Proceedings of the Second International Workshop on "Text-Based Information Retrieval"

TIR-05

Koblenz, September 11, 2005

In Conjunction with the 28th Edition of the
Annual German Conference on Artificial Intelligence

Edited by

Benno Stein
Sven Meyer zu Eißen

Contents

Preface

Being in the age of information—so to speak: information flooding—intelligent technologies for information mining and retrieval have become an important as well as exciting field of research. In this connection, methods of text-based information retrieval receive special attention, which results from the fundamental role of written text, but also because of the high availability of the Internet. E.g., information retrieval methods have the potential to improve the quality of the standard keyword search; moreover, they strike a path to the new developments from the field of the Semantic Web. In this sense, text-based information retrieval can be regarded as the heart of many IR applications.

There are various techniques and methods being used for text-based information retrieval tasks, which stem from different research areas: machine learning algorithms, models from computer linguistics and psychology, paradigms from the field of user interaction and modeling, or algorithms for information visualization. The development of powerful retrieval tools requires the combination of these developments, and in this sense the workshop shall provide a platform that spans the different views and approaches.

The following list gives examples from classic and ongoing research topics from the field of text-based information retrieval: document models and similarity measures for special retrieval tasks, automatic category formation in document corpora and search results; topic identification and auto-abstracting: characterization of documents by keywords; plagiarism analysis: identification and evaluation of similar text sections; ontologies and the Semantic Web: analysis of annotated texts (RDF, RDFS, DAML+OIL), usage, construction, and maintenance of ontologies for retrieval tasks, OWL; cross-language retrieval, multilingual retrieval, and machine translation for IR; concepts and techniques for information visualization, user modeling, and interaction in the context of particular retrieval tasks; link analysis and Web dynamics; relevance feedback and personalization; evaluation, construction of test collections, and user studies.

Program Committee

Michael Granitzer, Know-Center Graz
Heiko Holzheuer, Lycos Europe
Andreas Hotho, University of Kassel
Wolfgang Kienreich, Know-Center Graz
Theodor Lettmann, University of Paderborn
Mathias Lux, Technical University Graz
Sven Meyer zu Eissen, Bauhaus University Weimar
Oliver Niggemann, dSPACE Paderborn
Tobias Scheffer, HU Berlin
Benno Stein, Bauhaus University Weimar

Workshop Organization

Benno Stein, Bauhaus University Weimar
Sven Meyer zu Eißen, Bauhaus University Weimar

Cluster Analysis of Railway Directory Inquire Dialogs*

Mikhail Alexandrov,[1,2] Emilio Sanchis,[2] Paolo Rosso[2]

[1] National Polytechnic Institute, Mexico
dyner1950@mail.ru
[2] Polytechnic University of Valencia, Spain
{esanchis, prosso@dsic.upv.es}

Abstract. Cluster analysis of dialogs with transport directory service allows revealing the typical scenarios of dialogs, which is useful for designing automatic dialog systems. We show how to parameterize dialogs and how to control the process of clustering. The parameters include both data of transport service and features of passenger's behavior. Control of clustering consists in manipulating the parameter's weights and checking stability of the results. This technique resembles Makagonov's approach to the analysis of dweller's complaints to city administration. We shortly describe the MajorClust method developped by Benno Stein's group and demonstrate its work on real person-to-person dialogs provided by Spanish railway service.

1 Introduction

In the recent years, much effort has been devoted to development of automatic dialog systems. This topic is well represented at the conferences related with Dialog Processing [6]. One of the first phases in the design of such systems consists in definition of the dialog domain, as well as the different types of dialogs to be conducted. The type of dialog depends on the information requested (which is reflected in lexical and semantic restrictions of the task), the type of user (novice or expert), etc. Usually this analysis is performed manually basing on a set of real person-to-person dialogs. Then the obtained classification is used to define the behavior of the dialog manager or to acquire a corpus of dialogs by using the Wizard of Oz technique [5].

In this paper, we consider another technique consisting in:

manual parameterization of dialog set;
filtering selected parameters;
objective clustering.

These steps correspond to the approach developed by Makagonov [3] for the analysis of letters and complaints of Moscow dwellers directed to Moscow city administration.

* Work done under partial support of the Government of Valencia, Mexican Government (CONACyT), R2D2 CICYT (TIC2003-07158-C04-03), and ICT EU-India (ALA/95/23/2003/077-054). The paper is a reprint of a paper published in Proceedings of TSD-2005.

The set of parameters is supposed to reflect both the specificity of a given public service and some features of a passenger. We show how to make correct scaling for quantitative or qualitative parameters of a dialog. Analysis of the parameter's distribution allows detecting and eliminating non-informative parameters. Clustering is applied both to dialogs themselves and to their parameters.

The procedure of clustering uses the MajorClust method recently developed by Stein *et al*. [7]. This method was selected because it proved to be more adequate to data and problem settings than other methods. We briefly describe it below.

In the experiments we used examples related with Spanish railway directory inquires, obtained in the framework of Basurde projects [1].

2 Parameterization

2.1 Examples of dialogs and their parameterization

The final purpose of dialog parameterization is to present all acts of dialogs in the form of the numerical matrix "objects/parameters", where the objects are dialogs themselves and the parameters are characteristics reflecting both railway service and passenger behavior. Such a matrix allows calculating the distance (similarity) between objects, which is the input data for any method of cluster analysis.

Table 1 shows the difficulties of parameterization (the records are translated from Spanish into English). Here *US* stands for a user and *DI* for a directory inquire service. This example concerns the train departure from Barcelona to the other destinations both near the Barcelona and in other provinces of Spain. Such limited dialogs constitute approximately 25% of total number of dialogs, other dialogs being 2 to 5 times longer. This dialog gives an impression on the difficulties of dialog parameterization.

Table 1. Example of real dialog between passengers and directory inqui80

DI: Renfe, good day
US: Yes, good day
DI: Yes, well.
US: OK, could you inform me about the trains that go from here, from Barcelona to Valladolid?
DI: What day it will be?
US: The next Thursday.
DI: Let us to see. <PAUSE> on Thursday is off the one at thirteen, which come at twenty two hours to Valladolid
US: Are there any more?
DI: No, on Thursday only this one, eh?
US: Nothing more, say me, please.
DI: Exactly.
US: <CONTINUALLY> before the Wednesday or Thursday.
DI: The train will be exactly at evening, on Thursday or Friday it is off.
US: Thank you, bye

One can note the following three features of such dialogs: many aspects concerning the trip are not reflected in a specific dialog; many characteristics are diffuse; and much information is presented in a hidden form. To take into account these circumstances, we use nominal scales with the value "indifference" and interval scales, respectively. All parameters are normalized to the interval [0, 1]. The parameters we initially introduced are presented below:

(1) City weight (*City*). This parameter reflects the economic importance of the city. Its possible values are 0.75, 0.5, 0.25, 0, reflecting large, middle, small, and local cities, respectively. The value 1 is reserved.

(2) Complexity (*Cx*). In our case, this binary parameter reflects the necessity of transfer.

(3) Urgency and definiteness (*U/D*). This numerical parameter is introduced to reflect the profile of passenger rather then the railway service. Its possible values are 1, 0.5, and 0: urgent departure at the same day, departure at a certain day during a week or month, and the indifference to the day of departure.

(4) Round trip (*T/F*). It is a binary parameter with obvious values 1 and 0.

(5) Time of departure (*T*). This parameter is presented in the form of three nominal scales with two binary values 1 and 0 for each of them: indifference to time (*Ti*), leaving in the morning or in the day (*Tm*), leaving in the evening or at night (*Te*).

(6) Time of departure on return (*F*). This parameter is similar to the previous one.

(7) Sleeping car (*Car*). It is a binary parameter.

(8) Discounts (*Ds*). It is a binary parameter meaning whether or not a passenger discussed his/her possible discount with directory inquire service.

(9) Knowledge (*Kn*). This parameter reflects any a priori information a passenger possesses about railway service. Values 1 and 0.5 mean the passenger wants to check up or refers to any previous information, respectively; otherwise, we use 0.

(10) Length of talking (*Tk*). This parameter can serve as an indicator of question complexity or the passenger's competence. It has five numerical values from 1 to 0 with step 0.25, which correspond to non-uniform scale of question-answer numbers.

(11) Politeness (*Pl*). We introduced formal rules for evaluation of this characteristic. Value 1 means that the passenger uses "you" in the polite form (Spanish distinguishes two degrees of polite treatment, reflected in different forms of "you" and verbal conjugation), "please", and apologetic style (reflected in the use of subjunctive forms in Spanish). Value 0.5 means a passenger uses "you" in polite form or in normal familiar form together with subjunctive forms; otherwise, we use 0.

Given these parameters, our example can be presented in a parameterized form, as shown in Table 2. Here *Ti-Tm-Te* and *Fi-Fm-Fe* are nominal scales for qualitative parameters (5) and (6).

Table 2. Parameterized example

City	***Cx***	***U/D***	***T/F***	***Ti-Tm-Te***	***Fi-Fm-Fe***	***Car***	***Kn***	***Ds***	***Tk***	***Pl***
0.25	0	0.5	0	0 0 1	1 0 0	0	0	0	0	0.5

2.2 Parameter filtering

The clustering procedure requires that the introduced parameters be filtered in order to reduce the influence of any strong dominant processes and any sources of possible noise. The former can hide the real structure we want to reveal and the latter can disfigure it. For this, all parameters are divided into the following three groups [3]:

Parameters from the first group have a significant value for almost all objects;

Parameters from the second group have a significant value for a small fraction of the total number of objects;

Parameters from the third group have a significant value for more than, say, 30% of the total number of objects.

By a significant value of a parameter, we generally mean a value larger than 50% of the maximum value for this parameter. By almost all objects or a small fraction of all objects, we mean 90%–95% and 5%–10% of the total number of objects, respectively.

From the point of view of the system, the parameters in the first group reflect the processes in a system of higher level in comparison with the one under consideration. The parameters in second group reflect the processes in a subsystem [3]. On the current level of data consideration, the mentioned groups of parameters should be eliminated.

This conclusion is supported by cluster analysis. From the point of view of cluster analysis, the first group of parameters is oriented to the uniform object set, i.e. to one cluster, whereas the second group of parameters is oriented to very granulated object set, at least 10 clusters or more [4]. The first situation is not interesting at all, and the second one is too detailed for reflecting the structure as a whole. Therefore, cluster analysis approach also confirms the necessity of eliminating of both groups of parameters.

To apply these results to our data set, we calculated the average value for each parameter; see Table 3.

Table 3. Average value of each parameter for 100 dialogs, in percents.

City	***Cx***	***U/D***	***T/F***	***To-Tm-Te***			***Fi-Fm-Fe***			***Car***	***Kn***	***Ds***	***Tk***	***Pl***
37	7	44.5	35	32	32	36	80	9	11	18	4	9	31	40

Since the maximum value of each parameter is equal to 1, we can easily select the parameters of the second group: *Cx, Fm, Fe, Kn, Ds*. For all these parameters, the number of significant values is less or equal to 10%. As for the first group, we decided to eliminate the parameter *Fi*, because its value is very close to the boundary value of 90% and from the other hand, this parameter is not interesting for interpretation: it means the indifference to the time of departure from the point of destination.

3 Clustering

3.1 Method of clustering

For a moment, there are dozens of methods and their modifications in cluster analysis, which can satisfy practically all necessities of users. The most popular ones are *K*-means, oriented on the structures of spherical form, and Nearest Neighbor (NN), oriented on the extended structures of chain form [2]. In our work, we use the MajorClust method, recently developed by Stein *et al*. [7], which has the following advantages over the mentioned two:

MajorClust distributes objects to clusters in such a way that the similarity of an object to the assigned cluster exceeds its similarity to any other cluster. This natural criterion provides the grouping of objects, which better corresponds to the users' intuitive representation. Neither *K*-means nor NN methods possess such optimization property: they do not evaluate the similarity between clusters.

MajorClust determines the number of clusters automatically and in all cases tends to reduce this number. *K*-means requires the number of cluster to be given, and NN does not determine this number at all: cutting of the dendrite is performed by the user.

MajorClust has been successfully used with various data sets and demonstrated very good results [8]. The main disadvantage of MajorClust is its runtime. However, in case of sparse matrix of relations this disadvantage is not essential. In our problem, we are faced just with this case because of a weak similarity between the majority of objects. These weak connections are eliminated, which gives the mentioned matrix.

3.2 Distance matrixes and manipulations with them

In our clustering procedure, we used two distance matrices: objects/objects (cities/cities) and parameters/parameters. To construct such matrices, we define the distance measure and apply it to the source objects/parameters matrix. It is well known that:

Cosine measure is used if the proportion between object's coordinates is important, but not their specific values. This is the case when the coordinates have the same meaning.

Euclidean measure is used when the contribution of each coordinate to object's properties is important. This is the case when the coordinates have different meaning.

Therefore, we used the cosine measure to evaluate the distance between parameters whose coordinates were cities, and Euclidean measure to evaluate the distance between objects (cities) whose coordinates were parameters.

During clustering procedure we changed the distance matrix:

To emphasize the role of certain objects (cities) while clustering parameters or certain parameters while clustering objects;

To reveal stronger but less numerous clusters;
To determine the stable number of clusters.

The first goal is reached by weighting the coordinates of objects or parameters, respectively. The second goal is achieved by eliminating weak connections between objects. At in the last case we vary the connections between objects and observe the changes of number of clusters.

4 Experiments

4.1 Experimental data

The data we used in the experiments were a corpus of 100 person-to-person dialogs of Spanish railway information service. The short characteristic of the corpus (length of talking, volume of lexis) is described in [1]. The data were analyzed in detail in [5] for constructing artificial dialogs.

4.2 Clustering parameters

Here in all experiments we used the cosine measure with the admissible level of connections not less than 0.7. In the first experiment all objects (cities) had no any privileges. In the second one the more important cities, that is the large and middle cities (see above) obtained the weight 5. It was the minimum weight, which allowed revealing new result.

<u>Experiment 1.</u> Two parameters *City Weight* and *Length of Talking* were joined to one cluster and the others remained the independent ones.

<u>Experiment 2.</u> Three parameters *City Weight*, *Urgency and Definiteness* and *Length of Talking* were joined to one cluster and the other parameters remained independent.

These results can be easily explained: the larger the city, the more possibilities to get it, the longer discussion a passenger needs. The urgency of trip is usually related with large cities: usually the trip to the small cities is completed without any hurry.

4.3 Clustering objects (dialogs)

Here in all experiments, we used Euclidean distance measure with the admissible level of connections not greater than 0.5 of the maximum. Then the distances were re-calculated to the similarity measure. In the first experiment, all parameters were equal. In the second experiment, we wanted to emphasize the properties of passengers. For this, we assigned the weight 2 to the parameters *Urgency and Definiteness, Length of Talking* and *Politeness.* This weight was the minimum one to obtain the significant differences with the first experiment. Parameters presented in nominal scales were weighted by the coefficient 0.33 that is inverse value to the number of scales. Cluster descriptions are presented below.

Experiment 1.

Cluster 1 (10 objects). The large and middle cities, no urgent trips (only 10%), round trips, night trips (70%-90%), sleeping cars, enough long talking.

Cluster 2 (25 objects). No urgent trips (only 8%), round trips, a few number of night trips (25%), no sleeping cars.

Cluster 3 (8 objects). Small cities (75%), undefined day of departure, one-way trips, night trips (90%), sleeping cars.

Cluster 4 (57 objects). Small or local cities (75%), one-way trips, no sleeping cars, short talking (80%).

Experiment 2.

Cluster 1 (31 objects). No urgent trips, no night trips (only 20%), only ordinary politeness.

Cluster 2 (44 objects). Urgent trips or defined days of trips (95%), advanced level of politeness (85%).

Cluster 3 (12 objects). Only small and middle cities, no urgent trips, one-way trips (75%), short talking (85%), the highest level of politeness.

Cluster 4 (13 objects). Only small and local cities, undefined days of trip, one-way trips (75%), no night trips (only 15%), short talking (75%), advanced level of politeness.

Some of the clusters were expected (e.g. the cluster 4 in both experiments) and the others need to be analyzed more closely. In all cases in comparison with manual classification where only costs and time-table were considered, our experiments gave the additional information [5]. Table 4 presents some examples of clustered objects.

Table 4. Examples of objects from cluster 3 in the experiment 2

City	*U/D*	*T/F*	*Ti*	*Tm*	*Te*	*Car*	*Tk*	*Pl*	*Name of city*
0.25	0.5	0	1	0	0	0	0.25	1	Girona
0.5	0.5	0	0	1	0	0	0.25	1	Alicante

5 Conclusions

Results The quality of automatic dialog systems used in public transport service crucially depends on the scenarios of dialogs. These scenarios may be determined by means of clustering in the space of parameters defined by an expert. We have shown (a) how to parameterize the records of dialog and to solve the problems of incompleteness and diffuseness of the source data; (b) how to accomplish the clustering procedure providing stability of results and their usefulness for a user. We have tested the MajorClust method for this and recommend using it for such type of problems. The obtained results were judged by experts as interesting and useful for determining the main themes of dialogs and the profile of passengers related with these themes. This information can be used to the design of scenarios for an acquisition of dialogs person-to-machine by means of the Wizard of Oz technique.

Future work In the future, we plan to consider more extensively the problems of Knowledge Discovery and to use both geographic information and the other parameters related with transport service.

Acknowledgement. The authors thank Sven Meyer zu Eissen for valuable discussion of application and tuning MajorClust, which allowed us to significantly improve our results.

References

1. Bonafonte, A., et. al.: Desarrollo de un sistema de dialogo oral en dominios restringidos. In: *I Jornadas en Tecnologia de Habla, Sevilla, Spain, 2000*
2. Hartigan, J.: Clustering Algorithms. Wiley, 1975.
3. Makagonov, P.: Evaluating the performance of city government: an analysis of letters by citizens to the Mayor by means of the Expert Assistent System. *Automatic Control and Computer Sciences*, Allerton Press, N-Y, vol. 31, N_3, 1997, pp. 11-19
4. Makagonov, P., Alexandrov, M., Sboychakov, K.: A toolkit for development of the domain-oriented dictionaries for structuring document flows. In: *Data Analysis, Classification, and Related Method*, Springer, series “Studies in classification, data analysis, and knowledge organization”, 2000, pp. 83-88
5. Sanchis, E., Garcia, F., Galiano, I., Segarra E.: Applying dialog constraints to the understanding process in a dialog system. In: *Proc. of TSD-02* (*“Text, Speech, Dialog”*), Springer, LNAI, N 2248, 2002, pp. 389-395
6. Sojka, P., et. al. (Eds.): Proceedings of Conf. Text, Speech, Dialog. Springer, LNAI, N_3206, 2004
7. Stein, B., Eissen, S. M. Document Categorization with MajorClust. In: *Proc. 12th Workshop on Information Technology and Systems*, Tech. Univ. of Barcelona, Spain, 2002, 6 pp.
8. Stein, B., Eissen, S. M. Automatic Document Categorization: Interpreting the Performance of Clustering Algorithms. In: *Proc. 26th German Conference on Artificial Intelligence (KI-2003)*, Springer, LNCS, N 2821, 2003, pp. 254-266

Fast LSI-based techniques for query expansion in text retrieval systems

Luigi Laura, Umberto Nanni, and Fabiano Sarracco

Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria, 113 - 00198 Roma Italy.
{laura,nanni,sarracco}@dis.uniroma1.it

Abstract. It is widely known that spectral techniques are very effective for document retrieval. Recently, a lot of effort has been spent by researchers to provide a formal mathematical explanation for this effectiveness [3]. Latent Semantic Indexing, in particular, is a text retrieval algorithm based on the spectral analysis of the occurrences of terms in text documents. Despite of its value in improving the quality of a text search, LSI has the drawback of an elevate response time, which makes it unsuitable for on-line search in large collections of documents (e.g., web search engines). In this paper we present two approaches aimed to combine the effectiveness of latent semantic analysis with the efficiency of text matching retrieval, through the technique of query expansion. We show that both approaches have relatively small computational cost and we provide experimental evidence of their ability to improve document retrieval.

1 Introduction

One of the most important tasks of an information retrieval system is to return, in response to a user query generally expressed as a set of terms, the subset of the managed documents which best matches the information necessity expressed by the user. Text matching is, beyond doubt, the most common technique used to accomplish this document retrieval task; the search is performed by looking for (and returning to user) the documents which contain the set, or a subset, of the terms included in the query. Two classical factors may negatively affect the quality of text matching search: *polysemy*, i.e., the same term may have different meanings (e.g., "web"), and *synonymy*, i.e., two distinct terms with the same meaning may be used interchangeably (e.g., "car" and "automobile"). In the former case, documents that are not relevant may be erroneously returned, while in the latter case, documents relevant to the user query may not be returned. To avoid these two and other similar problems, one would ideally like to represent documents (and queries) not by terms, but by the underlying (latent, hidden) concepts referred to by the terms. Unfortunately, as many works clearly point out, it is not possible to once-and-for-all define a universal terms-concepts

mapping structure, since this one heavily depends on the specific collection of documents we are dealing with.
Latent Semantic Indexing (LSI), introduced by Deerwester *et al.* in [5,6], is a technique to automatically compute from a documents collection a sort of semantic structure; this is achieved by applying a spectral decomposition technique, like the Singular Value Decomposition, on the data structure representing the occurrence of terms in documents. Documents (and queries) are represented and compared into this reduced "concept" space where, possibly, a document can be adjacent to (and thus relevant for) a query, even if the two do not have any term in common, but nevertheless share some underlying concept. LSI proved to be a very effective retrieval technique: it has been repeatedly reported that LSI outperforms classical text matching techniques, also addressing the problems of polysemy and synonymy[1,12]. This success has been empirically attributed to its ability to reduce noise, redundancy and ambiguity in the managed data structures. One of the still unsolved central questions on LSI is that the computational cost of the retrieval process strongly depends on the size of the collection. This makes LSI not appropriate for retrieving documents from huge collections, when we want to provide a fast online query answer.

Therefore we have on one side text matching, with fast response time but unsatisfactory filtering capabilities on the the document base, and on the other side LSI and similar techniques having powerful selection capabilities but high computational costs. In this paper we present two techniques that take advantage of the information provided by LSI data structures to enhance the quality of text matching search without increasing its response time. Both approaches process the user query by altering the set of terms included in it. We wish to remark that even if we present our approaches as a way to improve the effectiveness of text matching, the techniques introduced here, and more generally query expansion techniques, are fairly independent from the underlying retrieval algorithm used.
Our first technique, that we named *LS-Thesaurus*, is inspired by the work by Qiu and Frei [14]: they showed that the retrieval effectiveness of text matching can be improved by using a thesaurus, i.e., a structure indicating the similarity between each pair of terms. The idea is simple and natural: if terms in the query are relevant to the documents the user is looking for, these documents should contain also terms similar to the ones in the query. Consider the following example: suppose that the query "`football games`" is provided, and that this paper is included in the document collection. A text matching search would certainly include this document in the answer set but, obviously, this is not a document relevant to *football games*. However, if we expand the query by adding terms that are conceptually similar to the ones included in it, relevant documents are likely to be ranked higher. Our approach differs from the original work of Qiu and Frei because we make use of the data structures provided by the LSI technique, to generate the thesaurus.
LS-Filter, that is the second technique we present in this paper, can be intuitively described as a way to put in the query the more appropriate

terms for retrieving documents "conceptually related" to the query. To do this we project the query in the reduced concepts space generated by LSI; here we emphasize the largest components, that are the "important" concepts embedded in the query, and at the same time we set to zero the components with a small value, i.e., that are not relevant to the query. The modified vector is projected back in the terms space; our hope is that this new set of terms is better representative of the most important concepts in the query.
The rest of the paper is organized as follows: Section 2 provides the necessary background to understand our techniques, detailed respectively in Section 3 and Section 4. We present preliminary experimental results in Section 5, and final remarks are addressed in Section 6

2 Basic concepts and notation

In this section we define the preliminary concepts and techniques required to describe our main results. While doing this, we also fulfill the twofold task of fixing the notation used in the rest of the paper and presenting the relevant works in literature.

2.1 Term-document matrix and Text-Matching search

Consider a collection $D = \{d_1, \dots, d_n\}$ of n text documents. Let $T = \{t_1, \dots, t_m\}$ be the set of distinct index terms in D. The index terms are usually a subset of all the distinct terms occurring in the documents in D. Generally, this subset is composed only of words that are "relevant to retrieve the document" thus, for example, common words like articles and connectives are not considered as index terms. The *term-document* matrix of D is an $m \times n$ matrix A, representing a function of the occurrences of index terms in the documents in a compact way that can be efficiently managed and used by a text retrieval system. Matrix A has the following structure:

$$A = \begin{array}{c} \\ t_1 \\ t_2 \\ \vdots \\ t_m \end{array} \begin{array}{c} \begin{array}{cccc} d_1 & d_2 & \cdots & d_n \end{array} \\ \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \end{array} \tag{1}$$

where element $a_{i,j}$ is generally a real number indicating the relative weight of term t_i in document d_j. In its simplest form, A is a binary matrix where:

$$a_{i,j} = \begin{cases} 1 & \text{if term } t_i \text{ occurs in document } d_j \\ 0 & \text{otherwise} \end{cases}$$

In literature a vast set of more refined term-weighting schemes are shown to provide, at least for particular data sets, better retrieval performances.

In the rather general *tf-idf* (term frequency-inverse document frequency) weighting scheme, for instance, the value of element $a_{i,j}$ is a function of two orthogonal factors:

– A *local factor* (or *term frequency factor*) $L(i,j)$, measuring the influence (or relevance) of term t_i in document d_j. A commonly used formula for $L(i,j)$ is:

$$L(i,j) = \frac{freq(i,j)}{\max_{i \in [1,m]} freq(i,j)}$$

where $freq(i,j)$ is the frequency of term t_i in document d_j, i.e., the number of occurrences of t_i in d_j, divided by the total number of index terms in d_j.

– A *global factor* (or *inverse document frequency factor*) $G(i)$, whose aim is to de-amplify the relative weight of terms which are very frequently used in the collection. A commonly used formula for $G(i)$ is:

$$G(i) = \log \frac{n}{n_i}$$

where n_i is the number of documents containing term t_i.

See [15] for a more detailed description of the various term weighting schemes presented in literature.

While using a text retrieval system, the user submits a set of (possibly weighted) terms which, by his knowledge, best represent the information he is looking for. In our vectorial model, we can represent a user query as an m-dimensional vector $\boldsymbol{q} = (q_1, \ldots, q_m)$ (the *query vector*), where $q_i = w_i$ if term t_i is present in the query with weight w_i (or $q_i = 1$ if terms are not weighted), and $q_i = 0$ otherwise. In the following we assume $0 \leq q_i \leq 1$, for all $i \in [1,m]$.
In response to a user query, the system returns a score for each document in D, indicating its relevance to the query. Formally, the output of a search algorithm is an n-dimensional vector $\boldsymbol{r} = (r_1, \ldots, r_n)$ (the *rank vector*), where the value r_i measures the *relevance* of document d_i for the query expressed by $\boldsymbol{q}$. We assume that if $r_i > r_j$ then document d_i is more relevant than document d_j for the given query.

Let A be a term-document matrix representing a collection D of documents, and let $\boldsymbol{q}$ be a non zero query vector[1] given as input to the *text-matching search algorithm* (or simply TM). Algorithm TM returns as output the following rank vector:

$$\boldsymbol{r} = \boldsymbol{q} \cdot A$$

Note that A is a sparse (possibly binary) matrix and $\boldsymbol{q}$ is a sparse (possibly binary) vector. Therefore the matrix-vector product $\boldsymbol{q} \cdot A$ can be computed very efficiently. For instance, if, as commonly happens, the size of the query is limited to k terms, then Algorithm TM on a document base of n documents has cost $O(kn)$.

[1] Here we are assuming that at least one term of the query occurs in the collection

2.2 Query Expansion and Similarity Thesaurus

By *query expansion* or, more generally, *query reweighting*, we mean the process aimed to alter the weights, and possibly the terms, of a query. Usually, this process can be driven by the feedback provided by the user to the system (for instance, by selecting, among the retrieved documents, the ones that are considered more interesting). In this case the target is to refine a previously performed search. Indeed, a query can be reweighted also by exploiting a "knowledge", stored in the system, about terms usage in the particular collection of documents under exam. For instance, if in the collection of documents the term "automobile" is used often, then the system should add this term to a query containing only the term "car". In this paper we focus on this latter approach.
A commonly used method to have a statistic estimation of relationships among terms in a collection of documents, is to compute the term-term correlation matrix $A\,A^T$. If, for instance, we use the *tf-idf* weighting scheme, then the entries of $A\,A^T$ measure the co-occurence of terms in documents.
Qiu and Frei present in [14] an alternative approach to measure the term-term relation. Their idea is to compute the probability that a document is representive of a term. To do that, they propose the following weighting scheme:

$$a_{i,j} = \begin{cases} \dfrac{(0.5+0.5*\frac{freq(i,j)}{\max_j freq(i,j)})*itf(j)}{\sqrt{\sum_{l=1}^{n}((0.5+0.5*\frac{freq(i,l)}{\max_l freq(i,l)})*itf(j))^2}} & \text{if } freq(i,j) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where, as before, $freq(i,j)$ is the frequency of occurences of term t_i in document d_j and, for each term t_i, $\max_j freq(i,j)$ is the maximum frequency of term t_i over all the documents in the collection. Furthermore, let m_j the number of distinct terms in the document d_j, the ***inverse term frequency*** is defined as,

$$itf_j = \log \frac{m}{m_j} \quad (3)$$

In order to distinguish this particular term-document matrix from the ordinary one defined in Section 2.1, we denote it as $\overline{A}$.
A *similarity thesaurus* is a term-term correlation matrix S defined as:

$$S = \overline{A}\,\overline{A}^T.$$

A query vector $\boldsymbol{q}$ can be reweighted by using matrix S in the following way. First the vector $\boldsymbol{s} = \boldsymbol{q}S$ is computed. Then a threshold function $\xi(\boldsymbol{s}, x_r)$ is applied to $\boldsymbol{s}$. Assuming that $\boldsymbol{s} = (s_1, \ldots, s_m)$, function $\xi(\boldsymbol{s}, x_r)$ returns an m-dimensional vector $\boldsymbol{s}' = (s'_1, \ldots, s'_m)$ such that:

$$s'_i = \begin{cases} s_i & \text{if } s_i \text{ is among the } x_r \text{ largest (absolute) elements in } \boldsymbol{s} \\ 0 & \text{otherwise} \end{cases}$$

In other words, function ξ sets to zero all the elements of $\boldsymbol{s}$ which are not among the largest x_r elements, considering their absolute value. Note

that computing the function ξ requires time linear in the size of $\boldsymbol{s}$. The obtained vector $\boldsymbol{s}'$ is then normalized, by dividing all its elements by $|q| = \sum_{i=1}^{m} q_i$, thus obtaining a new vector $\boldsymbol{s}''$. Finally, the expanded query $\boldsymbol{q}'$ is computed as $\boldsymbol{q}' = \boldsymbol{q} + \boldsymbol{s}''$.
Query expansion techniques dates back to the work done by Maron and Kuhns in 1960 [9]. Automatic query expansion, despite the not encouraging results shown in the early work done by Lesk [8], Sparck Jones and Barber [16] and Minker *et al.* [10], where it seemed not an effective technique, has been revaluated after the researches made by Vorhees [18], by Crouch and Yang [4] and by Qiu and Frei [13].
The use of a thesaurus to improve the retrieval effectiveness has been introduced by Qiu and Frei in [14], where the authors use a global similarity thesaurus, and in [4] by Crouch and Yang. The difference in their approaches is on the type of the thesaurus: Qiu and Frei use a similarity thesaurus, i.e., a thesaurus based on the similarity between terms; the thesaurus proposed by Crouch and Young is statistical, i.e., is based on statistical co-occurences of the terms.

2.3 Latent Semantic Indexing

If we look at matrix A as a set of n distinct column vectors, we can interpret the elements of each column vector as the coordinates of the corresponding document in the m-dimensional subspace spanned by the terms in the collection. Intuitively, the more two documents (or a document and a query) are similar, the more the corresponding vectors are "close" to each other in the m-dimensional subspace. Polysemy and synonymy issues get their own interpretation in this context: along the same vector may lay two or more distinct meanings (polysemy), or, vice versa, the same concept may be spread along different vectors (synonymy).
The main idea behind *Latent Semantic Indexing* is to pre-compute a k-rank approximation of matrix A (with $k \ll m$), and to score documents according to their proximity to the given query vector in this low dimensional subspace. Experimentally, LSI turned out to be very effective in documents retrieval, overcoming, in most cases, the problems of polysemy and synonymy. A widely accepted intuitive explanation is that, more than terms, are the "latent semantic" concepts that should index the documents and the query.

As we said, LSI algorithm has a preprocessing offline phase. During this step the Singular Value Decomposition (or simply SVD) of A is computed. This is a way to rewrite A as a product of three matrices:

$$A = U \, \Sigma \, V^T$$

where:

U is a $m \times m$ orthonormal matrix [2], whose column vectors are the eigenvectors of $A\,A^T$;

[2] A square matrix M is orthonormal if is regular ($\det M \neq 0$) and if $M^T = M^{-1}$; so, for a orthonormal matrix M it holds $M^{-1} \cdot M = M^T \cdot M = I$.
Also $\det M = \pm 1$.

V is a $n \times n$ matrix whose column vectors are the eigenvectors of $A^T A$;
Σ is a $m \times n$ matrix made as follows:

$$\Sigma = \begin{bmatrix} \Sigma_r & 0 \\ 0 & 0 \end{bmatrix}$$

where $\Sigma_r = diag(\sigma_1, \ldots, \sigma_r)$. The values $\sigma_1, \ldots, \sigma_r$ are called *singular values*, are decreasingly ordered, and correspond to the square roots of the eigenvalues of $A\,A^T$. Note that r is the rank of matrix A, and obviously $r \leq n$.

We get a k-rank approximation A_k of A, by considering only the largest k singular values and fixing the remaining $(r-k)$ singular values to zero. Therefore,

$$A_k = U_k\, \Sigma_k\, V_k^T \tag{4}$$

where:

- $U_k = U^{(1:m,1:k)}$ is the $m \times k$ matrix composed by the first k columns of U; [3]
- $V_k = V^{(1:n,1:k)}$ is the $n \times k$ matrix composed by the first k columns of V;
- $\Sigma_k = \Sigma^{(1:k,1:k)}$ is the diagonal $k \times k$ matrix whose elements are the k largest singular values.

Note that A_k has still size $m \times n$. A well known theorem by Eckart and Young (see [7]) states that, among all the $m \times n$ matrices of rank at most k, A_k is the one that minimizes the distance from A, measured by the Frobenius norm.

The i-th row vector of V_k defines the components of document d_i in the k-dimensional subspace spanned by the column vectors of U_k. Observe that $V_k = A^T\, U_k\, \Sigma_k^{-1}$. When a user query $\boldsymbol{q}$ is given as input to algorithm LSI, the projection of $\boldsymbol{q}$ in the k dimensional subspace is computed as: $\boldsymbol{q}_k = q\, U_k\, \Sigma_k^{-1}$. Then the score of each document d_i is obtained by measuring the distance between the i-th row vector of V_k and the query vector $\boldsymbol{q}_k$. One way to do this is to compute the cosine of the angle between the two vectors:

$$r_i = \cos\left(\boldsymbol{q}_k, V_k^{(i:i,1:k)}\right)$$

where $V_k^{(i:i,1:k)}$ is the i-th row of V_k. The computational cost of LSI is thus $O(kn)$. Note that, unlike TM, the matrices V_k, U_k and the vector $\boldsymbol{q}_k$ are dense and therefore their product cannot be efficiently computed. This computational cost, which heavily depends on the size of the collection of documents, makes LSI unsuitable for online applications dealing with huge collections of documents, like, for instance, web search engines.

Papadimitriou *et al.* present a formal probabilistic analysis of the LSI as a IR tool in [12]. Azar *et al.* [1], in a more general context from Papadimitriou *et al.*, justify from a theoretical point of view both the empirical success of LSI and its effectiveness against the problem of polysemy.

[3] In this paper we use the colon notation to define submatrices: by $A^{(i:j,k:l)}$ (with $i \leq j$ and $k \leq l$) we denote the submatrix of A having element $a_{i,k}$ as the upper left element and $a_{j,l}$ as the lower right element.

Algorithm 1 Algorithm LS-Thesaurus Pre-Process

```
1: Input: a collection of documents $D$;
2: Output: a Similarity Thesaurus, i.e., a $m \times m$ matrix $S_k$;
3:
4: Compute the term-document matrix $\overline{A}$ from $D$;
5: $(U, \Lambda) \leftarrow EIGEN(\overline{A}\,\overline{A}^T)$;
6: $U_k \leftarrow U^{(1:m,1:k)}$;
7: $\Lambda_k \leftarrow \Lambda^{(1:k,1:k)}$;
8: $S_k \leftarrow U_k\, \Lambda_k\, U_k^T$;
```

Algorithm 2 Algorithm LS-Thesaurus Expand

```
1: Input: a query vector $\boldsymbol{q}$, a similarity thesaurus $S_k$,
2:     a positive integer $x_r$;
3: Output: a query vector $\boldsymbol{q'}$;
4:
5: $\boldsymbol{s} \leftarrow \boldsymbol{q}\, S_k$;
6: $\boldsymbol{s}' \leftarrow \xi(\boldsymbol{s}, x_r)$;
7: $\boldsymbol{s}'' \leftarrow \frac{\boldsymbol{s}'}{|\boldsymbol{q}|}$, where $|q| = \sum_{i=1}^{m} q_i$;
8: $\boldsymbol{q}' \leftarrow \boldsymbol{q} + \boldsymbol{s}''$;
```

3 LS-Thesaurus Algorithm

In this section the first of our two approaches is presented. The idea here is to generate a similarity thesaurus like the one described in Section 2.2. In this case, however, the similarity matrix S is computed starting from a low rank approximation of $\overline{A}$ (as in LSI). We can formally define our similarity matrix S_k in the following way:

$$S_k = \overline{A}_k\, \overline{A}_k^T = \overline{U}_k\, \overline{\Sigma}_k\, \overline{V}_k^T\, \overline{V}_k\, \overline{\Sigma}_k^T\, \overline{U}_k^T = \overline{U}_k\, \overline{\Sigma}_k^2\, \overline{U}_k^T \tag{5}$$

Note that $\overline{U}$ and the diagonal elements of $\overline{\Sigma}$ correspond respectively to the eigenvectors and the eigenvalues of matrix $\overline{A}\,\overline{A}^T$. Therefore, in the preprocessing step, we compute the eigenvalues and the eigenvectors of matrix $\overline{A}\,\overline{A}^T$ (through the function $EIGEN()$). We assume that the set of eigenvalues $\lambda_1, \ldots, \lambda_r$ is returned in the form of a diagonal matrix $\Lambda = diag(\lambda_1, \ldots, \lambda_r)$, while the eigenvectors are returned as the column vectors of a matrix U. The preprocessing step ends by returning the similarity matrix S_k computed as described in Equation (5). The query expansion is performed in the same way as in the original similarity thesaurus.

4 LS-Filter Algorithm

In this section we present our second query expansion technique, denoted as *LS-Filter*. Here we start from the following assumption. In the LSI algorithm documents and queries are projected (and compared) in

Algorithm 3 Algorithm LS-Filter Pre-Process

```
1: Input: a collection of documents $D$;
2: Output: a pair of matrices $(P, P^{-1})$;
3:
4: Compute the term-document matrix $A$ from $D$;
5: $(U, \Sigma, V) \leftarrow SVD(A)$;
6: $U_k \leftarrow U^{(1:m,1:k)}$;
7: $\Sigma_k \leftarrow \Sigma^{(1:k,1:k)}$;
8: $P \leftarrow \Sigma_k^{-1} U_k^T$;
9: $P^{-1} \leftarrow U_k \Sigma_k$;
```

Algorithm 4 Algorithm LS-Filter Expand

```
1: Input: a query vector $\boldsymbol{q}$, matrices $(P, P^{-1})$,
2:     two positive integers $x_c$ and $x_t$;
3: Output: a query vector $\boldsymbol{q'}$;
4:
5: $\boldsymbol{p} \leftarrow P\,\boldsymbol{q}$;
6: $\boldsymbol{p}' \leftarrow \xi(\boldsymbol{p}, x_c)$;
7: $\boldsymbol{p}'' \leftarrow P^{-1}\,\boldsymbol{p}'$;
8: $\boldsymbol{q}' \leftarrow \xi(\boldsymbol{p}'', x_t)$;
```

a k-dimensional subspace. The axes of this subspace represent the k most important concepts arising from the documents in the collection. The user query tries to catch one (or more) of these concepts by using an appropriate set of terms. However, it may happen that, due to user obliviousness or to intrinsic properties of the collection, the set of terms in the user query is not the best suitable for a text matching search. What algorithm *LS-Filter* tries to do is to "guess" the concepts the user is indicating with its query and to find the most suitable terms for retrieving, by a text matching search, the most relevant documents for these concepts.
This is achieved by projecting the query vector $\boldsymbol{q}$ into the reduced ("concepts") space generated by LSI (through a precomputed matrix $P = \Sigma_k^{-1} U_k^T$). The k-dimensional vector $\boldsymbol{p}$ we obtain measures the relation between the query and each of the k (latent) concepts contained in the collection. The algorithm filters it by setting to zero the elements having small absolute values (by using function ξ). The filtered vector $\boldsymbol{p}'$ is then projected back in the terms space, thus obtaining a vector $\boldsymbol{p}''$ which provides the (weighted) terms that are the counterimage of the concepts is the query. Finallly, we apply again function ξ, thus leaving in the final expanded query $\boldsymbol{q}'$ only the x_t terms having the largest weights.

5 Experimental results

In this section we present the results of the experimental studies we accomplished so far. We compared the behavior of the following approaches:

- TM: the simple text matching;
- LS-T: text matching with queries previously expanded by *LS-Thesaurus* algorithm;
- LS-F: text matching with queries previously expanded by *LS-Filter* algorithm;
- LSI: the full *LSI* computation as described in Section 2.3.

Dataset. Our data set are three books from O'Reilly (www.oreilly.com). They are publicly downloadable (in HTML version) from [11], and are an interesting (and difficult) dataset because they are quite specific (they all are related with computer science), and the correlation between them is high if compared to more heterogeneous collections of documents. We considered each HTML page as a different document, and indexed all the text of the page (excluding the tags). The overall number of documents is more than 3000, 1500 of which were from the first book, 1000 from the second book and the remaining 500 were from the last one. This collection is larger than the standard small datasets like CISI and MED [2] but its size allows a fast experimental setup, as opposite, for example, to the huge corpora provided for the TREC conferences [17].

Retrieval performance evaluation. To evaluate the effectiveness of these different IR techniques we used the standard precision versus recall diagrams at the eleven standard recall levels [2]. We briefly recall its definition.
Let $\boldsymbol{q}$ be a given user query. Let $R \subseteq D$ be the subset of documents which are relevant to $\boldsymbol{q}$ (usually determined by human experts), and let $|R|$ be the number of such documents. Moreover, we define A_h, for any integer $h \in [1, n]$, as the set of the h most relevant documents for $\boldsymbol{q}$ (we assume there are no ties), according to the rank vector returned by the system we are evaluating. Let $Ra_h = R \cap A_h$ be the intersection of the sets A_h and R. The recall and precision measures are defined as follows:

- $Recall_h$ is the ratio between the number of relevant documents in A_h and the total number of relevant documents:

$$Recall_h = \frac{|Ra_h|}{|R|}$$

- $Precision_h$ is the fraction of relevant document in A_h:

$$Precision_h = \frac{|Ra_h|}{|A_h|} = \frac{|Ra_h|}{h}$$

Note that the value of $Recall_h$ increases as the value of h increases (for $h = n$, $Recall_n = 1$).
We first compute the values of $Recall_h$ and $Precision_h$ for each $h = 1, \ldots, n$. Then we plot the precision-recall diagram by computing, through interpolation, the values of $Precision_h$, corresponding to the values of h for which $Recall_h = 0.0, 0.1, 0.2, \ldots 1$ (the standard recall levels).

Overview of the results. The precision-recall diagram for the tests we conducted, is plotted in Figure 1; we can observe that the techniques we propose performs between LSI and TM. It is interesting also to note

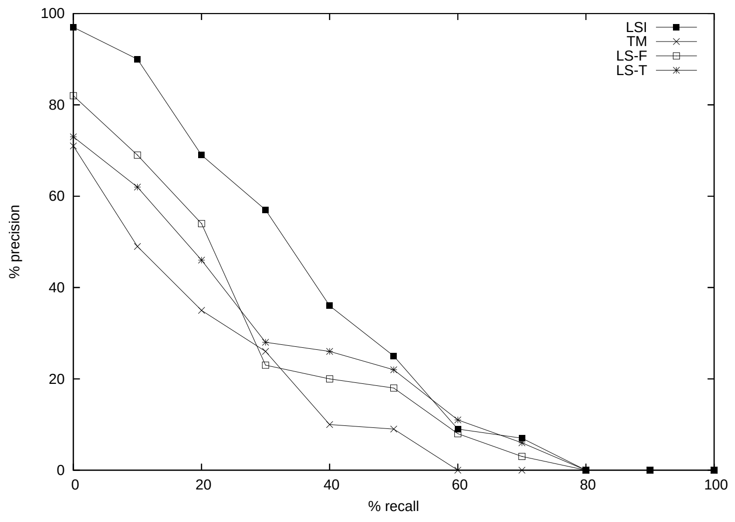

Fig. 1. Average precision at eleven standard recall levels

their relative performance: LS-T behaves better in the first half of the plot, and then is outperformed. To complete the picture, in Figure 2 is it shown the average of the times needed by each algorithm to respond a query, depending on the size of the collection. We notice that there are two different *y*-axis; the left one, whose range is from 0 to 60 seconds, measures the LSI while the right one, ranged 0 to 6 seconds, measures the other techniques (TM, LS-F and LS-Thaving roughly the same performances)[4]. The plot confirms at least one order of magnitude between LSI and the other techniques. These results indicate that the techniques we propose are a good trade-off between retrieval effectiveness and time performance.

Is it interesting to notice that, despite the (worst-case) running time of LS-T is obviously linear in the number of the terms in the collection (and definitely worse than LSI, linear in the number of documents), in practice the similarity thesaurus matrix is largely sparse (Qiu and Frei observed that usually around 3% of the entries are not zeroes). This means that every term can index its similar terms, and we can still provide a fast query answer. Consistently with the observation of Qiu and Frei our thesaurus has around 2,7% of non-zero entries.
We conclude by presenting, in Table 1, some examples of queries expanded by algorithm *LS-Filter*. We note that in some cases the system works as we would like to, for example when it adds the terms "bourne"

[4] We would like to point out that our IR system has been implemented to study and evaluate different techniques and therefore we focus only on the relative performances; in an optimized real IR system answer times can be significantly smaller.

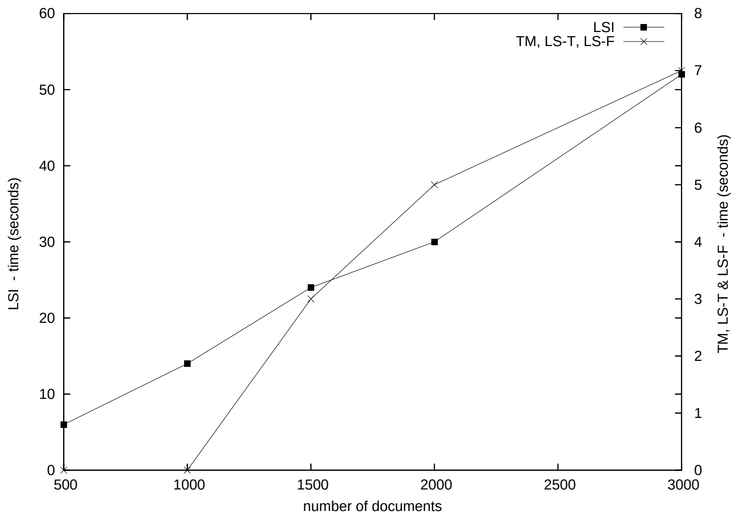

Fig. 2. Time comparison chart – left y-axis is for LSI, right y-axis for TM, LS-F and LS-T

and "prompt" to the query "shell logout"; sometimes, however, it adds misleading terms, like "ctrl" to the query "job control". We also see that the weight of the terms in the original query can be less than the one of the added terms; see for example the weight of term "control" in the query "job control".

6 Conclusions

In this paper we introduced two techniques, *LS-Thesaurus* and *LS-Filter*, that allow fast user query answer based on the LSI reduced space matrix. The preliminary experimental tests showed that our approaches perform well, and are a good trade-off between LSI and more simple methods like text matching. However, to evaluate the real effectiveness of both techniques we definitely need to perform more experiments on further data sets. We are working in this direction, and we are developing an open IR system that would allow to easily test different techniques.
We want to point out that the thesaurus generated according to *LS-Thesaurus* can be useful not only in the query expansion process, but also as a tool by itself. Moreover it can be used in an online style: the user can ask a query, see a list of similar terms and then he can decide to refine the query by adding some of these terms.
For the *LS-Filter*, we must remark its double behavior. In cases where the query terms correctly express the concept behind the query, this algorithm is able to outline and discard concepts of no interest, providing

java beans		job control		shell logout		zip file	
java	0.58136	job	0.42763	shell	0.68739	file	0.71548
bean	0.54392	background	0.12985	bourne	0.11818	util	0.16258
jdk	0.24954	echo	0.10754	login	0.09497	zip	0.15056
property	0.09444	number	0.09974	perl	0.09123	checksum	0.09776
nbsp	0.08534	list	0.07858	prompt	0.09851		
amp	0.07158	control	0.06929				
job	0.07119	ctrl	0.06882				
program	0.06206	object	0.06669				
value	0.05819	line	0.05827				
		filename	0.05827				

Table 1. Examples of query expanded by LS-Filter: user queries (first row) and the corresponding terms and weights.

more suited terms for the search. Sometimes (very often) query terms are still ambiguous and don't catch the real concept of interest. In these cases, we are not able to provide a correct retrieval; furthermore, the concept filtering can cut-off the concept of interest (with no high ranking), providing absolutely incorrect terms. The performances, in these cases, are vary bad, worse than simple text-matching. That is why in some cases the average graphics show for *LS-Filter* performances comparable to simple text-matching search.
This technique could be more effective if an appropriate user relevance feedback helps to discriminate relevant concepts in cases of ambiguous queries. In the system we are implementing, we want to provide the ability to select from the collection a set of terms with high discriminating power above concepts. In this way, every retrieved document is presented with all the discriminating terms it contains. Furthermore, everytime there is an ambiguity in the query among two or more concepts, the system is able to prompt the user for selection (/deselection) of the discriminating terms associated with ambiguous concepts. By selecting (/deselecting) one or more of these terms, or one or more of the retrieved documents, the user implicitly choices the concept of interest, allowing this way a more precise retrieval.

References

1. Y Azar, A. Fiat, A.R. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proc. of STOC 2001*, 2001.
2. R. Baeza-Yates and R. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.
3. H. Bast and D. Majumdar. Why spectral retrieval works. In *Prooceedings of ACM SIGIR*, 2005.
4. C.J. Crouch and B. Yang. Experiments in automatical statistical thesaurus construction. In *Proc. of the 15th ACM SIGIR Conference on Research and Development in Information Retrieval*, 1992.

5. S. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. *J.Soc.Inform.Sci.*, 41(6):391–407, 1990.
6. S.T. Dumais, G. Furnas, T.K. Landauer, and S. Deerwester. Using latent semantic analysis improve information retrieval. In *Proceedings of CHI'88: Conference Human Factors in Computing*, pages 281–285, 1988.
7. G. Golub and C. Reinsch. Handbook for Automatic Computation II, Linear Algebra. Springer-Verlag, New York, 1971.
8. M.E. Lesk. Word-word associations in document retrieval systems. *American Documentation*, 20(1):8–36, 1969.
9. M.E. Maron and J.L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Association for Computing Machinery*, 1960.
10. J. Minker, G.A. Wilson, and B.H. Zimmerman. An evaluation of query expansion by the addition of clustered terms for a document retrieval system. *Information Storage and Retrieval*, 8(6):329–348, 1972.
11. The O'reilly dataset. http://www.dis.uniroma1.it/~laura/.
12. C. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: A probabilistic analysis. *J. Comput. Systems Sciences*, 61(2):217–235, 2000.
13. Y. Qiu and H. Frei. Concept based query expansion. In *Proc. of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, 1993.
14. Y. Qiu and H. Frei. Improving the retrieval effectiveness by a similarity thesaurus. Technical Report 225, ETH Zurich, 1994.
15. G. Salton and C. Buckley. Term-weighting approaches in information retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
16. K. Sparck Jones and E.O. Barber. What makes an automatic keyword classification effective. *Journal of the American Society for Information Sciences*, 22(3):166–175, 1971.
17. Text REtrieval Conference. http://trec.nist.gov/.
18. E.M. Vorhees. *The Effectiveness and Efficiency of Agglomerative Hierarchic Clustering in Document Retrieval.* PhD thesis, Cornell University, 1986.

A Fast and Simple Path Index Based Retrieval Approach for Graph Based Semantic Descriptions

Mathias Lux, Michael Granitzer

[1]Institute for Knowledge Management and Visualization
Graz University of Technology and
Know-Center Graz, Austria
mathias.lux@tugraz.at

Know-Center Graz, Austria
[2]Competence Centre for Knowledge Based Applications and Systems R&D
mgrani@know-center.at

Abstract. The Semantic Web is a quite controversial concept, which is discussed extensively. Nevertheless most discussions showed that ways handling semantic metadata are needed urgently. Semantic metadata allows the storage of information which not only includes a set of concepts, but also the relations between the concepts in computable way. Within this paper an indexing and retrieval technique for semantic metadata is presented. The paper includes the discussion of a graph based data model for MPEG-7 based semantic metadata and an indexing technique for this model is outlined. Details on the implementation are given and a preliminary evaluation of the retrieval performance is included. As a last chapter related work is compared to our approach and pointers to related projects are given.

Introduction

While low level metadata of multimedia documents can be extracted automatically, high level and semantic metadata has to be created manually or semi-automatically. Current research projects aim to optimize the extraction of high level metadata and so to bridge the *semantic gap*, which identifies the inability of machines to gain understanding of the meaning of the data without human help (see also [DelBimbo1999], [Smeulders2000]). At present, semantic descriptions have to be created, at least in parts, manually. Human computer interaction (HCI) methods and information retrieval methods exist, that support the user in the annotation task. Different formats and approaches for creation and storage of semantic descriptions are currently discussed and in use, wherefrom MPEG-7 is one of them.

Given that the high level descriptions already exist, retrieval mechanisms for searching and browsing the multimedia content based on the high level descriptions have to be found. In case of textual descriptions and keywords the task can be simplified to text and keyword retrieval. Semantic descriptions, as they are used in this paper, consist of concepts and relations, which allow the expression of complex issues like for instance the fact “Mathias Lux is in Graz, which is in Austria” in a machine interpretable manner, leaving no room for misinterpretation. Semantic descriptions in our understanding belong to the area of knowledge representation and ontologies, which is described in short in the context of the Semantic Web in [Daconta 2003].

MPEG-7, called the *Multimedia Content Description Interface*, is, unlike MPEG-1, 2 and 4, no video or audio coding standard but a XML based standardized way to store annotations for multimedia documents (see e.g. [Martinez 2003]). MPEG-7 documents are built from descriptors, which are organized in descriptor schemes. One specific part of MPEG-7, the *Semantic Descriptor Scheme* (Semantic DS), allows the creation of semantic annotations based on a standardized and extendable ontology for annotation of multimedia content. Within the Semantic DS different *Semantic Descriptors* (Semantic Ds) are specified, which represent for instance agents, locations, events, time points or periods or concept. These Semantic Ds are interconnected pair wise with *Semantic Relations*, which are defined within the MPEG-7 standard (e.g. agentOf, patientOf, locationOf, etc.). The Semantic DS allows describing e.g. the content of a scene or an image like specifying: “The location of Mathias Lux is Graz, which in Austria”. An example of a visual representation of a semantic description is given in Fig. 4. MPEG-7 can be expressed either in XML or a binary representation optimized for transmission and storage. A comprehensive description of the Semantic DS and its usage within an application are given in [Martinez 2003]. The MPEG-7 standard and its applications are described in detail in [Kosch 2003].

While the MPEG-7 standard defines how semantic descriptions have to be stored and coded and how these definitions can be adopted and extended, it fails to define how to retrieve multimedia content using these semantic descriptions. Applying Semantic Web[1] ideas and methods the MPEG-7 based semantic annotations can be converted to RDF[2] (see [Hunter 2001] for details) and stored within a triple database, which allows querying the semantic descriptions with a RDF specific query language like SPARQL[3]. This allows data retrieval of the semantic descriptions (For more information on the Semantic Web see [Daconta 2003]). To realize information retrieval on semantic descriptions, featuring for instance retrieval and ranking of partially matching semantic annotation, above method is not applicable (for a detailed discussion on differences between information and data retrieval see [Baeza-Yates 1999]). Although with SPARQL a precise query using a complex query language can be defined, partial matches or result sorting based on relevance functions, which allow the ranking of the results of the query, are not supported. Ongoing work on searching ontologies and RDF based information is described at the end of this paper.

Within this paper an approach for an extended retrieval mechanism, based on standard information retrieval techniques, for MPEG-7 based semantic descriptions is given. The approach has been implemented within a prototype and extends a previous approach (see [Lux 2005]) in precision and recall. In the next chapter the method is

[1] http://www.w3.org/2001/sw/
[2] http://www.w3.org/RDF/
[3] http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/

described. The proposed method has been implemented within an open source project[4], details are given in section 3. The evaluation in section 4 is followed by a conclusion in section 5. An outlook on future developments and research task is given to close the paper.

Architecture

The main goal of this approach is to overcome the restrictions of inference used for searching semantic metadata by merging graph matching methods with current state of the art text methods. Thereby the benefits of semantic enriched descriptions and the performance of current retrieval information retrieval methods is combined. TF*IDF, which is short for Term Frequency - Inverse Document Frequency, is a method for term weighting in text document indices to enhance retrieval performance while inverted lists or files allow fast term based retrieval of text documents (see [Baeza-Yates 1999] or chapter 14 in [Hand 2001] for details).

The input for the retrieval process is a semantic description, given by the user. The output lists all relevant semantic descriptions in the database sorted by their relevance compared to the query. To achieve these goals a mathematical and data model for the semantic descriptions had been built and a retrieval strategy had been created.

The Model of the MPEG-7 Semantic Description Scheme

All semantic descriptions consist of nodes, which are semantic descriptors extended from the semantic base descriptor, and relations, which interconnect two different nodes. The MPEG-7 Semantic DS can be seen as directed graph, whereas the nodes are the vertices and the relations are the directed edges. The graph is not necessarily connected, as relations are not mandatory. As all the nodes and relations are identified by descriptors, a semantic description is a labeled graph, whereas the MPEG-7 descriptors are the labels for the edges and vertices. Screenshots of visual representations are given in Figure 1 and Figure 5.

For the definitions of graphs, directed graphs (digraphs) and labelled graphs see [Diestel 2000] or [Tittmann 2003]. For the sake of simplicity two nodes cannot be connected through two different relations and a relation cannot have one single node as start and end node. In graphs based on semantic DS no two nodes can have the same label (the same descriptor), so the node labels are unique. Each directed edge can be inverted as there exists an inverse of each MPEG-7 based semantic relation.

[4] Caliph & Emir is available at sourceforge.net: http://caliph-emir.sourceforge.net

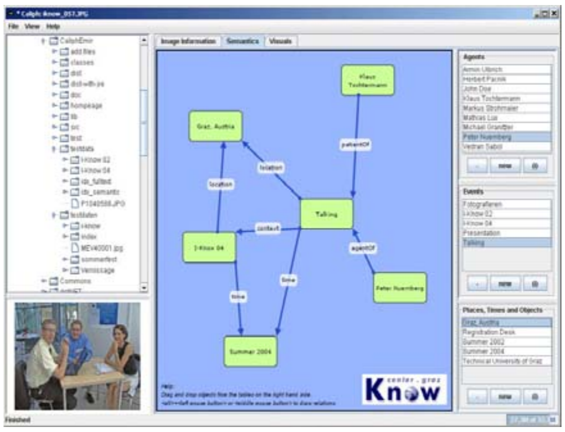

Fig. 1. Example of the representation of a MPEG-7 based semantic description. The green boxes represent the semantic objects, while the blue labeled arrows symbolize the relations of the description.

The Term Space for the MPEG-7 Semantic DS

Based on the identified constraints of a directed labelled graph with unique node labels and edges, that can be inverted, an efficient retrieval strategy can be designed as follows: The idea of fast retrieval of graph based structures is not new, as the contribution of Simmons in 1966 (see [Simmons 1966]) shows. Common retrieval techniques for graphs are the usage of metrics like the maximum common subgraph metric, or the graph edit distance. A straight forward implementation using this distance measures results in search time *O(n)*, whereas *n* defines the number of graphs in the database. Please note that the distance or similarity calculation between two graphs is NP-hard in respect to the number of nodes and edges of the graphs to compare (see e.g. [Valiente 2002]). Another approach is the filtering of the database with a fast (less than linear search time) algorithm and the ranking of the results with a slower metric which is described in chapter 12 in [Baeza-Yates 1999]. This method has been successfully used for graphs e.g. in [Fonseca 2004] for clipart retrieval by using graph eigenvalues as filters like in [Shokoufandeh 1999]. A more promising approach for MPEG-7 semantic DS, if the usage of an existing text search engine is constraint, is the usage of a path index (see e.g. the description of GraphGrep or the Daylight database in [Shasha 2002]). A path index allows the fast retrieval of graphs based on paths (sequences of nodes connected by edges, whereas the number of edges defines the length o the path) of different lengths extracted from the graphs. The extracted paths can be interpreted as index terms for a graph.

The graph can be expressed using all paths, which are sequences of nodes interconnected by edges, of chosen length. Paths of length 0 are the nodes themselves while paths of length 1 are triples as used in RDF. Paths of length 0 and length 1 have unique string representations for MPEG-7 based semantic descriptions as shown in

[Lux 2005]. To allow the usage of wildcard nodes at least the paths of length 2 have to be used, for which a unique string representation can be defined as shown below. The graph can be stored using the paths of length 0, 1 and 2 as index terms. Using a query graph all paths of the query graph are extracted and used as search terms. The ranking is done by TF*IDF on the index terms, which are the paths of the graphs.

Implementation

For the implementation an existing open source tool, called Emir (from Experimental Metadata based Image Retrieval), has been extended. Together with the annotation tool Caliph (Common And LIght-weight PHoto annotation) Emir allows the organisation, annotation and retrieval of digital photos based on MPEG-7 documents. Emir features content based and keyword based retrieval of images annotated by Caliph. Based on the open source retrieval engine *Lucene*[5] an index for node descriptors has been implemented in a previous project (see [Lux 2005]), where the string representations of paths of length 0 and 1 also have been implemented. As the query graph can consist of query strings for the node values, query expansion based on the node descriptors is used as described in [Lux 2005]. All path representations are constructed from node IDs, which identify a unique node descriptor in the index, and relation names or wildcards for nodes or relations. For the usage for terms within Lucene the path representations were adopted: all white spaces were replaced by '_' and all paths start with a leading '_'. The leading '_' allows the usage of wildcards at the start of a path expression.

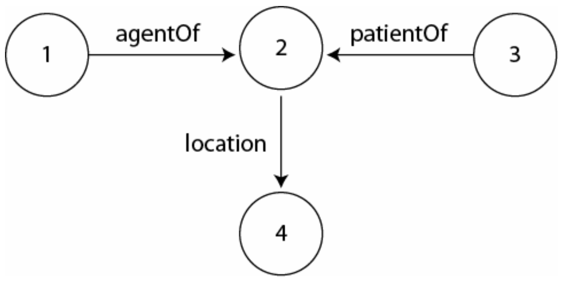

Fig. 2. Example for a graph following the model of MPEG-7 semantic DS graph. Node descriptors are already substituted with node IDs. Only the node IDs, which point to the node descriptors, are shown.

For the graph given in Figure 2 the terms for paths of length 0 and 1 would for example be:

[5] http://lucene.apache.org

Table 1. Extracted path terms of length 0 and 1 from graph shown in Fig 1. Note that the path _locationOf_4_2 has been inverted. This is done to *normalize* the edge directions in the index.

Term	**Path length**
_1	0
_2	0
_3	0
_4	0
_agentOf_1_2	1
_locationOf_4_2	1
_patientOf_3_2	1

For the creation of terms from paths of length 2 following method has been introduced: The input of the method is either a graph representing a semantic DS or a query graph. In a first step all paths of length 2 are extracted from the graph (see [Valiente 2002] for details on the algorithms). For each of these extracted paths the unique string representation has to be created as follows:

1. Compare the start node of the path with the end node of the path
2. If the start node is bigger than the end node reverse the path:
   a. Switch end and start node
   b. Switch and invert first and second relation
3. Create string in order: start node – first relation – middle node – second relation – end node with '_' as separator.
4. Prepend '_' to the string.

This results for the graph shown in Fig. 1 in following additional path terms:

Table 2. This table shows all available extracted path terms of length 2 from the graph shown in Figure 2.

Term	**Path length**
_1_agentOf_2_patient_3	2
_1_agentOf_2_location_4	2
_3_patientOf_2_location_4	2

All these above shown terms are used to index the semantic description with Lucene, all terms are used as Lucene tokens without stemming or other pre-processing. For queries the terms are constructed in a similar manner with one exception: Wildcards for nodes and relations can be used. For relations the adoption is straightforward: As Lucene supports wildcard queries for a wildcard relation the String '*' is inserted instead of the relation name, e.g. _*_1_2 instead of _agentOf_1_2. To support undirected wildcard relations two relation query terms are constructed and combined with a Boolean OR, like (_*_1_2 OR _*_2_1) . For paths of length 2 only the '*' is inserted instead of the relation name as the order of the path only depends on the start and end node.

For nodes in paths of length 0 the query string is omitted. For paths of length 1 and middle nodes in paths of length 2 the node ID is replaced with a '*'. For start and end nodes in paths of length 2 a Boolean query clause has to be constructed as the order of the start and end node cannot be used to identify the term representation, e.g. (_*_patientOf_2_location_4 OR 4_locationOf_2_patient_*). Note that the relations have to be inverted in this case.

A simple example for a wildcard query would be: "Find all semantic descriptions where *Mathias Lux* is *doing something* at the *I-Know*". In a first step possible candidates for nodes are identified to construct the query graphs. Assuming that for *Mathias Lux* the node with ID 28 has been found, while for *I-Know* the node with ID 93 has been found, the query graph would look like "[28] [93] [*] [agentOf 1 3] [locationOf 3 2]". The numbers within the relations reference the node using their position in the node list. Such a query would result in a query like "_28 _93 _agentOf_28_* _locationOf_*_93 _28_agentOf_*_locationOf_93".

Note that arbitrary methods, which are not restricted to text retrieval, could be used to identify candidate nodes in this approach. Methods different from term based text retrieval were not implemented in this state of the project but possible mechanisms include multimedia retrieval like content based image retrieval, or Latent Semantic Indexing.

Evaluation

The data repository, which was used for evaluation, consists of 85 different semantic descriptions of digital photos from two different scientific conferences, the I-Know 02 and the I-Know 04.

Table 3. This table summarizes the size of the graphs of the test data set. In the description 46 different semantic objects (names, locations, events, etc.) were used.

	Min	**Max**	**Median**
Nodes	3	11	5.5
Relations	2	12	5.6

All descriptions consist of a minimum of 3 nodes up to 11 nodes with a median of 5.5 nodes and 2 to 12 relations with a median of 5.6. Each of these descriptions was taken as query input for the evaluation and the average precision at 11 standard recall levels was calculated. The test set was generated by taking the query graph, and ranking all graphs based on the maximum common subgraph distance. The maximum common subgraph distance, formally defined and proved as metric in [Bunke 1998], was chosen, because it is a metric that takes structure as well as content into account and it is a good representative for a group of similar metrics including the graph edit distance and the minimum common supergraph metric The main idea of this metric is to compare the size of the maximum common subgraph *mcs(G1, G2)* to the maximum of the size of the Graphs *G1* and *G2* as shown in (1) (see also [Bunke 1997]).

$$similarity(G_1,G_2) = \frac{|mcs(G_1,G_2)|}{\max(|G_1|,|G_2|)} \qquad \textbf{(1)}$$

The best ten ranked documents made up the test set and precision and recall for the query results based on the path index and the full text index were calculated. For the full text index search the node IDs were converted back to the node labels, which were concatenated to a query string. The full text index itself contains all non-tag strings of the MPEG-7 XML document, including all semantic object labels, descriptions, etc.

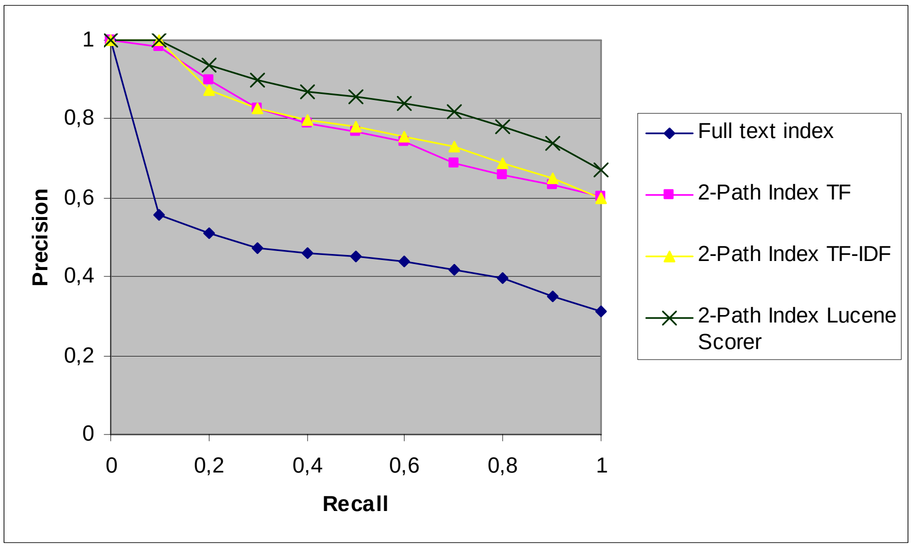

Fig. 3. Average precision at 11 standard recall levels using a test set generated with the maximum common subgraph metric. The lowest graph shows the performance of the full text search compared maximum common subgraph metric, while the other three graphs show the performance of path index based method using different term weighting schemes.

As can be seen easily the usage of the path index gives a better retrieval performance compared to the full text index for a test set generated with the maximum common subgraph distance. This can be easily explained as the full text index cannot take the structure of the semantic descriptions with its relations and paths into account, while the path index, to a limited account, can do this.

Experiments with different relevance functions, which are called “scoring functions” in Lucene, revealed that there is an obvious difference between the built-in Lucene Scorer and a straight forward classical TF*IDF implementation.

$$score(q,d) = \sum_{t \in q} TF(t,d) \cdot IDF(t) \cdot b(t.field,d) \cdot lNorm(t.field,d) \cdot coord(q,d) \cdot qNorm(q) \qquad (2)$$

The Lucene scoring function, shown in (2), can be explained as follows: *TF(t, d)* calculates the term frequency of term *t* in document *d* by using the square root of the term count, *IDF(t)* the inverse document frequency of the term defined by *log(numDocs/(docFreq+1)) + 1*. Function *b(t.field, d)* takes the boost factor for the specific field into account, while *lNorm(t.field, d)* penalizes longer fields with much content. With the function *coord(q, d)* documents matching more terms of query *q* are scored higher, while *qNorm(q)* tries to normalize the score based on the length of the query.

The tested term weighting schemes used in the valuation were taken from the Lucene implementation. For the classical TF*IDF weighting the score function shown in (2) was modified by removing all factors inside the sum apart from *TF(t, d)* and *IDF(t),* for the TF weigthing scheme the *IDF(t)* factor was removed too.

The Lucene scoring function outperforms the classical TF*IDF implementation and the term frequency scoring function. We assume that the *coord(q, d)* factor is the reason for the different performance of the classical TF*IDF and the Lucene score function by reflecting the denominator of the maximum common distance metric. Between classical TF*IDF implementation and the term frequency scoring function only slight differences in retrieval performance can be identified.

Conclusion

Comparing the path index with the full text index based approach and taking a look at the precision and recall it can be easily seen, that the resulting relevance function is more similar to the maximum common subgraph metric than the relevance function based on terms in the full text index, which allows the retrieval of descriptions that have a common sub-description with the query description. The adopted TF*IDF based scoring function of Lucene proves as best choice for calculating relevance of matching documents. The approach based on triples of 1-path indices stored in a file for retrieval with regular expressions, as presented in [Lux 2005], has in comparison to the method presented in this paper only limited use. While the 2-path index based technique allows the retrieval of model graphs, which have a common subgraph with the query graph, the approach in [Lux 2005] only allows retrieval of model graphs, where the query is a subgraph of the model graph.

One of the benefits of the presented method is the expected increased runtime performance. This is because of the term index based retrieval compared to the linear search although the worst case, where a term (or path in this case) is part of each graph in the database, also has runtime linear in the number of graphs. Linear search using the maximum common subgraph metric is rather slow as the problem of subgraph isomorphism is in NP (see e.g. [Valiente 2002]). Additional benefit is the increased focus on the graph structure of the semantic descriptors, which increases the precision compared to the full text index approach. The support for precise search with wildcard nodes is limited to one wildcard node in a 2-path:

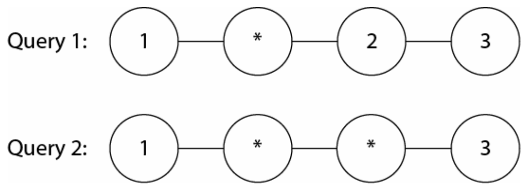

Fig. 4. Two examples for a graphical query, where '*' denotes a wildcard node. In Query 1 there is only one wildcard, while in Query 2 two wildcards in a sequence occur.

Following example shows the problem with two wildcards in a row based on the queries shown in Figure 4: For Query 1 in the retrieval of structures similar to the one shown above can be guaranteed, as the constructed query terms “_1_relation_*_relation_2” and “_*_relation_2_relation_3” define Query 1 unmistakeable. For Query 2 the terms “_1_relation_*_relation_*” and “_*_relation_*_relation_3” could also match a graph with only three nodes like the one identified by term “_1_relation_2_relation_3”.

Due to the usage of Lucene as search engine the retrieval is fast and stable and the implementation effort could be minimized as TF*IDF and inverted lists were not implemented for this project.

The usage of the query expansion mechanism for generation of the query graphs resulted in performance problems with very unspecific queries as reported in [Lux 2005]. Within this extension a solution based on query refinement was integrated: For each node defined by a query string a list of possible candidate nodes is shown. A selection of one candidate node is possible to reduce the number of expanded query graphs.

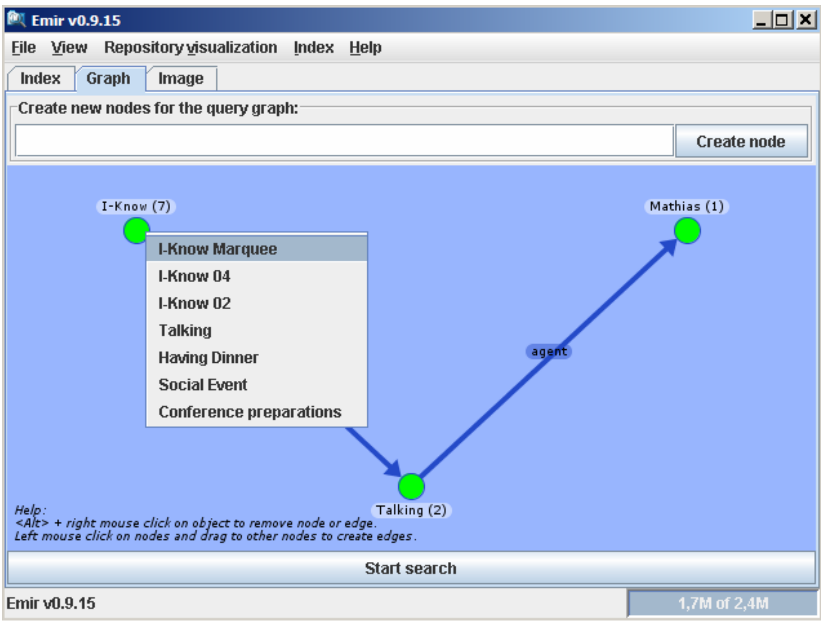
Emir v0.9.15
File View Repository visualization Index Help
Index Graph Image
Create new nodes for the query graph:
Create node
I-Know (7)
Mathias (1)
I-Know Marquee
I-Know 04
I-Know 02
Talking
Having Dinner
Social Event
Conference preparations
agent
Talking (2)
Help:
<Alt> + right mouse click on object to remove node or edge.
Left mouse click on nodes and drag to other nodes to create edges.
Start search
Emir v0.9.15
1,7M of 2,4M

Fig. 5. Integration of the selection of possible candidates for a node to reduce the number of expanded query graphs. With a mouse click a context menu appears which allows the selection of a specific node.

While the original query of Figure 5 would result after expansion in 14 query graphs, a selection of one of the nodes in the context menu would reduce the number of query graphs to 2.

Future Work

Above evaluation has shown that the usage of a 2-path index improves retrieval performance. Another interesting aspect to evaluate is the general performance for differing sizes of paths, longer than 2 edges. Another evaluation task will be to create test sets based on wildcard node queries and define result sets with an error correcting maximum common subgraph (mcs) distance measure as described in [Berretti 2004]. This error correcting maximum common subgraph distance allows the integration of node and edge distances and optimizes the distance between two graphs to a minimum. This method is currently used in the area of content based image retrieval, but can be adapted to the retrieval of semantic metadata easily provided that node and edge distance functions exist.

The technique introduced within this paper can also be applied to other graph based representations of semantic information like it is done in the Semantic Web standards RDF and OWL[6]. Further adoptions to standards and scenarios that aren't based on MPEG-7 are currently issue of research.

Another promising idea is the usage of the suffix tree retrieval model (see [Meyer zu Eissen 2005] for details) instead of the vector space model. Retrieval with the suffix tree model using the paths of a graph would allow the efficient search for graphs, which have paths in common. This allows the combination of content and structure, like it is done by sophisticated graph matching algorithms like error correcting maximum common subgraph distance of Berretti et. al.

State of the Art Graph based and Semantic Web Retrieval

In contrast to the presented approach inference has been identified in [Carroll 2002] as main tool for retrieval in the Semantic Web. Examples for inference based queries are "Show me every photo showing a person is living in Graz" or "Play a rock concert recorded in Graz, where at least one rock band member has long hair". All facts are investigated and all matching true facts fulfilling all given constraints are returned. This is a Boolean concept as there is no grade of falseness or truth. Although it was tried to show that distance calculation between graphs based on graph isomorphism, which is also called graph matching, does not prove useful for RDF, recent work shows possible applications for MPEG-7 (see Lux and Granitzer in [Lux 2005]) and conceptual graphs (see [Zhong 2002], [Zhu 2002] and [Yang 1993]). As Corby, Dieng and Hubert showed in [Corby 2000] that there is a bijective transformation between conceptual graphs and the RDF/XML serialization of the RDF Model the results of Zhong et. al. can be applied to RDF as well. Arguments against inference in semantic search engines are summarized in [Alesso 2004]. Main points of critique are the incompleteness and the halting problem for large ontologies.

Search engines for the Semantic Web, which do not rely on inference, have been already created using different approaches. In [Rocha 2004] a retrieval system, which allows the retrieval of resources in an ontology by spreading activation, was presented. The main flaw of this approach is that an overall ontology has to be built, which implies that many small heterogeneous ontologies have to be mediated. The OntoSeek system, described in [Guarino 1999], implements query expansion of search queries based on ontologies. The actual retrieval process does no take relations into account, but relies on terms identifying concepts. The ranking of the results is done using a graph similarity metric. The metadata search engine Swoogle (see [Ding 2004]) harvests and indexes RDF based metadata and ontologies and allows retrieval based on the literals used in the indexed RDF data. Ranking of the results is based on the in and out degrees of the found nodes and concepts, similar to the PageRank algorithm (for details on PageRank see e.g. [Rogers 2002]).

In [Stojanovic 2003] a ranking mechanism for matching expressions resulting from inference is given. The query is formulated in an ontology query language, matching instances within an ontology driven knowledge base are returned. The approach is based on node distance in hierarchical structures or in other words on a tree edit

[6] http://www.w3.org/TR/owl-features/

distance measure. To create a query a user has to acquire knowledge upon the underlying ontology. As though this approach could be used to solve a part of the problems solved with our method several differences can be stated: In comparison to the technique presented in this paper the query formulation heavily depends on the ontological definition of the conceptual hierarchy of the knowledge base. The querying mechanism focuses on the retrieval of concept fulfilling different aspects defined in the query, while the technique presented in this paper aims to retrieve graph based structures similar to the one a user specified. Unlike the approach in of Stojanovic et. al. the approach presented in this paper allows the retrieval using a not connected graph (a non empty graph *G* is called *connected* if any two vertices are linked by a path in *G*). Furthermore our approach is performance optimized as it uses adopted standard approved text retrieval methods.

In [Shasha 2002] *GraphGrep* was introduced, an approach for indexing undirected graphs with node labels, which are unique within a single graphs. The approach can be easily adopted to directed graphs with labelled nodes and edges and uses a path index of variable length, which means in this case that the maximum length of the indexed paths can be given at indexing time. However this approach does not allow wildcard queries, was only used for database filtering and does not give a relevance function for the retrieved graphs. It has not been tested upon retrieval performance and size of the index.

[Yan 2004] introduces a system for the retrieval of chemical structures, which is capable of indexing graphs based on their paths. The decision upon which paths are integrated in the index is made using a graph structure frequency measure. This frequency measure implements a TF*IDF variant and allows the indexing of paths, which are longer and more significant instead of many less significant paths, which are omitted for indexing. Unfortunately the authors did not evaluate the retrieval performance using precision and recall, although their approach is compared to GraphGrep. However a comparison with our approach would be inappropprioate: The use case of retrieving chemical structures is quite different to the use case of retrieval of MPEG-7 based semantic descriptions: Within molecules node labels are not unique, e.g. paths of different lengths consisting of multiple carbon atoms can occur, for instance in benzene hexachloride.

Acknowledgements

The Know-Center is a Competence Center funded within the Austrian Competence Center program K plus under the auspices of the Austrian Ministry of Transport, Innovation and Technology (www.kplus.at).

References

[Alesso 2004] Alesso, H. Peter, "Semantic Search Technology", SIGSEMIS: Semantic Web and Information Systems, http://www.sigsemis.org/columns/swsearch/SSE1104, last visited 15th July 2005

[Baeza-Yates 1999] Baeza-Yates, R. , Ribeiro-Neto, B., "Modern Information Retrieval", ACM Press and Addision-Wesley, 1999

[Berretti 2004] Berretti, S., Del Bimbo, A., Pala, P. A, "Graph Edit Distance Based on Node Merging", in Proceedings Image and Video Retrieval: Third International Conference, CIVR, Springer, LNCS 3115, pp. 464-472, 2004

[Bunke 1997] Bunke, Horst, "On a relation between graph edit distance and maximum common subgraph", Pattern Recognition Letters, Vol. 18, Num. 9, 1997, pp. 689-694

[Bunke 1998] Bunke, Horst, Shearer, Kim, "A graph distance metric based on the maximal common subgraph", Pattern Recognition Letters, Elsevier Science Inc., Vol. 19, 1998, pp. 255-259

[Carroll 2002] Carroll, Jeremy J., "Matching RDF Graphs", International Semantic Web Conference 2002 ISWC, Springer Lecture Notes in Computer Science, 2002, 2342, pp. 5-15

[Corby 2000] Corby, Olivier, Dieng, Rose and Hebert, Cedric "A Conceptual Graph Model for W3C Resource Description Framework", 8th International Conference on Conceptual Structures ICCS 2000, Springer, 2000, pp. 468-482

[Daconta 2003] Daconta, Michael C., Obrst, Leo J., Smith, Kevin T., "The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management", John Wiley & Sons, 2003

[DelBimbo 1999] Del Bimbo, Alberto, "Visual Information Retrieval", Morgan Kaufmann Publishers, 1999

[Diestel 2000] Diestel, R., "Graph Theory, 2nd Edition", Springer, 2000

[Ding 2004] Ding, Li, Finin, Tim, Joshi, Anupam, Pan, Rong, Cost, R. Scott, Peng, Yun, Reddivari, Pavan, Doshi, Vishal and Sachs, Joel "Swoogle: a search and metadata engine for the semantic web", CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management, ACM Press, 2004 , 652-659

[Fonseca 2004] Fonseca, Manuel, "Sketch-Based Retrieval in Large Sets of Drawings", PhD thesis, Universidade Tecnica de Lisboa, 2004

[Guarino 1999] Guarino, Nicola, Masolo, Claudio and Vetere, Guido, "OntoSeek: Content-Based Access to the Web", IEEE Intelligent Systems, IEEE Educational Activities Department, 14, 1999, pp. 70-80

[Hand 2001] Hand, David, Mannila, Heikki, Smyth, Padhraic, "Principles of Data Mining", MIT Press, 2001

[Hunter 2001] Hunter, Jane, "Adding Multimedia to the Semantic Web - Building an MPEG-7 Ontology", First Semantic Web Working Symposium (SWWS), Stanford, USA, 2001, pp. 261-281

[Kosch 2003] Kosch, Harald, "Distributed Multimedia Database Technologies supported by MPEG-7 and MPEG-21", CRC Press, November 2003

[Lux 2005] Mathias Lux and Michael Granitzer, "Retrieval of MPEG-7 based Semantic Descriptions", BTW-Workshop "WebDB Meets IR" in context of the "11. GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web", March 1st 2005, University of Karlsruhe, Germany. URL: http://caliph-emir.sourceforge.net/docs.html#publications

[Martínez 2003] Martínez, José M., "MPEG-7 Overview", Moving Picture Expert Group MPEG, Pattaya, March 2003, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm, last visited 15th July 2005

[Meyer zu Eissen 2005] Meyer zu Eissen, Sven, Stein, Benno and Potthast, Martin, "The Suffix Tree Document Model Revisited", I-KNOW 05: 5th Intl. Conference on Knowledge Management, Graz, Austria, 2005, pp. 596-603

[Rocha 2004] Rocha, Cristiano, Schwabe, Daniel and Aragao, Marcus Poggi, “A hybrid approach for searching in the semantic web”, WWW '04: Proceedings of the 13th international conference on World Wide Web, ACM Press, 2004, pp. 374-383

[Rogers 2002] Rogers, Ian, “The Google Pagerank Algorithm and How It Works”, IPR Computing 2002, http://www.iprcom.com/papers/pagerank/, last visited 15th Juli 2005

[Shasha 2002] Shasha, Dennis, Wang, Jason T. L., Giugno, Rosalba, “Algorithmics and applications of tree and graph searching”, in Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM Press, pp. 39-52, 2002

[Shokoufandeh 1999] Shokoufandeh, A., Dickinson, S.J., Siddiqi, K., Zucker, S.W., “Indexing using a spectral encoding of topological structure”, in Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999

[Simmons 1966] Simmons, R. F., “Storage and retrieval of aspects of meaning in directed graph structures”, Communications of the ACM, ACM Press, 9, pp. 211-215, 1966

[Smeulders 2000] Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R. "Content-based image retrieval at the end of the early years", Pattern Analysis and Ma-chine Intelligence, IEEE Transactions on, Vol. 22, No. 12, pp. 1349-1380, December 2000

[Stojanovic 2003] Stojanovic, Nenad, Studer, Rudi, Stojanovic, Ljiljana, “An Approach for the Ranking of Query Results in the Semantic Web”, in Proceedings International Semantic Web Conference 2003, pp. 500-516, 2003

[Tittmann 2003] Tittmann, Peter, “Graphentheorie”, Hanser Fachbuchverlag, ISBN: 3-446-22343-6, September 2003

[Valiente 2002] Valiente, Gabriel, “Algorithms on Trees and Graphs” Springer, ISBN 2-540-43550-6, 2002

[Yan 2004] Yan, Xifeng, Yu, Philip S. and Han, Jiawei, “Graph indexing: a frequent structure-based approach”, SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM Press, 2004, pp. 335-346

[Yang 1993] Yang, Gi-Chul and Oh, Jonathan, “Knowledge acquisition and retrieval based on conceptual graphs”, SAC '93: Proceedings of the 1993 ACM/SIGAPP symposium on Applied computing, ACM Press, 1993, pp. 476-481

[Zhu 2002] Zhu, Haiping, Zhong, Jiwei, Li, Jianming and Yu, Yong, “An Approach for Semantic Search by Matching RDF Graphs”, Fifteenth International Florida Artificial Intelligence Research Society Conference, AAAI Press, 2002, pp. 450-454

[Zhong 2002] Zhong, Jiwei, Zhu, Haiping, Li, Jianming and Yu, Yong, “Conceptual Graph Matching for Semantic Search”, ICCS '02: Proceedings of the 10th International Conference on Conceptual Structures, Springer, 2002, pp. 92-196

Density-based Cluster Algorithms in Low-dimensional and High-dimensional Applications

Benno Stein[1] and Michael Busch[2]

[1] Faculty of Media, Media Systems
Bauhaus University Weimar, 99421 Weimar, Germany
benno.stein@medien.uni-weimar.de

[2] IBM Silicon Valley Laboratory
WebSphere II OmniFind Edition Project

Abstract Cluster analysis is the art of detecting groups of similar objects in large data sets—without having specified these groups by means of explicit features. Among the various cluster algorithms that have been developed so far the density-based algorithms count to the most advanced and robust approaches.

However, this paper shows that density-based cluster analysis embodies no principle with clearly defined algorithmic properties. We contrast the density-based cluster algorithms DBSCAN and MajorClust, which have been developed having different clustering tasks in mind, and whose strengths and weaknesses can be explained against the background of the dimensionality of the data to be clustered.

Our motivation for this analysis comes from the field of information retrieval, where cluster analysis plays a key role in solving the document categorization problem. The paper is organized as follows: Section 1 recapitulates the important principles of cluster algorithms, Section 2 discusses the density-based algorithms DBSCAN and MajorClust, and Section 3 illustrates the strengths and weaknesses of both algorithms on the basis of geometric data analysis and document categorization problems.

Key words: density-based cluster analysis, high-dimensional data, document categorization

1 Cluster Algorithms

This section gives a short introduction to the problem of cluster analysis and outlines existing cluster approaches along the classical taxonomy. Moreover, it presents an alternative view to cluster algorithms, which is suited to explain different characteristics of their behavior.

1.1 The Classical Taxonomy of Cluster Algorithms

Definition 1 (Clustering). *Let D be a set of objects. A clustering $\mathcal{C} \subseteq \{C \mid C \subseteq D\}$ 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.*

With respect to the set of objects D the following shall be stipulated:

- $|D| = n$
- The objects in D represent points in the Euclidean space of dimension m.
- Based on a metric $d : D \times D \to \mathbf{R}$, the similarity or the dissimilarity between any two points in D can be stated.

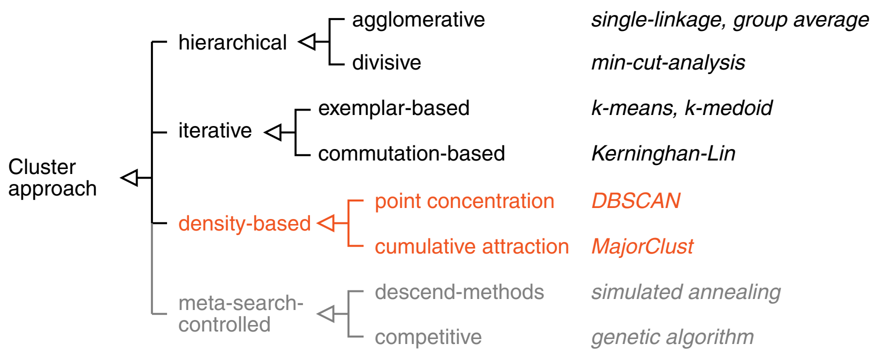

Figure 1. The classical taxonomy of cluster algorithms.

A cluster algorithm takes a set D of objects as input and operationalizes a strategy to generate a clustering $\mathcal{C}$. Informally stated, the overall objective of a cluster algorithm is to maximize the inner-cluster similarity and to minimize the intra-cluster similarity. The fulfillment of this objective can be quantified by a statistical measure like the Dunn index, the Davies-Bouldin index, or the Λ-measure [3, 5, 35].

Various cluster algorithms have been devised so far; usually they are classified with respect to their underlying algorithmic principle, as shown in Figure 1. Note that among these principles only the meta-search-controlled algorithms pursue a strategy of global optimization; hierarchical, iterative, as well as density-based algorithms are efficient implementations of particular heuristics. In the following we shortly outline the working principles.

Hierarchical Algorithms. Hierarchical algorithms create a tree of node subsets by successively subdividing or merging the objects in D. 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 are k-nearest-neighbor, linkage, Ward, minimum-spanning-tree, or min-cut methods . Usually, these methods construct a complete similarity matrix, which results in $\mathcal{O}(n^2)$ runtime [8, 10, 32, 17, 25, 38, 40].

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. Well-known representatives are k-Means, k-Medoid, Kohonen, Fuzzy-k-Means. The runtime of these methods is $\mathcal{O}(nkl)$, where l designates the number of iterations to achieve convergence [16, 27, 20, 23, 39, 13, 14, 24, 15, 36].

Commutation-based approaches take a random clustering or the outcome of another cluster algorithm as starting point and successively exchange nodes between the clusters until some cluster quality criterion is fulfilled [9, 21, 19].

Density-based Algorithms. Density-based algorithms try to separate the set D into subsets of similar densities. In the ideal case they can determine the cluster number k automatically and detect clusters of arbitrary shape and size. Representatives are DBSCAN, MajorClust, or Chameleon. The runtime of these algorithms is in magnitude of hierarchical algorithms, i, e., $\mathcal{O}(n^2)$, or even $\mathcal{O}(n\log(n))$ for low-dimensional data if efficient data structures are employed [34, 7, 18].

Meta-Search Algorithms. Meta-search algorithms treat clustering as an optimization problem where a global goal criterion is to be minimized or maximized. Though this approach offers maximum flexibility, there runtime is typically unacceptably high. Meta-search driven cluster detection may

	Cluster approach			
	hierarchical	iterative	density-based	meta-search controlled
Analysis strategy	relative comparison based on two items	absolute comparison based on k items	relative comparison based on k items	absolute comparison based on all items
Recovery characteristics	irrevocable	revocable	revocable	revocable

Table 1. Characterization of cluster algorithms with respect to the number of investigated items (points, clusters) and their ability to recover from suboptimum decisions.

be operationalized by genetic algorithms, simulated annealing, or a two-phase greedy strategy [2, 30, 31, 30, 29, 11, 22, 41].

1.2 An Alternative View to Cluster Algorithms

Experience has shown that—with respect to the clustering quality—density-based cluster algorithms outperform hierarchical as well as iterative approaches.[3] By definition, globally optimizing cluster algorithms will produce better clustering results; however, they play only an inferior role in practice:

(1) their runtime performance is usually unacceptable compared to the other approaches,
(2) for most clustering problems it is difficult to formally specify a global goal criterion.

Hierarchical cluster algorithms are well suited to detect complex structures in the data; however, they are susceptible to noise. Both properties result from the fact that—based on a pairwise similarity comparison of any two items—nearest neighbors are always fusioned. Iterative algorithms behave robustly with respect to noise but preferably detect spherical clusters. Density-based cluster algorithms provide a high flexibility with respect to cluster forms and address the problem of noise detection by simultaneously examining several items. Table 1 summarizes the properties.

The following section presents two density-based cluster algorithms in greater detail and discusses their properties.

2 Density-based Cluster Analysis with DBSCAN and MajorClust

A density-based cluster algorithm operationalizes two mechanisms:

(1) one to define a region $R \subseteq D$, which forms the basis for density analyses;
(2) another to propagate density information (the provisional cluster label) of R.

In DBSCAN a region is defined as the set of points that lie in the ε-neighborhood of some point p. Cluster label propagation from p to the other points in R happens if $|R|$ exceeds a given $MinPts$-threshold (cf. Figure 2). The following advantages (+) and disadvantages (–) are bound up with this concept:

+ good clustering results for geometrical and low dimensional data, if cluster distances can be inferred unambiguously from the density information in D,

[3] With respect to runtime performance density-based algorithms can be considered being in between hierarchical and iterative approaches.

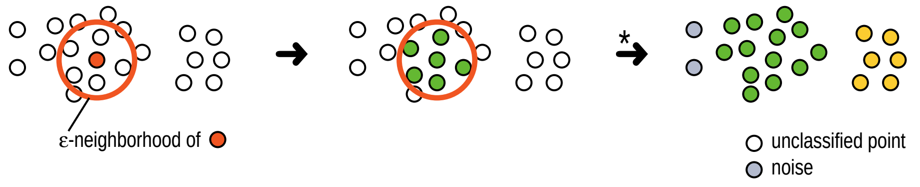

Figure 2. Illustration of DBSCAN's cluster process; to obtain the shown clustering result a $MinPts$-value from $\{3, 4, 5\}$ is required.

- + efficient runtime behavior of $O(n \log(n))$, if the R-tree data structure is employed to answer region queries for D,
- – parameters that characterize the density (ε, $MinPts$) are problem-specific and must be chosen manually,
- – if great variations in the point density or unreliable similarity values require a large ε-neighborhood, the density analysis becomes questionable: DBSCAN implements no intra-region-distance concept, and all points in the ε-neighborhood of some point are treated equally.
- – in high dimensions (> 10-20), the underlying R-tree data structure degenerates to a linear search and makes an MDS-embedding of D necessary, which affects both runtime and classification performance.

In MajorClust a region is not of a fixed size but implicitly defined by the current clustering $\mathcal{C}$. While DBSCAN uses the point concentration in ε-neighborhoods to estimate densities, MajorClust derives its density information from the attraction a cluster C exerts on some point q, which is computed as the sum of all similarity values $\varphi(q,p)$, $p \in C$. Cluster label propagation from C to q happens if the attraction of C with respect to q is maximum among the attraction values of all clusters in $\mathcal{C}$ (cf. Figure 3). Observe that the "true" density values for some data set D evolve during the clustering process. The following advantages (+) and disadvantages (–) are bound up with this concept:

- + robust for narrow data sets, for high-dimensional data, and for unreliable similarity values since the density computation does not rely on a fixed number of points and does consider distance information as well,
- + adapts automatically to different problems: no parameters must chosen manually,
- – with respect to runtime not so efficient as DBSCAN, since points are re-labeled several times,
- – clusters that are extended in one dimension are not reliably identified, since during re-labeling a tie-situation may occur ("tie-effect").

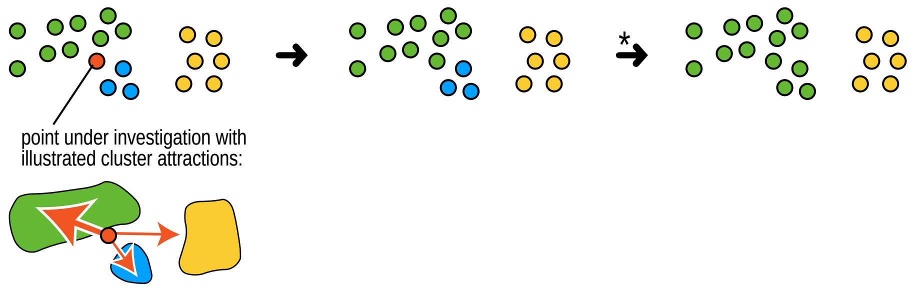
*
point under investigation with
illustrated cluster attractions:

Figure 3. Illustration of MajorClust's clustering process; each cluster C exerts attraction to some point q depending on both its size, $|C|$, and distance to q.

The above discussion reveals the strong and weak points of both algorithms, and the experiments presented in Section 3 will illustrate this behavior at realistic cluster problems. The next two subsections give a pseudo-code specification of both algorithms.

2.1 DBSCAN

DBSCAN operationalizes density propagation according to the principle "accessibility from a core point": Each point whose ε-neighborhood contains more points than $MinPts$ is called a core point. Each point which lies in an ε-neighborhood of a core points p adopts the same cluster label as p. The DBSCAN-algorithm propagates this relation through the set D. The algorithm terminates if each points is either assigned to a certain cluster or classified as noise.

Algorithm DBSCAN.
Input: object set D, region radius ε, density threshold $MinPts$.
Output: function $\gamma : D \to \mathbf{N}$, which assigns a cluster label to each point.

```
(01)  $Label := 1$
(02)  $\forall p \in D$ do $\gamma(p) := 'UNCLASSIFIED'$ enddo
(03)  $\forall p \in D$ do
(04)      if $\gamma(p) = 'UNCLASSIFIED'$ then
(05)          if $ExpandCluster(D, p, Label, \varepsilon, MinPts) = true$ then $Label := Label + 1$
(06)      endif
(07)  enddo
```

Function ExpandCluster.
Input: object set D, $CurrentPoint$, $Label$, ε, $MinPts$.
Output: $true$ or $false$.

```
(01)  $seeds := regionQuery(D, CurrentPoint, \varepsilon)$
(02)  if $|seeds| < MinPts$ then
(03)      $\gamma(CurrentPoint) := 'NOISE'$
(04)      return $false$
(05)  else
(06)      $\forall p \in P$ do $\gamma(p) := Label$ enddo
(07)      $seeds := seeds \setminus \{CurrentPoint\}$
(08)      while $seeds \neq \emptyset$ do
(09)          $p := seeds.first()$
(10)          $result := regionQuery(D, p, \varepsilon)$
(11)          if $|result| \geq MinPts$ then
(12)              $\forall ResPoint \in P$ do
(13)                  if $\gamma(ResPoint) \in \{'UNCLASSIFIED', 'NOISE'\}$ then
(14)                      if $\gamma(ResPoint) = 'UNCLASSIFIED'$ then $seeds := seeds \cup \{ResPoint\}$
(15)                      $\gamma(ResPoint) := Label$
(16)                  endif
(17)              enddo
(18)          endif
(19)          $seeds := seeds \setminus \{p\}$
(20)      enddo
(21)      return $true$
(22)  endif
```

Remarks. The function $regionQuery(D, p, \varepsilon)$ returns all points in the ε-neighborhood of some point p.

2.2 MajorClust

MajorClust operationalizes density propagation according to the principle “maximum attraction wins”: The algorithm starts by assigning each point in D its own cluster. Within the following re-labeling steps, a point adopts the same cluster label as the majority of its weighted neighbors. If several such clusters exist, one of them is chosen randomly. The algorithm terminates if no point changes its cluster membership.

```
Algorithm MajorClust.
Input:   object set $D$, similarity measure $\varphi : D \times D \to [0;1]$, similarity threshold $t$.
Output: function $\gamma : D \to \mathbf{N}$, which assigns a cluster label to each point.

(01)   $i := 0,\ ready := false$
(02)   $\forall p \in D$ do $i := i + 1,\ \gamma(p) := i$ enddo
(03)    while $ready = false$ do
(04)      $ready := true$
(05)      $\forall q \in D$ do
(06)        $\gamma^* := i$ if $\sum\{\varphi(p,q) \mid \varphi(p,q) \geq t$ and $\gamma(p) = i\}$ is maximum.
(07)        if $\gamma(q) \neq \gamma^*$ then $\gamma(q) := \gamma^*,\ ready := false$
(08)      enddo
(09)    enddo
```

Remarks. The similarity thershold t is not a problem-specific parameter but a constant that serves for noise filtering purposes. Its typical value is 0.3.

3 Illustrative Analysis

This section presents selected results from an experimental analyses of the algorithms DBSCAN and MajorClust. Further details and additional background information can be found in [4].

3.1 A Low-Dimensional Application: Analysis of Geometrical Data

The left-hand side of Figure 4 shows a map of the Caribbean Islands, the right-hand side shows a monochrome and dithered version (approx. 20,000 points) of this map, which forms the basis of the following cluster experiments.

A cluster analysis with DBSCAN requires useful settings for ε and $MinPts$. Figure 5 shows the clustering results with selected values for these parameters. Note that in this application the quality of the resulting clustering was more sensitive with respect to ε than to $MinPts$.[4]

[4] Ester et al. propose a heuristic to determine adequate settings for ε and $MinPts$; however, the heuristic is feasible only for two-dimensional data [7].

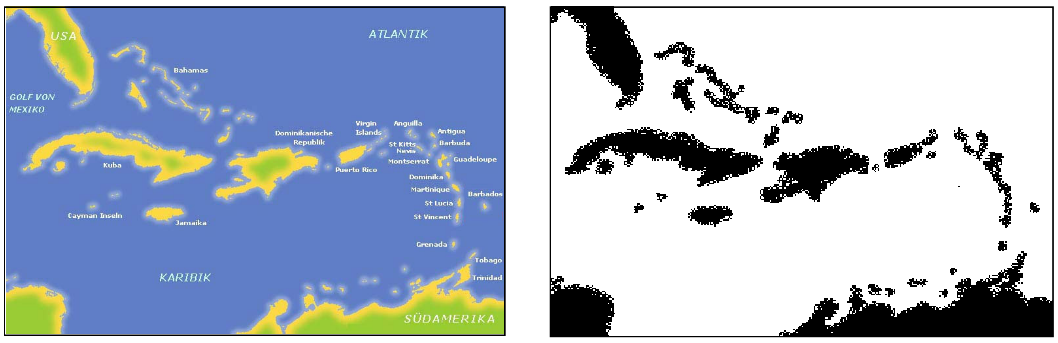

Figure 4. Map of the Carribean Islands (left) and its dithered version (right).

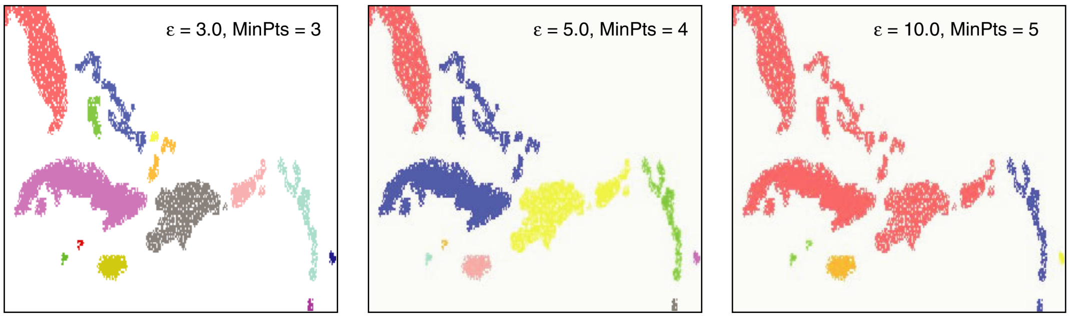

Figure 5. DBSCAN-clusterings of the Caribbean Islands for selected parameter settings.

A cluster analysis with MajorClust does not require the adjustment of special parameters. However, to alleviate noise effects the algorithm should apply a similarity threshold of (about) 0.3, i. e., discard all similarity values that are below this threshold. Figure 6 shows an intermediate clustering (left) and the resulting final clustering (right); obviously not all islands where correctly identified—a fact for which the formerly explained “tie-effect” is responsible.

3.2 A High-Dimensional Application: Document Categorization

Cluster technology has come into focus in recent years, because it forms the backbone of most document categorization applications [33]. At the moment it is hard to say which of the approaches shown in Figure 1 will do this job best: k-means and bisecting k-means are used because oft their robustness, the group-average approach produces similar or even better clustering results but is much less efficient, and from the density-based approaches the MajorClust-algorithm has repeatedly proved is high classification performance especially for short documents [1].

In this place we will report on experiments that rely on the Reuters-21578 corpus, which comprises texts from politics, economics, culture, etc. that have been carefully assigned to approx. 100 (sub-) categories by human editors [26]. For our experiments we constructed sub-corpora with 10 categories, each consisting of 100 documents that belong exclusively to a single category. The selected documents were transfered into the vector space model (VSM); the necessary pre-processing includes stop-word filtering, stemming, and the computation of term weights according to the tf-idf-scheme. Note that the resulting word vectors represent points in a space with more than 10,000 dimensions.

Since DBSCAN relies on the R-tree data structure it cannot process high-dimensional data (see Subsection 3.3), and multi dimensional scaling (MDS) had to be employed to embed the data into a

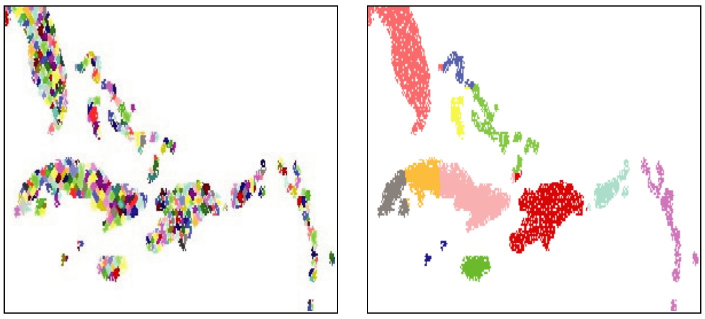

Figure 6. Left: Clustering after the first iterations of MajorClust; right: the resulting clustering.

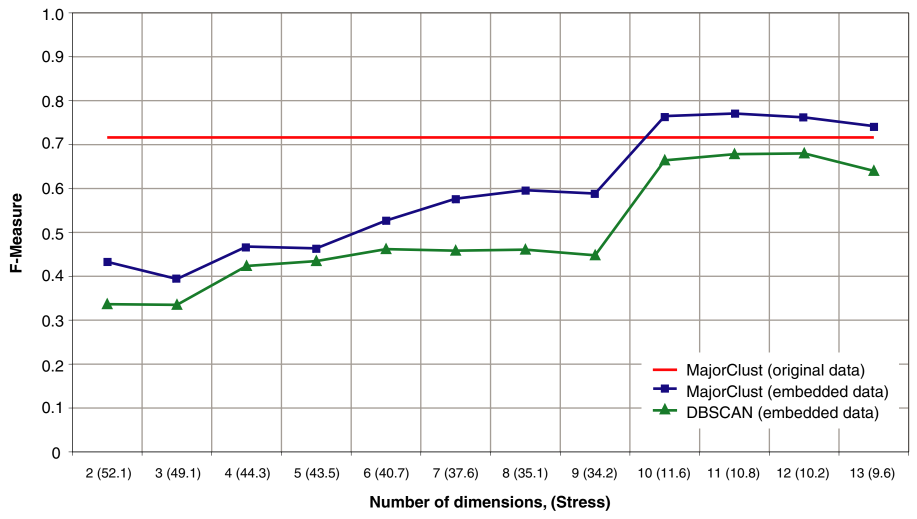

Figure 7. Classification performance of the algorithms in the document categorization task.

low-dimensional space. To account for effects that result from embedding distortions, MajorClust was applied to both the high-dimensional and the embedded data. The quality of the generated clusterings was quantified by the F-measure, which combines the achieved precision- and recall-values relative to the 10 classes.[5]

Figure 7 shows the achieved classification performance: The x-axis indicates the number of dimensions of the embedding space (from 2 to 13), the y-axis indicates the F-measure value. The horizontal line with an F-measure value of 0.72 belongs to MajorClust when applied to the original, high-dimensional data; with respect to the embedded data MajorClust dominates DBSCAN—independent of the dimension. In particular, it should be noted that the shown (high) F-measure values for DBSCAN are the result of extensive experimenting with various parameter settings.

Observe in Figure 7 that embedding can actually improve classification performance: If the number of dimensions in the embedding space equals the cluster number or is slightly higher, the noise-reduction effect compensates for the information loss due to the embedding error. [6] Put another way: Each dimension models a particular, hidden *concept*. This effect can be utilized for retrieval purposes and became known as *Latent Semantic Indexing* [6].

3.3 A Note on Runtime Performance

Both algorithms implement a density propagation heuristic that analyzes a region R with k contiguous points (recall Table 1). Based on R, DBSCAN performs a simple point count, while MajorClust evaluates the similarities between all points in R and some point q. Given an efficient means to construct the region R as ε-neighborhood of a point p, the runtime of both algorithms is in the same order of magnitude (cf. Figure 8).

To answer region queries, the R-tree data structure was employed in the above experiments [12]. For low-dimensional data, say, $m < 10$, this data structure finds the ε-neighborhood for some p in $O(\log(n))$, with $n = |D|$. As a consequence, the runtime of both algorithms is in $O(n \log(n))$. Since

[5] An F-measure value of 1 indicates a perfect match; however, F-measure values > 0.7 must be considered as certainly good since we are given a multi-class assignment situation where 10 classes are to be matched.

[6] The embedding error is called “stress” in the literature on the subject.

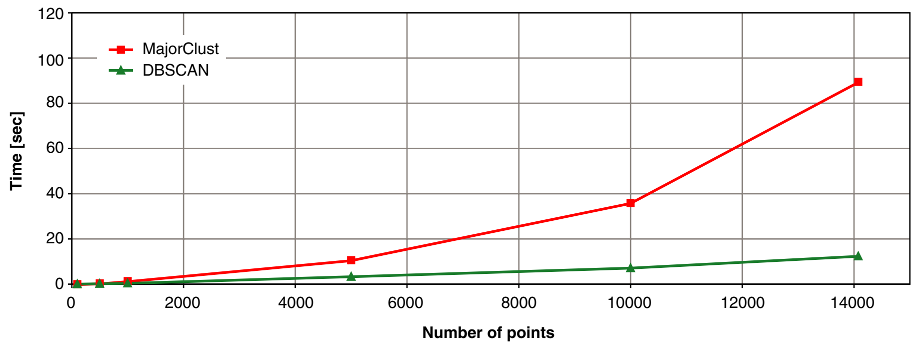

Figure 8. Runtime-behavior of the algorithms on two-dimensional data.

MajorClust evaluates the attraction of each point several times, its runtime is a factor above the runtime of DBSCAN.

Note that the R-tree data structure degenerates for higher dimensions ($m > 100$) and definitely fails to handle the high-dimensional vector space model.[7] I. e., a high-dimensional cluster analysis task like document categorization cannot directly be tackled with DBSCAN but requires a preceding data embedding step. At the moment the fastest embedding technology is the MDS-variant described in [28], which has not been tested for high-dimensional document models yet.

Summary and Current Work

This paper presented the classical and a new view to cluster technology and then delved into a comparison of the density-based cluster algorithms DBSCAN and MajorClust. This comparison is the first of its kind and our discussion as well as the experiments are interesting for the following reasons:

(1) Density-based cluster analysis should be considered as a collective term for (heuristic) approaches that quantify and propagate density information. In particular, no uniform statements regarding runtime-behavior or suited problem classes can be made.

(2) Since strengths and weaknesses of (density-based) cluster algorithms can be explained with the dimensionality of the data, a better mapping from algorithms to cluster problems may be developed.

Our current work concentrates on runtime issues of density-based cluster analysis for high-dimensional data: We investigate how the technology of Fuzzy fingerprints can be utilized to speed-up the region query task; the key challenge in this connection is to handle the inherent incompleteness of this retrieval technology.

References

[1] Mikhail Alexandrov, Alexander F. Gelbukh, and Paolo Rosso. An approach to clustering abstracts. In *NLDB*, pages 275–285, 2005.

[2] Thomas Bailey and John Cowles. Cluster Definition by the Optimization of Simple Measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1983.

[3] J. C. Bezdek and N. R. Pal. Cluster Validation with Generalized Dunn's Indices. In N. Kasabov and G. Coghill, editors, *Proceedings of the 2nd international two-stream conference on ANNES*, pages 190–193, Piscataway, NJ, 1995. IEEE Press.

[7] This relates also to other data-partitioning index trees such as Rf-tree or X-tree, as well as to space-partitioning methods like grid-files, KD-trees, or quad-trees [37].

[4] Michael Busch. Analyse dichtebasierter Clusteralgorithmen am Beispiel von DBSCAN und MajorClust. Study work, Paderborn University, Institut for Computer Science, March 2005.

[5] D.L. Davies and D.W. Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Learning*, 1(2), 1979.

[6] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[7] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. 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)*, 1996.

[8] B. S. Everitt. Cluster analysis. New York, Toronto, 1993.

[9] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the nineteenth design automation conference*, pages 175–181. IEEE Press, 1982. ISBN 0-89791-020-6.

[10] K. Florek, J. Lukaszewiez, J. Perkal, H. Steinhaus, and S. Zubrzchi. Sur la liason et la division des points d'un ensemble fini. *Colloquium Methematicum*, 2, 1951.

[11] D. B. Fogel and L. J. Fogel. Special Issue on Evolutionary Computation. *IEEE Transaction of Neural Networks*, 1994.

[12] Antonin Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In Beatrice Yormark, editor, *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984*, pages 47–57. ACM Press, 1984.

[13] Eui-Hong Han and George Karypis. Centroid-Based Document Classification: Analysis and Experimental Results. Technical Report 00-017, Univercity of Minnesota, Department of Computer Science / Army HPC Research Center, March 2000.

[14] Eui-Hong Han, George Karypis, and Vipin Kumar. Text Categorization Using Weight Adjusted k-Nearest Neighbor Classification. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 53–65, Minneapolis, USA, 2001. Univercity of Minnesota, Department of Computer Science / Army HPC Research Center.

[15] Makoto Iwayama and Takenobu Tokunaga. Cluster-based text categorization: a comparison of category search strategies. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of SIGIR-95, 18th ACM International Conference on Research and Development in Information Retrieval*, pages 273–281, Seattle, USA, 1995. ACM Press, New York, US.

[16] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering in Data*. Prentice Hall, Englewood Cliffs, NJ, 1990. ISBN 0-13-022278-X.

[17] S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32, 1967.

[18] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. Technical Report Paper No. 432, University of Minnesota, Minneapolis, 1999.

[19] George Karypis and Vipin Kumar. Analysis of multilevel graph partitioning. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 29. ACM Press, 1995. ISBN 0-89791-816-9. doi: http://doi.acm.org/10.1145/224170.224229.

[20] Leonard Kaufman and Peter J. Rousseuw. *Finding Groups in Data*. Wiley, 1990.

[21] B.W. Kernighan and S. Lin. Partitioning Graphs. *Bell Laboratories Record*, January 1970.

[22] R. W. Klein and R. C. Dubes. Experiments in Projection and Clustering by Simulated Annealing. *Pattern Recognition*, 22:213–220, 1989.

[23] T. Kohonen. *Self Organization and Assoziative Memory*. Springer, 1990.

[24] T. Kohonen, S. Kaski, K. Lagus, J. Salojrvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. In *IEEE Transactions on Neural Networks*, volume 11, may 2000.

[25] Thomas Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. John Wiley & Sons, New York, 1990.

[26] David D. Lewis. Reuters-21578 Text Categorization Test Collection. `http://www.research.att.com/∼lewis`, 1994.

[27] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[28] Alistair Morrison, Greg Ross, and Matthew Chalmers. Fast Multidimensional Scaling through Sampling, Springs and Ixnterpolation. *Information Visualization*, 2(1):68–77, 2003.

[29] V. V. Raghavan and K. Birchand. 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*, pages 10–22, 1979.

[30] Tom Roxborough and Arunabha. Graph Clustering using Multiway Ratio Cut. In Stephen North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer, 1996.

[31] Reinhard Sablowski and Arne Frick. Automatic Graph Clustering. In Stephan North, editor, *Graph Drawing*, Lecture Notes in Computer Science, Springer, 1996.

[32] P. H. A. Sneath. The application of computers to taxonomy. *J. Gen. Microbiol.*, 17, 1957.

[33] Benno Stein and Sven Meyer zu Eißen. Document Categorization with MAJORCLUST. In Amit Basu and Soumitra Dutta, editors, *Proceedings of the 12th Workshop on Information Technology and Systems (WITS 02), Barcelona Spain*, pages 91–96. Technical University of Barcelona, December 2002.

[34] Benno Stein and Oliver Niggemann. On the Nature of Structure and its Identification. In Peter Widmayer, Gabriele Neyer, and Stefan Eidenbenz, editors, *Graph-Theoretic Concepts in Computer Science*, volume 1665 LNCS of *Lecture Notes in Computer Science*, pages 122–134. Springer, June 1999. ISBN 3-540-66731-8.

[35] Benno Stein, Sven Meyer zu Eißen, and Frank Wißbrock. On Cluster Validity and the Information Need of Users. In M. H. Hanza, editor, *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 03), Benalmádena, Spain*, pages 216–221, Anaheim, Calgary, Zurich, September 2003. ACTA Press. ISBN 0-88986-390-3.

[36] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. Technical Report 00-034, Department of Computer Science and Egineering, University of Minnesota, 2000.

[37] Roger Weber, Hans-J. Schek, and Stephen Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *Proceedings of the 24th VLDB Conference New York, USA*, pages 194–205, 1998.

[38] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 1993.

[39] J. T. Yan and P. Y. Hsiao. A fuzzy clustering algorithm for graph bisection. *Information Processing Letters*, 52, 1994.

[40] C. T. Zahn. Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters. *IEEE Transactions on computers*, C-20(1), 1971.

[41] Ying Zaho and George Karypis. Criterion Functions for Document Clustering: Experiments and Analysis. Technical Report 01-40, Univercity of Minnesota, Department of Computer Science / Army HPC Research Center, Feb 2002.