</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Model</td>
<td>Vector Space</td>
</tr>
<tr>
<td>File Structure</td>
<td>Inverted File</td>
</tr>
<tr>
<td>Query Operations</td>
<td>Parse, Cluster</td>
</tr>
<tr>
<td>Term Operations</td>
<td>Weight, Theasaurus, Stoplist</td>
</tr>
<tr>
<td>Document Operations</td>
<td>Parse, Cluster, Display, Ranking, Sort, Assign IDs</td>
</tr>
<tr>
<td>Hardware</td>
<td>vonNeumann, Magnetic Disk</td>
</tr>
</tbody>
</table>

With respect to this classification, the following structures in the DBMS are defined to represent a Vector Space Model [15] for the collected documents (for the future we aim at a hybrid search engine that switches to full-text search in case of insufficient clustering). Inverted files are applied as a general and widely applied method [14]. In order to represent that an ordered list of possibly weighted terms is related to the documents the terms occur in, we store word/id and doc/id (note that the use of identifiers is not entirely necessary as the documents and terms can also be directly addressed; however, superimposing ids saves some space). On the basis of freq, i.e. the overall frequency of the terms, weight, i.e. the inverse document frequency (idf) is computed (also considering doc_freq from table words). This value represents the significance of the terms in the documents. The weights will be used in clustering (words which do not belong to the table words are supposed to have the value 0, i.e. we ignore them completely for clustering). These weights can be computed beforehand and when the documents are filled into the data base scheme. This helps to gain run-time efficiency in clustering. As this table relates documents and terms (indexing) we call it search_index.

In the table words, word/id uniquely refers to an associated word (cf. column word²). The value total_number accounts for the overall occurrence of the word in all documents. Accordingly, low-frequency words (i.e. total_number < threshold) can be ignored. The column doc_freq signifies in how many documents the word occurs. From this value the inverse document frequency (idf) can be computed. The last column cluster/id filter specifies:

- \( n \) means that the word is low-frequency according to a threshold that is given to the system as a parameter (we ran our evaluation with a threshold of 8, 140);  
- \( s \) denotes stopwords according to table stopwords which are ignored for clustering however they reside in the table as they have to be accessed in a full-text search;  
- for words which do not fulfil any of these constraints, the cell remains empty.

In the table docs, doc/id uniquely refers to an associated document (cf. column name). The values of doc_weight (in 7In our implementation, the length of words is restricted to 20 symbols for space and performance reasons. If this value is exceeded the word is ignored as many of these words are compound nouns where the constituents also occur as terms themselves and are regarded anyhow.)
our system we compute the cosine coefficient) support full- 
text search, i.e. they reduce the number of queries for the 
weighted terms. For our considerations here the column can 
basic ally be ignored. 

The table stopwords contains words that are too frequent 
among the document so that they are not appropriate for 
discrimination. Our list contains 556 German words taken 
discretely from the document set. Another optional term op- 
eration relies on table thesaurus where synonymous words 
and some basic stemming is provided in order to reduce the 
number of index terms.

These tables constitute the Vector Space Model that un- 
underlines our information retrieval system (see Figure 1 for a 
snapshot of the relational structures between the individual 
lists). In the preprocessing that assembles the DBMS structure 
of the filtered documents, the inverse document frequencies 
are finally computed to be used in clustering.

![Figure 1: A snapshot of the DMBS representation of the 
Vector Space Model for our test suite](image)

<table>
<thead>
<tr>
<th>Table docs</th>
<th>Table stopwords</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc_id</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>122.html</td>
</tr>
<tr>
<td>1</td>
<td>122l.html</td>
</tr>
<tr>
<td>17</td>
<td>122.html</td>
</tr>
<tr>
<td>18</td>
<td>124.html</td>
</tr>
<tr>
<td>123</td>
<td>125.html</td>
</tr>
<tr>
<td>124</td>
<td>129.html</td>
</tr>
<tr>
<td>125</td>
<td>126.html</td>
</tr>
<tr>
<td>126</td>
<td>127.html</td>
</tr>
</tbody>
</table>

![Figure 2: System design](image)

Methods. In table cluster, cluster.id, word.id and weight 
are yielded. Some statistical values with respect to clustering 
are collected in the table cluster.status which consists of columns cluster.id, cluster.size and cluster.weight. A 
snapshot of these tables is provided in Figure 2. These cells are actually filled is topic of the next section.

![Figure 3: A snapshot of the DMBS representation of the 
cluster results](image)

4. CLUSTERING METHODS

According to [12] clustering can be defined as “the art 
of finding groups in data”, i.e. m documents are divided over 
n groups according to the relatedness of their content. Re- 
latedness is expressed in terms of a similarity measure (e.g., 
Euclidean Distance). For instance, [18] or [7] divide clustering 
into the classes hierarchical methods and non-hierarchical 
methods. Hierarchical clustering, i.e. sequences of nested 
partitions are found by merging two partitions every step, is 
divided into the classes agglomerative (glue together smaller 
clusters) and divisive techniques (fragment a larger cluster 
into smaller ones) according to the processing direction 
(bottom-up vs. top-down). In contrast to hierarchical clustering, 
where every pair of objects is compared, non-hierarchical 
(also called heuristic or partitional) clustering methods of- 
ten start out with a partition based on randomly selected 
seeds (one seed per cluster) and then refine this initial part- 
tition. Most non-hierarchical methods employ several passes 
of reallocating objects to the currently best cluster.

Due to the well-known worst case time- and space-complexity of hierarchical cluster algorithms (e.g. complete 
link with a time-complexity $O(n^2)$ and space-complexity $O(n^2)$ [7]) we decided to impose the non-hierarchical 
clustering algorithm single pass (cf. [16]) on our system. In 
order to evaluate the results they are compared to the hierar- 
chic group average method (cf. next Section). Group average 
means that the average similarity (a compromise between
the single link method's account for the greatest similarity and the complete link method's account for the least similarity is the criterion for merges. In terms of the criteria presented by van Rijsbergen [17]:

1. the theoretical soundness and
2. the efficiency of clustering algorithms

the single pass cluster algorithm satisfies criterion 2. Criterion 1 could not be fulfilled because the algorithm is not independent of the initial order of the documents. For our approach we have proven that this fact does not influence the retrieval effectiveness.

The single pass method [16] is one of the simplest and fastest non-hierarchic clustering methods (time complexity \(O(n \log n)\) and space complexity \(O(m)\)). In this method the similarity is computed between the currently considered input and a single representative of any existing cluster. Accordingly, each document is only once regarded (single pass). Actually, several disadvantages of this method are well-known, e.g., at the beginning of the clustering process large clusters could be computed and the cluster structure can be influenced by the processing order of the documents. However, these disadvantages have had no critical impact on the system for our test suite (cf. next Section).

In line with approaches like BIRCH [22], CURE [8] and Grouper[26] which all rely on a shallow first phase (see Section 6) we try to gain an appropriate grouping behaviour by revising the results of the single pass method by more fine-grained clustering in a second phase. A further clustering process (called SP+GA method) lines up the single pass and the group average method. The objective of such a two-step clustering is the coarse-grained decrease of the documents set in a first step and an elaborate clustering of a smaller set of documents with a hierarchic algorithm (cf. the experience with two-step hierarchic method BIRCH [22]; see Section 6). Compared to pure hierarchic clustering, the processing time can be decreased considerably \((O(2(n \log n))\), because the hierarchic method only considers a subset of the document set. Furthermore, the undesirably production of very large or very small clusters, respectively, is relieved. For our test suite we could show that this method partially generates the same cluster structure as the simple group average method in a quarter of its time.

5. EVALUATION

Time and space complexity

In order to compare the runtime behaviour of any clustering method described in the previous Section, Table 3 considers only a small subset of 133 documents.

Table 3: Execution time for 133 documents

<table>
<thead>
<tr>
<th>Method</th>
<th>Execution time</th>
<th>Worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td>group average</td>
<td>32.85 sec</td>
<td>(O(n^2))</td>
</tr>
<tr>
<td>single pass</td>
<td>11.87 sec</td>
<td>(O(n \log n))</td>
</tr>
<tr>
<td>SP+GA</td>
<td>12.24 sec</td>
<td>(O(2(n \log n)))</td>
</tr>
</tbody>
</table>

In order to give a flavour of the overall off-line clustering process Table 4 provides results with respect to execution time for clustering the randomly collected document set with the fastest non-hierarchic method. All computations were performed by a Pentium III with 500 MHz and 512 MB work space running windows NT 4.0 (under windows 98 the performance slightly differs; under LINUX and Solaris the programs basically work but we did not run benchmark tests).

Grouping behaviour

In order to overlooked and estimate the grouping behaviour of the two clustering methods, 133 documents (all taken from the YAHOO portal site) underlay the following study. In order to contrast the two heuristic methods the grouping behaviour of the group average method is also portrayed here. As far as possible, i.e. if this criterion can actually be triggered, we parameterise the system to compute 8 clusters. In the following we explore the clustering methods described above. Additionally, the non-hierarchic clustering method of the commercial system IBM Intelligent Miner — see http://www-4.ibm.com/software/data/miner — is compared to the results of the single pass method with respect to the actually produced clusters.

As outlined in Table 5 and Table 6, the group average and the single pass method distribute all 133 documents over 8 clusters (as the number of clusters can be parameterized here). Both methods provide frequently occurring terms in order to characterize the cluster (keywords4).

Table 4: Execution time of the single pass method regarding document-set size

<table>
<thead>
<tr>
<th>Number of documents</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, 890</td>
<td>ca. 21 min</td>
</tr>
<tr>
<td>56, 233</td>
<td>ca. 10 h</td>
</tr>
<tr>
<td>463, 450</td>
<td>ca. 7 days</td>
</tr>
</tbody>
</table>

Table 5: Clusters of the group average method (here and in the following tables, column # denotes the number of documents)

<table>
<thead>
<tr>
<th>ID #</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33 football, league, yahoo, union, sport, ski</td>
</tr>
<tr>
<td>2</td>
<td>42 yahoo, assurance, categories, finance, abc</td>
</tr>
<tr>
<td>8</td>
<td>1 yahoo, costs, categories, finance, personalize</td>
</tr>
<tr>
<td>9</td>
<td>45 sport, yahoo, Olympia, sydney, tennis</td>
</tr>
<tr>
<td>38</td>
<td>7 stock, exchange, yahoo, capital, success</td>
</tr>
<tr>
<td>40</td>
<td>3 jordan, ferrari, live, f1, sport</td>
</tr>
<tr>
<td>75</td>
<td>1 investment, much, finance, Germany, exchange</td>
</tr>
<tr>
<td>124</td>
<td>1 insurer, treatment, performance, disagreement, yahoo</td>
</tr>
</tbody>
</table>

In Table 6, the first cluster discusses the subject sport and in detail soccer. All documents grouped in cluster (2) concentrate on the subject assurances. The third cluster with only one document about finances is a special case, because the single document is the homepage of a finance portal with few information for grouping. Cluster (4) groups also sport pages but especially documents about the olympic games. The next cluster again deals with finance information and in detail with stock investments. Cluster 6 discusses formula-one relevant subjects. Cluster (7) is again a sport cluster especially about the soccer world championship. The last cluster is also a special case with only one document about finances.

A detailed comparison of both cluster structures shows that the following pairs of cluster ID numbers refer to the same document groups (using the notation (Cluster ID of single pass clustering):(Cluster ID of group average clustering)): (1):(1); (3):(8); (5):(38); (6):(40); (8):(75). Interestingly, cluster (2) of the single pass method is divided into

4 As we work on a German test suite, the keywords in the following tables have been translated to give the reader a flavour of the grouping behaviour. The symbol " _ " between words indicates German compounds originally spelled as one word.

5 The non-consecutive Cluster IDs result from the entire method of merging clusters and are no basis for a direct comparison with the single pass method.
two separate clusters by the group average method (cf. cluster (2) and (124)). The splitting performed by the group average method is comprehensible by a deeper analysis of the document's contents. Single pass cluster (2) discusses the theme ‘assurances’ in general whereas the one detached document in group average cluster (124) only deals with a fringe of the global subject. In contrast to this observation, the documents of cluster (4) and (7) with respect to the group average method are grouped by the single pass method together in cluster (9). This is an example that the group average algorithm does not always divide the documents as strict as the single pass method. All in all, the single pass method provides reasonable results and an acceptable runtime (O(n log n)).

In the hybrid non-hierarchic method SP+AG (cf. Table 7, the single pass is slightly differently parameterized than for the results in Table 5 in order to produce more clusters to be merged in the second pass (six clusters become actual merged in the second step to end up finally with eight clusters). ID₁ enumerates the number after the first pass whereas ID₂ enumerates the final cluster ids. The method delivers the same classification as the pure group average method. We have performed several tests with larger document sets and a variety of numbers of clusters to be actually constructed by this method. As we observed, this hybrid method can compute very similar cluster to the group average method in a quarter of its time. The optimal parameter setting should trigger that a reasonable fine-grained clustering is performed in the first step by the single pass method and only some comparisons with the centroid of a cluster is left to the group average method.

Let us now use the same document set to cluster with the non-hierarchic clustering method of the IBM Intelligent Miner. However, this tool does not provide a parameter to specify the desired number of clusters. IBM Intelligent Miner's binary relational clustering ends up with 10 clusters (cf. Table 8) which do not coincide with the group average clustering with 30 clusters (only one cluster is identical). The binary relational clustering seems to mix topics all over the cluster structure. This grouping is apparently not very helpful for navigation in search lists or on portal sites.

Table 6: Clusters of the single pass algorithm

<table>
<thead>
<tr>
<th>ID</th>
<th>#</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>33</td>
<td>football_league, yahoo, union, sport, ski</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>yahoo, assurance, categories, finance, abc</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>yahoo, costs, categories, finance, personalize</td>
</tr>
<tr>
<td>4</td>
<td>29</td>
<td>sport, Olympia, modal, yahoo, sydney</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>stock, exchange, yahoo, capital, success</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>jordan, ferrari, live, fl, sport</td>
</tr>
<tr>
<td>7</td>
<td>16</td>
<td>sport, soccer, world_championship, sporty, yahoo</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>investment, much, finance, Germany, exchange</td>
</tr>
</tbody>
</table>

6. RELATED WORK

Basically, we refer to other studies which outline promising grouping behaviour and runtime efficiency of various clustering methods. Particularly, we portray some recent approaches where clustering is applied on the internet. Scatter/Gather [8] is a cluster-based approach to browse large document collections. It relies on the methods Buckshot and Fractionation to compute the initial centers. Even imposing the non-deterministic Buckshot method results in a reliable cluster structure. However, the authors report that a complete clustering with the linear time methods Buckshot or Fractionation is too slow for extremely large corpora at least on an on-line application. BIRCH [22] is an efficient data cluster method (e.g., for codebook generation in image compression or pixel classification). It first computes summaries that retain as much distribution information as possible and then clusters these summaries instead of the original data set. This method reminds to [20], where the so-called “snippets” provided by the search engines to characterize retrieved documents are clustered. We return to this method based on Suffix Tree Clustering (STC) later in this Section. CURE [8] employs a combination of random
sampling and partitioning, i.e. also imposing a preprocess to prevent the time-consuming clustering from dealing with any document. A random sample drawn from the data set is first partitioned and then each partition is partially clustered. As reported by the authors, CURE is twice as fast as BIRCH. Our method SP+GA is basically influenced by these methods which propose a preprocessing of the data set before clustering. Actually we rely in a decent clustering in the first phase.

Marti Hearst [9] reports on characteristics of clustering retrieval results. Basically, it can be observed that the actual grouping behaviour directly depends on the homogeneity/heterogeneity of the document set (also reported in [1]). This study suggests that clusters help to filter out sets of documents that are clearly not relevant and should be ignored for homogeneous document sets on the one hand.

On the other, clustering of heterogeneous document sets splits into general genres — which is a beneficial property for search engines. Clustering based on Genetic algorithms (see, e.g. [11]) are reported to perform not as good as traditional methods. In [13], an experiment comparing several systems with different approaches of clustering is comprised. However, the emphasis of this study lies on the supplied effects of visualization. Recently, cluster-based search engines such as http://www.vivisimo.com or http://wisesum.com provide clustering as a visualization and navigation tool. In general, the applications of clustering can be divided in on-line and off-line systems. The NorthernLight engine (cf. http://www.northernlight.com) provides “Customer Folders” in which the retrieved documents are organized. According to [21] the folders seem to be computed off-line. No description of execution time is provided. According to [3], 30M HTML-documents can be grouped in over 10 CPU-days with a non-hierarchical method based on a predetermined similarity threshold. This seems to be a comparable result to our experience. Grouper [21] clusters on-line results of a meta-search engine (on the machine of the user) on the basis of suffix tree clustering [29]. This method runs in linear time. On a DEC AlphaStation 500 (333MHz with 512 RAM) 800 documents are clustered in 2.3 seconds.

7. CONCLUSIONS
We have described a large scale system based on heuristic clustering methods. All in all, single pass and SP+AG provide a convincing grouping behaviour. At least in off-line systems the execution time allows to supply automatic grouping such as for an acquisition component for portals. We have observed that navigation bars on HTML-pages can spoil clustering. Consequently, we have conducted superficial syntactic analyses of the document structure in order to find these lists and hide them away from entire clustering. First attempts are quite promising. Furthermore we have tried to superimpose incrementally on our approach (cf. BIRCH). Any clustering method that relies on the inverse document frequency requires a recomputation of the centroids. A simple solution, which totally ignores weights, i.e. any value is set to one, leads to a quite acceptable grouping behaviour. However, the list of characteristic terms has become arbitrary due to the fact that the frequency of a word in a document cannot be accounted in relation to its absolute frequency. Consequently the visualization of clusters scores more badly.

8. REFERENCES