AISEARCH: Category Formation of Web Search Results

Benno Stein and Sven Meyer zu Eissen
Department of Computer Science
Paderborn University, Germany
stein@upb.de, smze@upb.de

July 30, 2003

Abstract. Automatic category formation plays a key role in the development of future interfaces for Web-based search. We introduce the meta search engine AISEARCH that implements state-of-the-art technology for a category search in huge document collections. AISEARCH combines algorithms for query analysis, category formation, category labeling, and category visualization. Among the mentioned tasks the category formation step is the most crucial one: A smart grouping of a large number of documents that is returned by a standard query search is extremely useful—but difficult to derive.

Clustering algorithms are considered as a technology that is capable to master this "ad-hoc" categorization task. However, it is hard to say which of the existing clustering approaches is suited best. A major part of this article is devoted to this question and presents results of a comprehensive analysis of clustering algorithms in connection with category formation. The contributions relate to exemplar-based, hierarchical, and density-based clustering algorithms. In particular, we contrast ideal and real clustering settings and present runtime results that are based on efficient implementations of the investigated algorithms.

Keywords: Category Formation, Document Clustering, Meta Search, Information Retrieval

1. Web-based Search and Clustering

The World Wide Web provides a huge collection of documents, and its use as a source of information is obvious and became very popular. As pointed out in [7] there is a plethora of Web search technology, which can broadly be classified into four categories:

1. *Unassisted Keyword Search.* One or more search terms are entered and the search engine returns a ranked list of document summaries. Representatives: Google (`www.google.com`) or AltaVista (`www.altavista.com`).

2. *Assisted Keyword Search.* The search engine produces suggestions based on the user's initial query. Representative: AskJeeves (`www.askjeeves.com`).

3. *Directory-based Search.* Here, the information space is divided into a hierarchy of categories, where the user navigates from broad to specific classes. Representative: Yahoo! (`www.yahoo.com`).

4. *Query-by-Example.* The user selects an interesting document snippet, which is then used as the basis of a new query.

We think that the ideal search interface should model the search process within three phases: (a) An initialization phase according to the plain unassisted keyword search paradigm, (b) a categorization phase similar to the directory-based search paradigm, and (c) a refinement phase that may combine aspects from assisted keyword search and the query-by-example paradigm. Our realization of this process pursues a meta search strategy similar to that of Vivisimo [49]; i. e., it employs existing search technology within the initialization phase.

This idea of an ideal search process grounds on the following observations:

Existing search engines do an excellent and convenient job. They organize up to billions of documents which can be searched quickly for keywords, and, the plain keyword search forms the starting point for the majority of users. However, while this strategy works fine for the experienced human information miner, many users are faced either with an empty result list or with a list containing thousands of hits. The former situation is the result of misspelling or contradictory Boolean query formulation; it can be addressed with a syntactic analysis. The latter situation lacks a meaningful specification of context—it requires a semantic analysis, which can be provided by means of category narrowing. In this connection some search engines use a human-maintained predefined topic hierarchy with about 20 top-level categories like sports, art, music etc. Such static hierarchies are unsatisfactory within two respects: They require a considerable human maintenance effort, and, for special topics (example: "sound card driver") the categories constitute an unnecessary browsing overhead which defers the search process. A powerful focusing assistance must be based onto a query-specific—say: ad-hoc—categorization of the delivered documents.

Ad-hoc categorization comes along with two major challenges: Efficiency and nescience. Efficiency means that category formation must be performed at minimum detention, while nescience means that the category formation process is unsupervised: Except for experimental evaluation purposes, no predefined categorization scheme is given from which classification knowledge can be acquainted.

1.1. Organization of this Article

The next section gives a brief overview of the AiSearch system. AiSearch offers a convenient interface for Web-based search and combines algorithms for the formation, labeling, and visualization of categories as well as a smart spelling analysis. Since category formation plays a key role in the search process, the remaining Sections, 3 and 4, concentrate on this aspect.

Section 3 introduces background knowledge on document representation models, different types of clustering algorithms, and clustering quality measures. Section 4 reports several results from our experimental analyses and provides insights with respect to the following points:

- computational effort of essential data processing steps,
- separability of categories in terms of supervised classification tasks, and
- performance of new and well-known clustering algorithms and document models concerning cluster quality and runtime.

2. The AiSearch Meta Search Engine

A search process with the AiSearch Web interface starts as usual: A query in the form of interesting search terms is entered within a dialog field. The query is sent to several search engines and—for a syntactic analysis—to a SmartSpell server. The query results, i. e., the HTML document snippets, are collected and analyzed with respect to the similarity of their contents. Based on this analysis, adequate categories are formed and labeled, and a tree of the categories, which shows related categories at a closer distance than unrelated categories, is drawn in the hyperbolic plane. Figure 1 shows a snapshot of the AiSearch Web interface for the query "tea flavour".

Aside from the hyperbolic category tree, the returned document snippets can also be browsed in a list format. The list groups all snippets of the same category together, and,

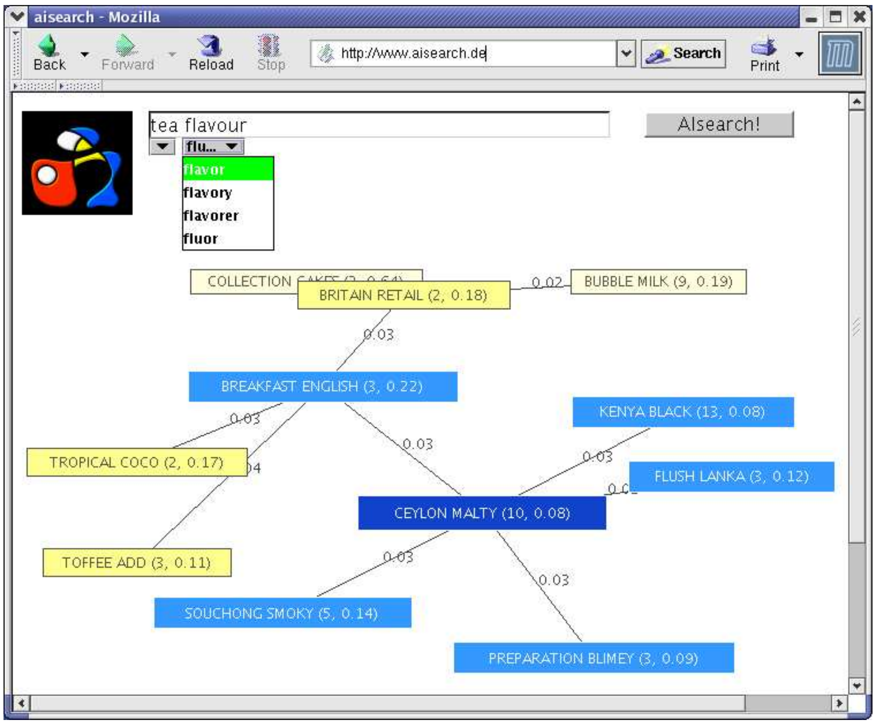

Figure 1. The AISEARCH Web interface. The query field (top) contains two search terms along with two list-boxes containing SMARTSPELL proposals with similar terms. Below the query field the category tree for the current query is displayed; its nodes correspond to categories each of which comprising 5-20 documents that belong thematically together.

immediate access to each sublist is possible by simply clicking the leafs in the category tree.

2.1. Query Analysis

During the meta search, within another background process, the terms of the query are checked with respect to both correct spelling and similar terms. For this job the powerful SMARTSPELL algorithm [1] is used. SMARTSPELL analyzes spelling errors with regard to the editing distance, the Levinshtein distance, and the phonological distance against a dictionary [15].

Especially the analysis that grounds on a phonological interpretation is a demanding task; it depends on a language's level of phonemicity and is realized with a sophisticated, phoneme-dependent word similarity measure. To efficiently find syntactic and phonetic similar words for a search term, SMARTSPELL operationalizes several paradigms of heuristic search: nogood-lemma generation, search space pruning based on over- and underestimation, iterative deepening search, and memorization [33, 32]. Table I shows some examples of misspelled words along with SMARTSPELL's proposals and similarity estimations.

SMARTSPELL's proposals of similar search terms are directly integrated in the query field; they enable the reformulation, extension, or correction of a query by the press of a button.

2.2. Category Formation

To assure user acceptance, the category formation process must be efficient with respect to both category quality and response time. That AISearch is able to fulfill these performance requirements can be seen in Section 3, which gives a snapshot of our comprehensive analysis based on document collections that were categorized by human editors.

The performance requirements are also reflected in the implemented software technology: To compare different clusterings of search results, AISearch employs strategy patterns to make term weighting schemes, similarity measures, clustering algorithms, and cluster validity measures interchangeable at runtime. For efficient text handling, the symbol processing algorithms for text parsing, text compression, and text comparison utilize specialized flyweight patterns [13].

2.3. Category Visualization

Data visualization is a research topic for decades, and a large number of methods for the visualization of document collections is available[5, 6, 10, 14, 18, 23, 29, 31, 37, 40, 43, 45]. AISearch uses a two-dimensional, navigable, hyperbolic layout (see Figure 1) for the following reasons. Firstly, it is reported that three-dimensional, navigable layouts often confuse users [30]. Secondly, only a small part of the found categories is interesting for the user. A distortion-based view that scales up interesting parts around the center of the screen and scales down the non-relevant parts at the border of the screen addresses this point. Furthermore, the interesting categories can be focused at the center of the screen with one mouse click. Thirdly, using a two-dimensional layout, related categories can be shown at a closer distance than unrelated categories; a category list cannot take this aspect into account.

As mentioned above, this tree view is combined with a list view, which organizes the documents and corresponding links according to the categories.

2.4. Software Architecture and Deployment Concepts

Figure 2 shows how the AISearch components are deployed to machines. When a user enters the AISearch URL in his browser, a Java Applet that contains the AISearch user interface is delivered from the Web server, which in turn communicates with the load balancing module. All requests from the client, such as a request for spelling or a request for search, are coded in a proprietary protocol that contains several commands. Whenever a command reaches the load balancing module, one of the AISearch engines is chosen to perform the associated task. All commands are processed asynchronously.

All computationally expensive tasks are performed as threads, which allows us to run several commands simultaneously on a single AISearch engine. Moreover, the thread-

Table I. Examples for misspelled words (left column) and the SmartSpell proposals with similarity estimations (right column).

Misspelled word	SmartSpell proposal (similarity)
aksekjushon	execution (81%)
angenearing	engineering (92%)
blu	blue (93%), blew (92%)
buysikel	physical (85%), bicycle (82%)

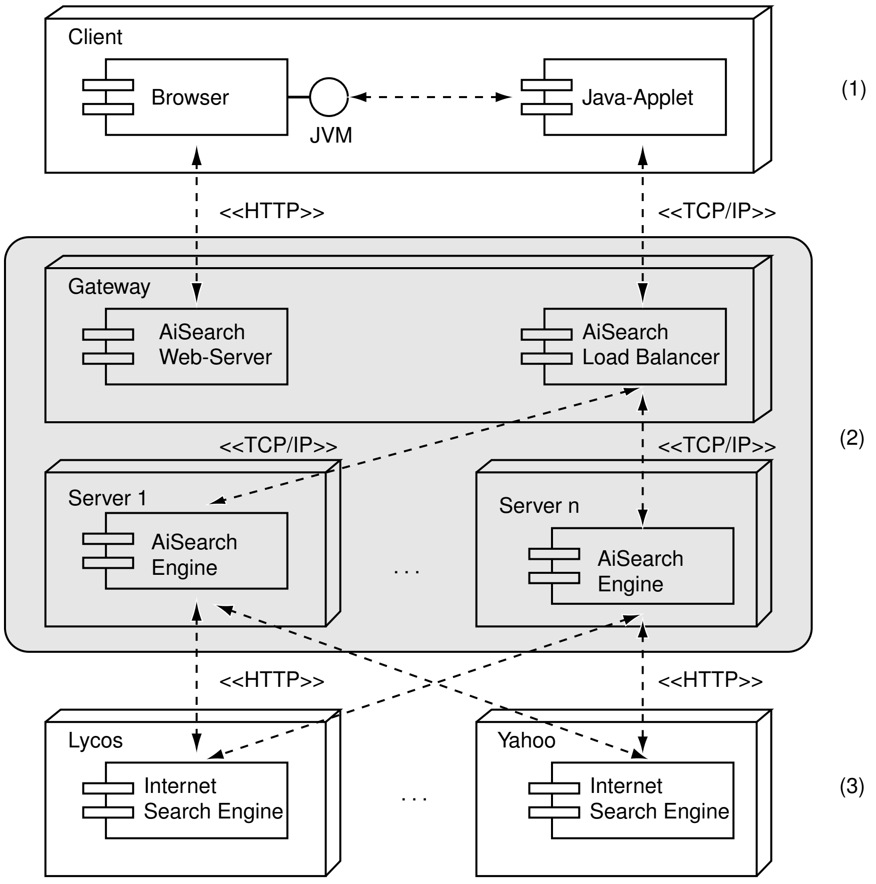

Figure 2. AISEARCH deployment diagram. The AISEARCH Web server delivers Java Applet code to the client browser (1), which in turn sends a request to the AISEARCH load balancer, which selects an AISEARCH engine to process the query (2). When the dedicated engine has queried Internet search engines (3) and completed the category formation task, the results are transfered back to the Java Applet.

ing model supports multiprocessor machines ideally, and, combined with a load balancing concept, assures a simple scalability of the architecture.

3. Background: Document Representation, Clustering, and Quality Measures

The statistical method of variance analysis is used to verify whether a classification of objects by means of nominal features is reflected in significant differences of depending metric features. Clustering can be considered as some kind of inverse operation: It tries to identify groups within an object set such that elements of different groups show significant differences with respect to their metric features.

Clustering algorithms operate on object similarities, which, in turn, are computed from abstract descriptions of the objects. Each such description is a vector $\mathbf{d}$ of numbers comprising values of essential object features. This section outlines the necessary concepts in connection with text documents: A suited object description, a related similarity measure, an overview of clustering algorithms, and—in particular, clustering quality measures for the analysis of an algorithm's categorization performance.

3.1. Document Representation

A common representation model for documents is the vector space model, where each document is represented in the term space, which roughly corresponds to the union of the m words that occur in a document collection [36, 24]. In this term space, common words are filtered out by means of a stop word list, words that are unique in the collection are omitted, and stemming is applied to reduce words towards a canonical form. Each document d_j in a document collection D can then be described by means of a vector $\mathbf{d}_j = (w_{1j}, \ldots, w_{mj})$, where w_{ij} designates the weight of term t_i in d_j. Widely accepted variants for the choice of w_{ij} are the following.

1. The term frequency $tf(t_i, d_j)$ denotes the frequency of term i in document j. Defining the weights w_{ij} as $tf(t_i, d_j)$ implies that terms that are used more frequently are rated as more important.
2. The inverse document frequency is defined as $idf(t_i) := \log(\frac{n}{df(t_i)})$, where n is the total number of documents in the collection and $df(t_i)$ is the number of documents which contain the term t_i. The hypothesis is that terms that occur rarely in a document collection are of highly discriminative power. Defining $w_{ij} := tf(t_i, d_j) \cdot idf(t_i)$ combines the hypothesis with Point (1) and has shown to improve the retrieval performance [41]. Note that the representation of a single document requires knowledge of the whole collection if the idf-concept is used.

3.2. Document Similarity

Clustering exploits knowledge about the similarity among the objects to be clustered. The similarity φ of two documents, d_i, d_j, is computed as a function of the distance between the corresponding term vectors $\mathbf{d}_i$ and $\mathbf{d}_j$. There exist various measures for similarity computation, from which the cosine-measure proved to be the most successful for document comparison. It is defined as follows.

$$\varphi(d_i, d_j) = \frac{\langle \mathbf{d}_i, \mathbf{d}_j \rangle}{||\mathbf{d}_i|| \cdot ||\mathbf{d}_j||},$$

where $\langle \mathbf{d}_i, \mathbf{d}_j \rangle = \mathbf{d}_i^T \mathbf{d}_j$ denotes the scalar product, and $||\mathbf{d}||$ the Euclidean length. It computes the cosine of the angle between two documents in $\mathbf{R}^m$. Note that a distance measure can easily be derived from φ by subtracting the similarity value from 1.

3.3. Clustering Algorithms

Definition 1 (Clustering) *Let D be a set of objects. A clustering $\mathcal{C} = \{C \mid C \subseteq D\}$ of D is a division of D into sets for which the following conditions hold: $\bigcup_{C_i \in \mathcal{C}} C_i = D$, and $\forall C_i, C_j \in \mathcal{C} : C_i \cap C_{j \neq i} = \emptyset$. The sets C_i are called clusters.*

Remarks. Here, the set of objects, D, corresponds to a document collection. Moreover, it is useful to consider the elements in D as nodes of a weighted graph G. G is completely connected, and the weight of the edge that connects two documents, d_i, d_j, corresponds to their similarity $\varphi(d_i, d_j)$.

Clustering algorithms, which generate a clustering $\mathcal{C}$, are distinguished with respect to their algorithmic properties. The following overview cannot be complete but outlines the most important classes along with the worst-case runtime behavior of prominent representatives. Again, n designates the number of documents in a given collection.

Iterative Algorithms. Iterative algorithms strive for a successive improvement of an existing clustering and can be further classified into exemplar-based and commutation-based approaches. The former assume for each cluster a representative, i. e. a centroid (for interval-scaled features) or a medoid (otherwise), to which the objects become assigned according to their similarity. Iterative algorithms need information with regard to the expected cluster number, k. Representatives: k-Means, k-Medoid, Kohonen, Fuzzy-k-Means [16, 28, 20, 22, 47]. The runtime of these methods is $\mathcal{O}(nkl)$, where l designates the number of iterations to achieve convergence.

Hierarchical Algorithms. Hierarchical algorithms create a tree of node subsets by successively subdividing or merging the graph's nodes. In order to obtain a unique clustering, a second step is necessary that prunes this tree at adequate places. Agglomerative hierarchical algorithms start with each vertex being its own cluster and union clusters iteratively. For divisive algorithms on the other hand, the entire graph initially forms one single cluster which is successively subdivided. Representatives: k-nearest-neighbor, linkage, Ward, minimum-spanning-tree, or min-cut methods [9, 11, 42, 17, 26, 46, 48]. Usually, these methods construct a complete similarity graph, which results in $\mathcal{O}(n^2)$ runtime.

Density-based Algorithms. Density-based algorithms try to separate a similarity graph into subgraphs of high connectivity values. In the ideal case they can determine the cluster number k automatically and detect clusters of arbitrary shape and size. Representatives: DBSCAN, MAJORCLUST (see Appendix A), CHAMELEON [44, 8, 19]. The runtime of these algorithms cannot be stated uniquely since it depends on diverse constraints. Typically, it is in magnitude of hierarchical algorithms, $\mathcal{O}(n^2)$, or higher.

Meta-Search Algorithms. Meta-search algorithms treat clustering as an optimization problem where a given goal criterion is to be minimized or maximized [2, 38, 39, 38]. Though this approach offers maximum flexibility, only less can be stated respecting its runtime. Representatives: Meta-search driven cluster detection may be realized by genetic algorithms [35, 12], simulated annealing [21], or a two-phase greedy strategy [50].

3.4. CLUSTERING QUALITY MEASURES

Many clustering algorithms do not return a definite clustering but a set of clusterings from which the best one has to be chosen. In particular, uniqueness within exemplar-based algorithms requires information about the cluster number, uniqueness within hierarchical algorithms requires an agglomeration threshold, or, within density-based algorithms, uniqueness requires a threshold for interpreting the neighborhood graph. If we had a measure to assess the quality of a clustering, the ambiguity could be mastered by simply computing several candidate clusterings and choosing the best one with respect to that measure. Note, however, that this is not a runtime problem in first place, but a problem of defining a suited quality measure.

Clustering quality measures evaluate the validity of a clustering and can be grouped into two categories: external and internal[1]. The following paragraphs introduce two clustering quality measures that are used within our experiments.

External Measures. External clustering quality measures use statistical tests to quantify how well a clustering matches the underlying structure of the data. In our context, the underlying structure is the known categorization of a document collection D as provided by a human editor. A broadly accepted external measure is the F-Measure, which combines the precision and recall ideas from information retrieval [25].

[1] Several authors also define relative clustering qualtity measures, which can be derived from internal measures by evaluating different clusterings and comparing their scores [20].

Let D represent the set of documents and let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of D. Moreover, let $\mathcal{C}^* = \{C_1^*, \ldots, C_l^*\}$ designate the human reference classification. Then the recall of cluster j with respect to class i, $rec(i,j)$, is defined as $|C_j \cap C_i^*|/|C_i^*|$. The precision of cluster j with respect to class i, $prec(i,j)$, is defined as $|C_j \cap C_i^*|/|C_j|$. The F-Measure combines both values as follows:

$$F_{i,j} = \frac{2}{\frac{1}{prec(i,j)} + \frac{1}{rec(i,j)}}$$

Based on this formula, the overall F-Measure of a clustering is:

$$F = \sum_{i=1}^{l} \frac{|C_i^*|}{|D|} \cdot \max_{j=1,\ldots,k} \{F_{i,j}\}$$

A perfect clustering matches the given categories exactly and leads to an F-Measure value of 1.

Internal Measures. In absence of an external judgment, internal clustering quality measures must be used to quantify the validity of a clustering. Bezdek presents a thorough analysis of several internal measures, and, in this paper we rely on a measure from the Dunn Index family, which came off well in Bezdek's experiments [4, 3].

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering, $\delta : \mathcal{C} \times \mathcal{C} \to \mathbf{R}_0^+$ be a cluster-to-cluster distance measure, and $\Delta : \mathcal{C} \to \mathbf{R}_0^+$ be a cluster diameter measure. Then all measures $d : \mathcal{C} \to \mathbf{R}_0^+$ of the form

$$d(\mathcal{C}) = \frac{\min_{i \neq j}\{\delta(C_i, C_j)\}}{\max_{1 \leq l \leq k}\{\Delta(C_l)\}}$$

are called Dunn Indices. Of course there are numerous choices for δ and Δ, and Bezdek experienced that the combination of

$$\delta(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} \psi(x, y) \quad \text{and}$$

$$\Delta(C_i) = 2\left(\frac{\sum_{x \in C_i} \psi(x, c_i)}{|C_i|}\right)$$

gave reliable results for several data sets from different domains. Here, ψ denotes a distance measure between the objects to be clustered, and c_i is the centroid of cluster C_i. Since we use the cosine similarity φ as similarity measure, we set $\psi = 1 - \varphi$.

Remarks. As mentioned at the outset, the use of external and internal measures corresponds to an idealized and realistic experimental scenario respectively: During ad-hoc categorization, only very little is known a-priori about the underlying structure of a document collection.

4. Experimental Setting and Results

This section reports results of our analysis of clustering algorithms for automatic category formation. The first two subsections, 4.1 and 4.2, provide interesting information related to document preprocessing tasks and category separability. The main contribution can be found in Subsection 4.3, which describes the clustering experiments.

Table II. Runtime and impact of selected preprocessing steps, depending on the sample size (1. column). Hardware: Pentium IV 1.7GHz.

# Documents in sample	Indexing time	Compression time	Compression ratio	# Terms (raw)	# Terms (reduced)	Term reduction
400	1.80s	0.23s	98.6%	6010	4153	31%
800	2.88s	0.64s	99.0%	8370	5725	32%
1000	3.40s	0.89s	99.1%	9192	6277	32%

For all experiments samples were drawn from the Reuters-21578 text document database [27]. Note that in this database a considerable part of the documents is assigned to more than one category. To uniquely measure the classification performance, only single-topic documents have been considered here. The sample sizes vary between 400, 800, and 1000 documents, and each sample contains documents from exactly 10 different classes. To account for the biased a-priory probabilities in the class distribution of Reuters-21578, all samples have been constructed as uniformly distributed.

4.1. Preprocessing

The generation of a term vector $\mathbf{d}$ for a document d in a sample requires certain preprocessing effort that must not be underestimated. Note that AISearch affords these computations for the results of each query.

Preprocessing includes the reading and parsing of the documents, the elimination of stop words according to standard stop word lists, the application of Porter's stemming algorithm [34], the computation of term frequencies, the creation of compressed index vectors, etc. Table II shows the runtime of important preprocessing steps, compression ratios, and term reduction ratios for different sample sizes.

4.2. Supervised Learning: Classification Performance

Though our main objective is unsupervised text categorization, we performed classification experiments as well to get an idea the difficulty of the learning problem. For this purpose a linear classifier in the form of a neural network was employed, which forced us to substantially reduce the dimension of the feature space. In this connection, the LSI-reduction with the target dimensions of 40, 20, and 10 was applied.

The LSI-reduction did not include the test documents; in fact, they were *projected* into the reduced feature space. Note that if LSI were performed on a matrix which contained both the test and the training documents, knowledge of the test data would be compiled into the LSI-reduced training data. As a consequence, the resulting classifier would be biased towards an increased classification performance.

Aside from the LSI-transformed features we also made experiments with randomly chosen indices of the document vectors. Table III comprises the results.

4.3. Unsupervised Learning: Clustering Performance

This subsection reports the categorization performance and the runtime of the following clustering algorithms: k-Means (exemplar-based), Single-Link (hierarchical), Group-Average (hierarchical), and MajorClust (density-based). The algorithms are tested within an idealized and a realistic scenario (explained below) and under both the tf-

Table III. Classification performance of the linear classifier, depending on the sample size (1. column), the number of features (3. column), and the feature reduction method (4. column).

# Documents in sample (training/test)	# Classes in sample	# Features	Feature reduction method	correctly classified
400/100	10	40	LSI	0.92
400/100	10	40	Random	0.16
800/200	10	40	LSI	0.86
800/200	10	40	Random	0.23
800/200	10	20	LSI	0.83
800/200	10	20	Random	0.11
800/200	10	10	LSI	0.75
800/200	10	10	Random	0.11

Table IV. Categorization performance of the clustering algorithms under the tf-document model. The maximum F-Measure (4. column) corresponds to the ideal scenario, the 5. and the 6. column correspond to the realistic scenario.

Clustering algorithm	# Documents in sample	# Classes in sample	F-Measure maximum	F-Measure Dunn Index	F-Measure elbow criterion
k-Means	1000	10	0.75	0.35	0.69
Single-Link	1000	10	0.18	0.18	0.18
Group-Average	1000	10	0.63	0.18	0.18
MAJORCLUST	1000	10	0.63	0.51	0.57

document model and the $tf\text{-}idf$-document model. Table IV and Table V comprise the experiments for the former and the latter document model respectively.

Each clustering algorithm is applied within a wide range of its respective variable parameter p while paying attention to special algorithmic properties and strengths. From the resulting set of clusterings $\boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{C}} = \{\mathcal{C}(p_1), \ldots, \mathcal{C}(p_n)\}$, the optimum clustering with respect to a given quality measure $Q, Q : \boldsymbol{\mathcal{C}} \to \mathbf{R}^+$ is chosen:

$$q = \text{argmax}_{p,\ p=p_1,\ldots,p_n}\ Q(\boldsymbol{\mathcal{C}}) \qquad (1)$$

In our setting, the variations of p relate to the following parameters: For k-Means, variations in the cluster number k, $k = 1, \ldots, 20$ and three random restarts for each k are considered. For Single-Link and Group-Average, all clusterings of the last 20 agglomeration levels are analyzed. For MAJORCLUST, the threshold for edge weights is successively advanced within 20 steps, and three random restarts for each threshold are performed.

A matter of particular interest is the distinction between an idealized and a realistic scenario. Within the idealized scenario, the best clustering of an algorithm is determined by means of the F-Measure. This gives us information about the quality that could be achieved by the algorithm and is reported in the 4. column in Table IV and Table V. Note, however, that the computation of the F-Measure requires knowledge about the true document classification—which, of course, is unknown when categorizing query results.

Within the realistic scenario, the quality of a clustering is assessed by means of internal measures. Put another way, the optimum parameters for the cluster number, the agglomeration level, and the edge weight threshold must be estimated. From the various number of internal measures we have chosen the approved Dunn Index and the variance drop (elbow criterion) to evaluate the clustering quality, say, to compute Equation (1). To get an idea of the reliability of these measures, we did also compute the F-Measure of that clustering that maximizes the respective internal measure (see the 5. and the 6. column in the tables).

All clustering experiments were performed on a Pentium IV 1.7GHz; Table VI shows selected numbers of the averaged runtime of the investigated algorithms. It should be noted that our text processing and classification environment is implemented in Java—but has been developed in the face of efficiency. Among others, we developed tailored classes for symbol processing, efficient vector updating, and compressed term vectors.

Note that the standard versions of Single-Link, Group-Average, and MAJORCLUST operate on a completely connected distance or similarity graph. Figure 3 shows how the edge weights are distributed in our samples. Of course, the creation of the graph imposes a severe performance burden, which can also be observed in the 4. column of Table VI.

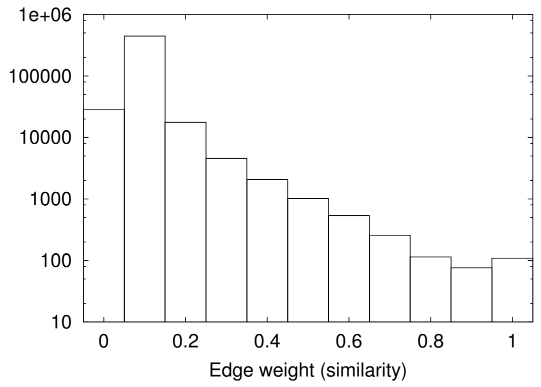

Figure 3. Distribution of edge weights under the tf-document model in a completely connected graph with thousand nodes; nodes correspond to documents, and edge weights correspond to similarities. Observe the logarithmic scale.

5. Summary

This article introduced AISEARCH, a Web search interface that combines state-of-the-art technology for query analysis, category formation, category labeling, and category visualization. Because of its great importance, a major part of this article is devoted

Table V. Categorization performance of the clustering algorithms under the tf-idf-document model. The maximum F-Measure (4. column) corresponds to the ideal scenario, the 5. and the 6. column correspond to the realistic scenario.

Clustering algorithm	# Documents in sample	# Classes in sample	F-Measure maximum	F-Measure Dunn Index	F-Measure elbow criterion
k-Means	1000	10	0.75	0.75	0.56
Single-Link	1000	10	0.18	0.18	0.18
Group-Average	1000	10	0.30	0.18	0.30
MAJORCLUST	1000	10	0.80	0.73	0.70

Table VI. Runtime of the clustering algorithms. Hardware: Pentium IV 1.7GHz.

Clustering algorithm	# Documents in sample	Preprocessing time	Graph creation time	Clustering time
k-Means	1000	4.29s	–	2.33s
Single-Link	1000	4.29s	5.78s	1.58s
Group-Average	1000	4.29s	5.78s	1.58s
MAJORCLUST	1000	4.29s	5.78s	2.38s

to the category formation step. To automate this step two key problems must be addressed: Efficiency—category formation must be performed at minimum detention, and nescience—no predefined categorization scheme is given.

Clustering algorithms are considered as a technology that is capable of mastering these challenges, and this paper provides selected results of a comprehensive analysis. We compared the categorization performance of exemplar-based, hierarchical, and density-based clustering algorithms within an idealized and a realistic scenario and under two document models. The main result of the experiments can be summarized as follows.

Aside from the Single-Link algorithm, the categorization performance for samples drawn from the Reuters-21578 text database achieves acceptable values—especially in an ideal scenario, where an external clustering quality measure is given. Moreover, our analysis shows that even in a realistic scenario reasonable F-Measure values can be achieved. Here, a crucial role comes up to the internal clustering quality measure, which can completely ruin smart clustering technology. The presented figures give an example: The well-known Dunn Index performs not better than a simple variance-based elbow criterion—or, put another way, a quality measure must be chosen with respect to the document model, the similarity measure, and the clustering algorithm.

The AISEARCH system provides efficient implementations of several clustering algorithms and quality measures. Its response time is about 2s for a query on a Pentium IV 1.7Ghz; this time includes the query analysis and the preprocessing, categorization, labeling, and visualization of 200 results. AISEARCH has left its prototype stage and shall be deployed in a productive environment in the near future.

Appendix

A. The MAJORCLUST Algorithm

MAJORCLUST is a new clustering algorithm presented in [44]. It strives at a maximization of a graph's *weighted partial connectivity* Λ, which is defined as follows.

Definition 2 **(Λ)** *Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of a weighted graph $G = \langle V, E, \varphi \rangle$.*

$$\Lambda(\mathcal{C}) := \sum_{i=1}^{k} |C_i| \cdot \lambda_i,$$

where λ_i designates the weighted edge connectivity of $G(C_i)$. The weighted edge connectivity, λ, of a graph $G = \langle V, E, \varphi \rangle$ is defined as $\min \sum_{\{u,v\} \in E'} \varphi(u, v)$ where $E' \subset E$ and $G' = \langle V, E \setminus E' \rangle$ is not connected. λ is also designated as the capacity of a minimum cut of G.

Initially, MAJORCLUST assigns each node n of a graph its own cluster $c(n)$. Within the following re-clustering steps, a node adopts the same cluster as the majority of its weighted neighbors. If several such clusters exist, one of them is chosen randomly. If re-clustering comes to an end, the algorithm terminates. Figure 4 shows the different assignment situations pictorially.

MAJORCLUST.

Input. A graph $G = \langle V, E, \varphi \rangle$.
Output. A function $c : V \rightarrow \mathbf{N}$, which assigns a cluster number to each node.

```
(1) $n = 0, t = \mathit{false}$
(2) $\forall v \in V$ do $n = n + 1, c(v) = n$ end
(3) while $t = \mathit{false}$ do
(4)     $t = \mathit{true}$
(5)     $\forall v \in V$ do
(6)         $c^* = \text{argmax}_{i,\ i=1,\ldots,n} \left( \sum_{\substack{\{u,v\} \in E \\ \wedge\ c(u)=i}} \varphi(u,v) \right)$
(7)         if $c(v) \neq c^*$ then $c(v) = c^*, t = \mathit{false}$
(8)     end
(9) end
```

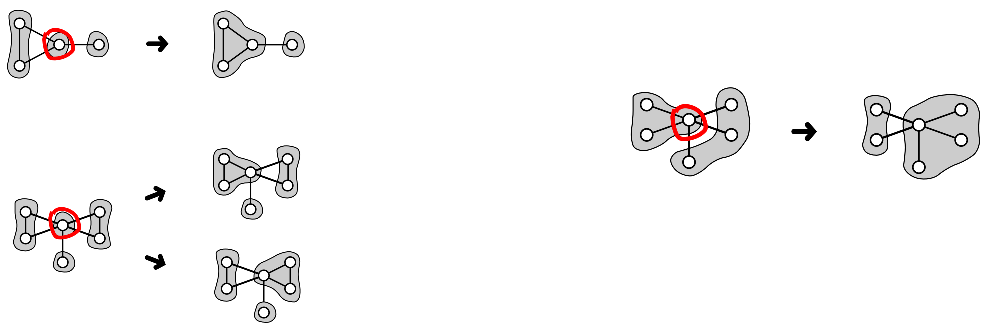

Figure 4. Illustration of Step 6+7 in MAJORCLUST: A definite majority decision (left top), and an indefinite decision (left bottom) when assigning a single node to a cluster. The right-hand side shows a situation where a node changes its cluster.

The runtime complexity of MAJORCLUST is $\Theta(|E| \cdot |C_{max}|)$, where $C_{max} \subseteq V$ designates a maximum cluster. Note that choosing a node $v \in V$ in Step (5) and choosing between clusters with the same attraction in Step (6) must happen randomly.

MAJORCLUST can be classified as non-hierarchical, exclusive cluster algorithm; it operationalizes an implicit density criterion and has the salient property that it automatically determines the number of clusters. MAJORCLUST finds a fast, but possibly suboptimal solution for the problem of Λ-maximization.

References

1. Art Systems Software Ltd.: 1998, ‘A Generic Method to Correct Orthographic Mistakes, which is Based on Sound’. Technical report, Art Systems Software Ltd., Paderborn.
2. Bailey, T. and J. Cowles: 1983, ‘Cluster Definition by the Optimization of Simple Measures’. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

3. Bezdek, J., W. Li, Y. Attikiouzel, and M. Windham: 1997, ‘A Geometric Approach to Cluster Validity for Normal Mixtures’. *Soft Computing 1*.
4. Bezdek, J. and N. Pal: 1995, ‘Cluster Validation with Generalized Dunn’s Indices’. In: N. Kasabov and G. Coghill (eds.): *Proceedings of the 2nd international two-stream conference on ANNES*. Piscataway, NJ, pp. 190–193.
5. Buja, A., D. Swayne, M. Littman, N. Dean, and H. Hofmann: 2001, ‘XGvis: Interactive Data Visualization with Multidimensional Scaling’. *Journal of Computational and Graphical Statistics*.
6. Chalmers, M.: 1993, ‘Using a Landscape Metaphor to Represent a Corpus of Documents’. In: *Proc. European Conference on Spatial Information Theory*, Vol. 716 of *LNCS*. pp. 377–390.
7. Dennis, S., P. Bruza, and R. McArthur: 2002, ‘Web searching: A process-oriented experimental study of three interactive search paradigms’. *JASIST* **53**(2), 120–133.
8. Ester, M., H. Kriegel, J. Sander, and X. Xu: 1996, ‘A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise’. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD96)*.
9. Everitt, B.: 1993, *Cluster Analysis*. New York, Toronto.
10. Fabrikant, S.: 2001, ‘Visualizing Region and Scale in Information Spaces’. In: *20th International Cartographic Conference*. Beijing, China, pp. 2522–2529.
11. Florek, K., J. Lukaszewiez, J. Perkal, H. Steinhaus, and S. Zubrzchi: 1951, ‘Sur la liason et la division des points d’un ensemble fini’. *Colloquium Methematicum* **2**.
12. Fogel, D. and L. Fogel: 1994, ‘Special Issue on Evolutionary Computation’. *IEEE Transaction of Neural Networks*.
13. Gamma, E., R. Helm, R. Johnson, and J. Vlissides: 1998, *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley Longman Publishing Co., Inc.
14. Geroimenko, V. and C. Chen: 2003, *Visualizing the Semantic Web*. Springer.
15. Gusfield, D.: 1997, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press.
16. Jain, A. and R. Dubes: 1990, *Algorithms for Clustering in Data*. Englewood Cliffs, NJ: Prentice Hall.
17. Johnson, S.: 1967, ‘Hierarchical clustering schemes’. *Psychometrika* **32**.
18. Kandogan, E.: 2001, ‘Visualizing multi-dimensional clusters, trends, and outliers using star coordinates’. In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 107–116.
19. Karypis, G., E. Han, and V. Kumar: 1999, ‘CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling’. Technical Report Paper No. 432, University of Minnesota, Minneapolis.
20. Kaufman, L. and P. Rousseuw: 1990, *Finding Groups in Data*. Wiley.
21. Klein, R. and R. Dubes: 1989, ‘Experiments in Projection and Clustering by Simulated Annealing’. *Pattern Recognition* **22**, 213–220.
22. Kohonen, T.: 1990, *Self Organization and Assoziative Memory*. Springer.
23. Kohonen, T., S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela: 2000, ‘Self organization of a massive document collection’. In: *IEEE Transactions on Neural Networks*, Vol. 11.
24. Kowalsky, G.: 1997, *Information Retrieval Systems—Theory and Implementation*. Kluwer Academic.
25. Larsen, B. and C. Aone: 1999, ‘Fast and Effective Text Mining Using Linear-time Document Clustering’. In: *Proceedings of the KDD-99 Workshop San Diego USA*. San Diego, CA, USA.
26. Lengauer, T.: 1990, *Combinatorial Algorithms for Integrated Circuit Layout*. New York: John Wiley & Sons.
27. Lewis, D.: 1994, ‘Reuters-21578 Text Categorization Test Collection’. `http://www.research.att.com/~lewis`.
28. MacQueen, J.: 1967, ‘Some Methods for Classification and Analysis of Multivariate Observations’. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. pp. 281–297.
29. Meyer zu Eißen, S. and B. Stein: 2002, ‘The AIsearch Meta Search Engine Prototype’. In: A. Basu and S. Dutta (eds.): *Proceedings of the 12th Workshop on Information Technology and Systems (WITS 02), Barcelona Spain*.
30. Nakazato, M., L. Manola, and T. S. Huang: 2002, ‘ImageGrouper: Search, Annotate and Organize Images by Groups’. In: *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems*. pp. 129–142.
31. Navarro, D.: 2001, ‘Spatial Visualization of Document Similarity’. Defence Human Factors Special Interest Group Meeting.
32. Norvig, P.: 1992, *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann Publishers Inc.
33. Pearl, J.: 1984, *Heuristics*. Addison-Wesley.

34. Porter, M.: 1980, ‘An Algorithm for Suffix Stripping’. *Program* **14**(3), 130–137.
35. Raghavan, V. and K. Birchand: 1979, ‘A Clustering Strategy Based on a Formalism of the Reproduction Process in a Natural System’. In: *Proceedings of the Second International Conference on Information Storage and Retrieval*. pp. 10–22.
36. Rijsbergen, C.: 1979, *Information Retrieval*. London: Buttersworth.
37. Rohrer, R., D. Ebert, and J. Sibert: 1998, ‘The Shape of Shakespeare: Visualizing Text using Implicit Surfaces’. In: *IEEE Symposium on Information Visualization*. North Carolina, USA, pp. 121–129.
38. Roxborough, T. and Arunabha: 1996, ‘Graph Clustering using Multiway Ratio Cut’. In: S. North (ed.): *Graph Drawing*. Springer.
39. Sablowski, R. and A. Frick: 1996, ‘Automatic Graph Clustering’. In: S. North (ed.): *Graph Drawing*. Springer.
40. Sabol, V., W. Kienreich, M. Granitzer, J. Becker, K. Tochtermann, and K. Andrews: 2002, ‘Applications of a Lightweight, Web-Based Retrieval, Clustering and Visualisation Framework’. In: *4th International Conference on Practical Aspects of Knowledge Management*, Vol. 2569 of *LNAI*. pp. 359–368.
41. Salton, G.: 1988, *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley.
42. Sneath, P.: 1957, ‘The application of computers to taxonomy’. *J. Gen. Microbiol.* **17**.
43. Song, M.: 1998, ‘BiblioMapper: A Cluster-based Information Visualization Technique’. In: *IEEE Symposium on Information Visualization*. North Carolina, USA, pp. 130–136.
44. Stein, B. and O. Niggemann: 1999, ‘On the Nature of Structure and its Identification’. In: P. Widmayer, G. Neyer, and S. Eidenbenz (eds.): *Graph-Theoretic Concepts in Computer Science*, Vol. 1665 LNCS of *Lecture Notes in Computer Science*. pp. 122–134.
45. Weippl, E.: 2001, ‘Visualizing Content-based Relations in Texts’. In: *Proceedings of the 2nd Australasian conference on User interface*. pp. 34–41.
46. Wu, Z. and R. Leahy: 1993, ‘An optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation’. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
47. Yan, J. and P. Hsiao: 1994, ‘A fuzzy clustering algorithm for graph bisection’. *Information Processing Letters* **52**.
48. Zahn, C.: 1971, ‘Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters’. *IEEE Transactions on computers* **C-20**(1).
49. Zamir, O. and O. Etzioni: 1998, ‘Web Document Clustering: A Feasibility Demonstration’. In: *SIGIR’98: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. University of Washington, Seattle, USA, pp. 46–54.
50. Zhao, Y. and G. Karypis: 2002, ‘Criterion Functions for Document Clustering: Experiments and Analysis’. Technical Report 01-40, Univercity of Minnesota, Department of Computer Science / Army HPC Research Center.