ETH Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Computer Vision Laboratory
Prof. Luc Van Gool

Master's Thesis

Cortina: A System for Large-scale, Content-based Web Image Retrieval and the Semantics within

Till Quack

April 2004

Advisor ETH: Prof. Luc Van Gool,
Swiss Federal Institute of Technology,
Zürich, Switzerland.

Advisor UCSB: Prof. B.S. Manjunath,
University of California at Santa Barbara,
Santa Barbara, USA.

Hunc ubi vix multa maestum cognovit in umbra, sic prior adloquitur: "quis te, Palinure, deorum eripuit nobis medioque sub aequore mersit? dic age. namque mihi, fallax haud ante repertus, hoc uno responso animum delusit Apollo,qui fore te ponto incolumem finisque canebat venturum Ausonios. en haec promissa fides est?" ille autem: "neque te Phoebi cortina fefellit, dux Anchisiade, nec me deus aequore mersit. Namque gubernaclum multa vi forte revulsum, cui datus haerebam custos cursusque regebam, praecipitans traxi mecum.[1]

Virgilio, Aeneidos, Liber VI, vi. 340-351

[1] A translation can be found in Appendix B

Acknowledgments

This thesis came into existence during a stay at the the University of California at Santa Barbara, USA. Many people helped to make it a great experience, and I would like to thank in particular:

Prof. Luc Van Gool from ETH Zürich,who supported me in going abroad for my Master's Thesis and made this stay possible in the first place,

Prof. B.S. Manjunath from UCSB for his generous support and the discussions during the conception of this project as well as his kind hospitality,

Ullrich Moenich and Lars Thiele who contributed as exchange students from Germany to our joint project.

I also want to express my gratitude to all the other members of the Vision Research Laboratory at UCSB, who made my stay so enjoyable, and in particular Sitaram Bhagavathy, Baris Sumengen and Jelena Tešić for many interesting and helpful discussions.

Financial support for the exchange from ETH Zürich is also gratefully acknowledged.

Santa Barbara, 25 April 2004

Till Quack

Abstract

Recent advances in processing and networking capabilities of computers have led to an accumulation of immense amounts of multimedia data such as images. One of the largest repositories for such data is the World Wide Web. There is an urgent need for systems which allow to search these vast on-line collections.

We present *Cortina*, a large-scale image retrieval system for the World Wide Web. It handles over 3 Million images to date. The system retrieves images based on visual features and collateral text. Methods are introduced to investigate these multi-modal characteristics of the data and to gain insights into the semantics within the data. We show that a search process which consists of an initial query-by-keyword and followed by relevance feedback on the visual appearance of the results is possible for large-scale data sets. We also show that it is superior to the pure text retrieval commonly used in large-scale systems. The precision is shown to be increased by exploiting the semantic relationships within the data and by including multiple feature spaces into the search process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years advances in processing and networking capabilities of computers have led to an accumulation of immense amounts of data. Many of these data are available as multimedia documents, i.e. documents consisting of different types of data such as text and images. By far the largest repository for such documents is the Word Wide Web (WWW). Currently, Google [1] offers access to over 4.2 billion documents, which gives a rough estimate of the size of this online data collection.

Clearly, the amount of information available on the WWW creates enormous possibilities and challenges at the same time. While *everything can* be found it is hard to find *the right thing* - while this is true for pure text retrieval (i.e. web-pages), it is even harder for multimedia content.

Much progress has been made in both text based search on the WWW and content-based image retrieval for research applications. However, little work has been done to combine these fields to come up with large-scale, content-based image retrieval for the WWW.

Commercial search engines like Google [1] have successfully implemented methods [2] to improve text-based retrieval. While text-based search on the WWW has reached astounding levels of scale and some degree of sophistication, commercially available search for *images* or other multimedia documents also relies on the use of text only. In addition, no commercial search engine offers the possibility to refine the search using relevance feedback on visual appearance, i.e. the option to look for results visually similar to an image selected from a first set of results.

The content based image retrieval (CBIR) systems introduced by the image processing or computer vision community on the other hand, exist

in a research form and do not have commercial potential. In general, the database of the proposed systems is relatively small, and the proposed system does not scale well with increase in the database's size. Few scale to some extent [3], but only if they are restricted and tuned to a certain class of images — e.g. aerial pictures or medical images. In addition, many of these CBIR systems rely only on low level features and do not incorporate other data to improve the semantic quality of the search.

The goal of this project is to take content-based web-image retrieval to larger scales and thus one step closer to commercial applications. Further contributions focus on the exploration and exploitation of the multi-modal characteristics of web-data to increase the semantic quality of search results, as will be explained in detail in the following sections.

1.1 Retrieving Images from the World Wide Web

Many of the challenges for image retrieval on the WWW stem from its characteristics as an information source:

- Size: The sheer amount of documents poses challenges in computational requirements which have immediate consequences on the choice of methods to handle these data.
- Diversity: The documents can appear in many contexts and in many languages. The quality (in a semantic sense) of the information source is hard to determine. Images can be of any dimension, many file-formats and wide range of qualities.
- Control: There is no control on what people publish and in what form. Especially, there is no control on what text appears in the context of an image on a website.
- Dynamics: The data collection is always changing. There are new web-sites every day, web-sites become outdated or go off-line.

Usually, the information is *multi-modal*: Different types of information appear together. In our case: images usually appear embedded within text. In this sense image retrieval is similar to other multi-modal information repositories like newspaper archives, stock photography or any kind of database that has *meta-data* assigned to images. However, all of these collections

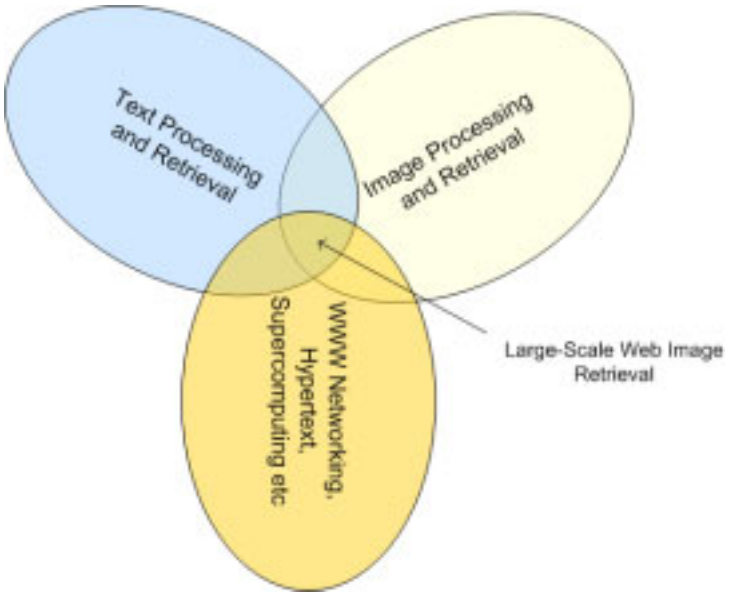

Figure 1.1: Web-Image retrieval

are usually smaller, limited in context and rather well-controlled. Image retrieval of the web is an interdisciplinary problem which touches several areas such as image processing and retrieval, text retrieval, networking and even supercomputing as shown in figure 1.1. The challenge is to combine these areas such that the overall performance of the system is increased. The thesis at hand will focus on the information retrieval side of the problem with simultaneous consideration of all the other factors.

1.2 Previous Work

In recent years, much work has been done and published on CBIR. The QBIC project [4] at IBM was one of the earlier systems that allowed content-based retrieval by color or texture and also was applied commercially. Numerous CBIR systems followed: BlobWorld [5] and NeTra [6] included image segmentation, and PicHunter [7] and MARS [8] offered relevance feedback.

Most of these systems rely on pure visual features and few focused specifically on content based image retrieval from the web: WebSeer [9] included some image features which allowed the user to restrict a search to sketches, photographs or faces. Newsam et al introduced a category based system [10] for about 600 000 images with relevance feedback. They limit the search

space for nearest neighbor search to the DMOZ [11] categories which means their system relies on a predefined categorization of the data.

Work on systems and methods that integrate text *and* visual features will be discussed in detail at the beginning of chapter 3.

1.3 Task

Content-based image retrieval on the web is taken one step closer to real world applications. We propose and implement a system which makes large-scale, content-based image retrieval on the web possible. We collect a dataset consisting of several Millions of items. We scale image-retrieval methods to these large amounts of data. The system uses several MPEG-7 visual features and text. It also offers the possibility to refine the search based on relevance feedback. I chose the focus of my research and the thesis at hand to be on the multi-modal characteristics of the data set. Search in different feature spaces is discussed and implemented. The semantic relationships within the data are explored and scalable methods are identified and implemented to derive improvements in the precision of our system. The system is made available on the World Wide Web. The quality of the search is quantified as far as it is possible for large-scale search engines.

1.4 Organization

The document is organized as follows. Chapter 2 contains an overview of the system modules, discusses the features, their extraction and search in the feature spaces. Chapter 3 presents methods to discover and exploit semantic relationships in web-data - it contains the largest part of my theoretical contributions to the system. In chapter 4 we discuss the details of the system architecture, i.e. the software and hardware part of this project. Chapters 5 and 6 contain experimental results and conclusions, respectively.

Chapter 2

System Overview

This section gives a short overview of the image-retrieval system. It introduces the basic modules and processes of the system for a more in depth discussion of the semantics within, which will be presented in chapter 3. The focus of this chapter is to present the design and and the elements of the system in an abstract manner. Details on the specific software design of the system will be discussed in chapter 4.

The system as shown in figure 2.1 can be conceptually separated in two large blocks: One for the retrieval, i.e. the interaction with the user, and another one for the off-line collection and preprocessing of the data.

We will first describe how the user experiences a typical search process, i.e. the retrieval part of the system. The search process typically starts with a keyword query through a web-interface. The request is sent to an inverted keyword index. As a response, the system delivers matching images, ranked by textual and visual features. The user now can choose the images that are visually closest the semantic concept he was looking for. Based on this information, the system returns a refined answer, i.e. a new, more precise set of images. This interaction is generally known as (short-term) relevance feedback and can be repeated several times, until one is satisfied with the results.

To enable such a search process the data from the WWW has to be collected, encoded and stored in a way that describes the information in a compact way. All this happens in the off-line part of the system. First, the information is gathered from the WWW. A custom web-crawler, the image spider, collects images and associated text automatically. The DMOZ Open Directory [11] served as a starting point for the image spider. From each

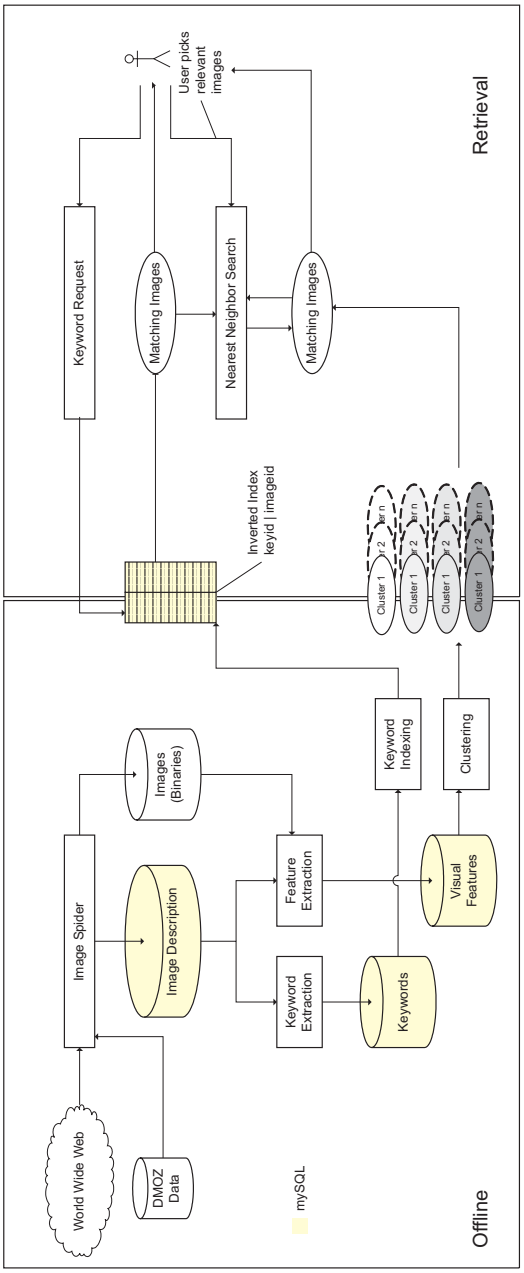

Figure 2.1: System overview

element of data (i.e. an image with associated text) textual and visual features are extracted and stored. Text is encoded into keywords and linked to corresponding images by an inverted index. Visual features are extracted and stored in several clusters.

The following sections of this chapter deal with the basic kinds of data and retrievals in our system: In section 2.1 the different types of features are introduced. In section 2.2 we discuss how to search these individual feature spaces. Section 2.3 contains a description of the relevance feedback process.

2.1 Features

The search process returns images that match a query by comparing several types of features of the query image and the retrieval candidates. Before the retrieval is described (Chapter 2.2) we give an overview over the features that have been used in our system.

2.1.1 Visual Features

For the visual features we chose four feature types from the MPEG-7 standard [12] which were applied in a global manner to the whole image. In particular no segmentation or tiling of the images was done. The reasoning behind this is that the collection of images from the WWW is extremely diverse and of very different qualities. It is thus questionable if a segmentation would give consistent results. Also, considering the large amount of images we intend to collect, processing time is crucial. By omitting segmentation or tiling and applying the feature extraction to the image as a whole the process of extracting the features can be sped up significantly.

The four descriptors chosen from the MPEG-7 standard included two texture descriptors and two color descriptors, respectively: The homogeneous texture descriptor (HTD), the edge histogram descriptor (EHD), the scalable color descriptor (SCD) and the dominant color descriptor (DCD).

The HTD is a vector consisting of the outputs of a Gabor filter bank. The 2-D frequency plane is partitioned into 30 channels which are modeled by Gabor filters with different scales and orientations. The feature vector layout is

$$HTD = [f_{DC}, f_{SD}, e_1, e_2 \ldots, e_{30}, d_1, d_2, \ldots, d_{30}]$$

where e_i and d_i are the nonlinearly scaled and quantized mean energy and energy deviation of the ith channel, respectively. f_{DC} and f_{SD} are the mean and standard deviation of the whole image. This means, the HTD is a 62-dimensional feature vector. For retrieval only the distance between two feature vectors $distance(HTD_{Query}, HTD_{Database})$ needs to be calculated [1].

The EHD contains information about the spatial distribution of edges in an image. The image is divided into 4×4 sub-images. For each of these sub-images the local-edge histogram is stored. For the edge histogram edges are categorized into five types: vertical, horizontal 45 degrees diagonal, 135

[1] Details on the choice of a specific distance function are given in the next section.

degrees diagonal, and non-directional. This results in a $5 \times 16 = 80$ dimensional vector. To obtain the histogram for each sub-image it is subdivided into smaller image blocks within which edge detectors are applied. The vector layout is

$$EHD = [h^{90}_{0,0}, h^{0}_{0,0}, h^{45}_{0,0}, h^{135}_{0,0}, h^{nondir}_{0,0}, \ldots, h^{90}_{3,3}, h^{0}_{3,3}, h^{45}_{3,3}, h^{135}_{3,3}, h^{nondir}_{3,3}]$$

where $h^{\alpha}_{i,j}$ is the histogram count for tile (i, j) and direction-bin α. The two texture descriptors were chosen because they complement each other: The EHD performs best on large non-homogeneous regions, while the HTD operates on homogeneous texture regions. A detailed description of the texture descriptors can be found in [12].

The SCD is a color histogram in the HSV color space, which is encoded by a Haar transform. The basic unit of the Haar transform consists of a sum and a difference of two adjacent histogram bins — primitive low- and high-pass filters. This unit is applied across the 256-bin color histogram of the image. Optional repetition results in lower resolution descriptors of 128, 64, 32 and 16 bits. Thus the name *scalable* color descriptor. For retrieval the L_1-Norm in the Haar space is used. A detailed description of the SCD can be found in [12].

The DCD provides a compact description of the most dominant colors in an image. Unlike histogram-based color descriptors, the dominant colors are calculated for each image instead of being fixed in the space defined by the histogram bins. The Layout of the DCD is

$$DCD = \{(\mathbf{c}_i, p_i)\}, i = 1, 2, \ldots, N$$

where N is the number of dominant colors. $\mathbf{c}_i$ stands for a 3-D color vector (e.g. LUV or RGB space), p_i is the percentage of pixels in the image corresponding to color i. The maximum value for N is 8. To extract the dominant colors, the colors in the image are clustered, usually in a perceptually uniform color space such as the LUV space. The retrieval process is different from the other descriptor types since the individual values of the DCD vector do not stand for dimensions in a feature space. Consider two DCDs,

$$DCD_1 = \{(\mathbf{c}_{1i}, p_{1i})\},\ i = 1, 2, \ldots, N_1$$
$$DCD_2 = \{(\mathbf{c}_{2i}, p_{2i})\},\ i = 1, 2, \ldots, N_2$$

Figure 2.2: From images on web-pages to keywords

Then the dissimilarity can be computed as

$$D(DCD_1, DCD_2) = \sum_{i=1}^{N_1} p_{1i}^2 + \sum_{j=1}^{N_2} p_{2i}^2 - \sum_{i=1}^{N_1}\sum_{j=1}^{N_2} 2a_{1i,2j}p_{1i}p_{2j} \qquad (2.1)$$

where $a_{k,l}$ is the similarity coefficient between two colors c_k and c_l

$$a_{k,l} = \begin{cases} 1 - d_{k,l}/d_{max} & d_{k,l} \leq T_d \\ 0 & d_{k,l} > T_d \end{cases}$$

with $d_{k,l} = ||\mathbf{c}_k - \mathbf{c}_l||$ the Euclidean distance between two colors $\mathbf{c}_k$ and $\mathbf{c}_l$. T_d is the maximum distance for two colors still considered to be similar and $d_{max} = \alpha T_d$, where α is a parameter. This dissimilarity measure is introduced and shown to be equivalent to the common quadratic (Mahalanobis) distance measure between two color histograms in [13].

2.1.2 Text and Keywords

In addition to the image features we collect textual information related to each image from the WWW. The process that leads from web-images to keywords is shown in figure 2.2.

Each web-page our web-crawler visits is analyzed. We look for the HTML tag `<img src= alt= />` which embeds an image into the HTML code. The image is saved and later on the visual features are extracted as described above. The text in the `alt` tag is saved. The information around the `<img/>` tag is analyzed, too. HTML tags are removed and we are left with *collateral text*. So called *stopwords*, i.e. words like *I*, *the*, *that* are removed from the text. The remaining terms are normalized in that their commoner morphological and inflexional endings are removed and only the remaining stem is stored. For this purpose an implementation of the Porter stemmer [14]

was used. Doing so, more documents match a keyword. If, for instance, one document was assigned the keyword *training* and another one the word *trained* they now have the common, stemmed keyword *train*. As the attentive reader may have realized, the stemmer introduces also ambiguities since *train* is not only the stem for the verb *to train* but also the noun *train*, i.e. a mean of transportation. These ambiguities could be removed to some extend with a syntactical analysis of the text, however, this is out of the scope of this work.

Note that collecting the collateral text around the image often gives extremely noisy results. The text is often less precise than for instance in newspaper or magazine articles. Many of our images are collected from shopping pages, text around these images often contains useless words such as *item*, *price*, *size*, *color* which stem from interfaces where the user can define these options. Or in many cases an image like that in figure 2.2 would not have the most characteristic keyword *shoe* but some words such as *red* or *adidas* only. In addition to these WWW-related problems there are issues that affect every document collection like the *paraphrase* problem, i.e. that the same object or concept can be described with completely different words. This can only be overcome in controlled environments by using a restricted vocabulary with precise rules for manual annotation as often used in stock photography databases like [15, 16].

We will get back to some of these issues in chapter 3 where we deal with the semantics in our collection.

2.2 Searching the Feature Spaces

In the last section our visual and textual features were introduced. In this section we discuss how a search in these feature spaces can be done and which implementations we chose for our system.

2.2.1 Visual Feature Space

A search in the visual feature space usually boils down to a search in a n-dimensional vector space. In our system the HTD, EHD and SCD can be treated this way, since they are all described by a vector in their feature space. The DCD however needs special treatment since its layout is different as the reader may remember from the previous sections. For the search in the n dimensional vector space the similarity between query x and retrieval candidate y is measured by a distance d. We used variations of the Minkowski metric for the distance measures in our system:

$$d = L_k(x, y) = ||x, y||_{L_k} = \sqrt[k]{\sum_{i=1}^{n} |x_i - y_i|^k} \tag{2.2}$$

In particular, for the EHD and the SCD the L_1 norm was used, and for the HTD the L_2 norm. For the HTD the MPEG-7 standard [12] suggests a distance based on the Mahalanobis distance, however it has been show that the L_2 norm performs equally. Usually one is not interested in all the distances, especially when the dataset is very large, but only in the k closest images. Finding the k closest neighbors is known as k-nearest neighbor search or $k-NN$ search for short. Thus, especially for retrieval systems as ours, approximations are necessary, which retrieve the $k-NN$ in reasonable time. This means that the data needs to be preprocessed, in simple words: the data needs to be "pre-sorted" in some manner which enables fast access to the $k-NN$ of any query vector. This is a challenging problem, which gets even more complicated for high dimensional (usually $n > 20$ is considered high dimensional) feature spaces.

Clustering the Visual Feature Space

One of the methods to pre-sort our data is to divide the feature space into partitions, so called *clusters*. All of the vectors for each of these clusters are stored in one separate file. Instead of searching the whole dataset for

the $k - NN$, first the query is compared to a representative, the *centroid* for each cluster. The cluster corresponding to the closest centroid is chosen and the $k - NN$ search is limited to this cluster. This way the amount of disk accesses is limited and datasets that don't fit into main memory can be searched reasonably fast.

Since HTD, EHD and SCD are high-dimensional feature vectors, some remarks have to be made about clustering in these environments. Our conception of space is based on our experience with three dimensional spaces. However, high dimensional spaces behave differently which is often referred to as the *curse of dimensionality*. An extensive discussion of this matter can be found in [17]. A short summary of the problems that affect our system can be given as follows:

- In high dimensional spaces of dimension n, volumes are spread along the surface of objects. It can be shown that, because of this effect, the probability that an arbitrary item lies in a spherical query approaches 0 as $n \to \infty$.
- The concept of nearest neighbors loses its meaning in high-dimensional spaces, since it has been shown that minimum and maximum distance for a query point are almost the same — for any distance metric or data distribution.
- The data in high dimensional spaces is usually very sparse.

These problems actually make meaningful clustering based on distances very unlikely in high-dimensional spaces. However, the current literature gives no measures when a distance based clustering will work and when it will fail. For us, the hope is that at least clustering will provide some partitioning of the data which enables faster $k - NN$ search and some dimensionality reduction for applications built on top of it. In fact, for once having a large dataset is theoretically an advantage — the more data is available the better the sparsity of the data can be overcome. However, the computational demands make it usually infeasible to use the whole dataset, which means that only a sample can be taken to create the clusters.

The choices for and implementation of clustering of the visual features were done in a separate work [18]. Thus, we will give only a short summary to make the reader familiar with the basic concepts. The K-Means algorithm was chosen as a clustering procedure - mainly because it is known to

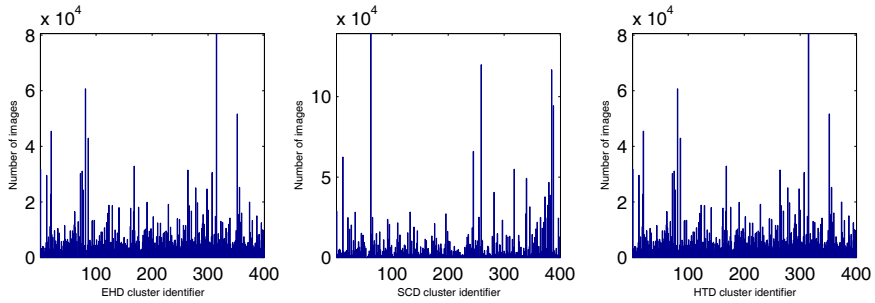

Figure 2.3: Sizes of the EHD, HTD and SCD clusters

be fast which is crucial considering the size of our data set. It is an iterative clustering technique that results in K data clusters. K centroids are initialized by choosing points that are mutually farthest apart. In each iteration, the algorithm recomputes the set of better partitions of the input vectors, and their centroids. In our project 10% of the database were taken as a sample to calculate the cluster centroids. The whole dataset was assigned to the closest centroids to form the clusters. 400 such clusters were created per feature type. The distribution of the images over the clusters is shown in figure 2.3. Note, that unfortunately the distribution is very uneven.

In figure 2.4 an imminent problem of such a clustering is shown: What if a query is close to the border of a cluster? By considering only the cluster in which the query lies, we loose a lot of the nearest neighbors. The solution suggested in [18] is to extend each cluster to a hypersphere S_i around the centroid c_i. This way, overlapping clusters are created and more close neighbors are captured. The radius for the hypersphere around a cluster centroid was chosen to be 2 times the distance to the outermost member of the original cluster. Since the original distribution was very uneven, the largest cluster contained 4×10^5 images - roughly 13% of our dataset. This means that the scalability of this method is not very good. We will come back to this problem in chapter 3 where we suggest an alternative solution based on semantic information.

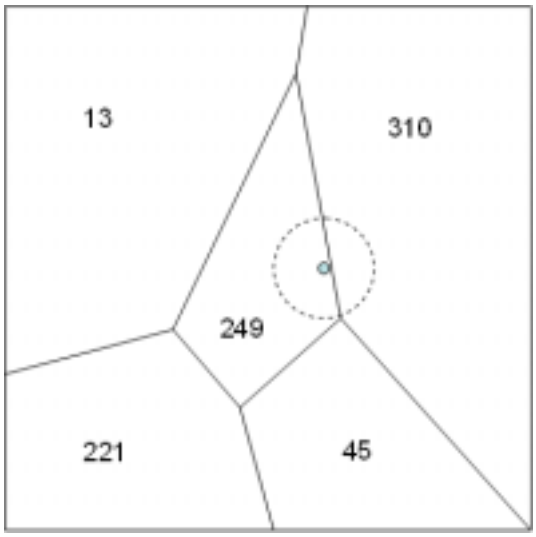

Figure 2.4: A query (symbolized by the dot) is close to the border of the voronoi cell

k-NN-search and Clustering for the DCD

Since the concept of the DCD differs from the one for HTD, EHD and SCD, i.e. it is not a vector in a high dimensional feature space, the search process is not the same either. k-NN search in the context of the DCD means searching the database for images with a similar distribution like the query image. The database is first searched for each of the dominant colors of the query separately. Then the results are combined. In [13] the search procedure is given as follows:

1. For each query color, find the matching images that contain similar colors. To quickly eliminate some false matches, a threshold T_p is set for the difference between the query percentage q_i and the retrieved percentage p_i. A candidate image is eliminated if the following condition is not true
$$|q_i - p_i| < T_p$$

2. Join the results from step 1, i.e. find the images that contain all the query colors and passed step 1. In addition
$$\sum_i p_i > T_t$$
must be true. T_t was set to 0.6.

3. Calculate the distances between all the remaining retrievals and the query with the distance measure given in equation 2.1.

To find the similar colors [13] defines a special lattice indexing structure. We decided just to cluster the colors with the K-Means algorithm that we had already in place. The good news here is that each color i is represented by a 3-dimensional vector $\mathbf{c_i}$, i.e. searching for individual colors does not happen in a high-dimensional feature space. Thus clustering the individual colors presents no problem. Note that each image appears in several clusters, one for each dominant color. Also, we decided to skip step 3, i.e. we just select the images which have similar color and percentage distributions like the query.

2.2.2 Visual Feature Combination

Since we have several visual feature types, we need to combine the results of the retrievals for each feature-type, if we want to do a joint search. We decided to combine the distances of the four descriptors in a linear way.

$$d = \varepsilon * d_{EHD} + \eta * d_{HTD} + \varsigma * d_{SCD} + \delta * d_{DCD}, \quad \varepsilon + \eta + \varsigma + \delta = 1 \quad (2.3)$$

The following questions need to be considered:

- The distances have different ranges for each descriptor type. How do we normalize them?
- Which ratios should be used to combine the distances, i.e. what values to set for ε,η,ς and δ?

To normalize the distances their distribution has to be examined. The distribution of the distances of the four descriptor types is shown in figure 2.5. We tried to match each distribution with a known distribution. The parameters were determined with a Maximum-Likelihood parameter estimation.

Based on these distances we decided to transform each of the distributions to a Gaussian distribution in the range $[0, 1]$ to be able to compare and combine the distances.

The ratios ε, η, ς and δ should be set query-dependent, i.e. for many queries, there is a descriptor type which is better suited to describe a semantic concept. E.g. images of objects on a uniform background are well-described with the EHD, for a query related to the ocean one of the color

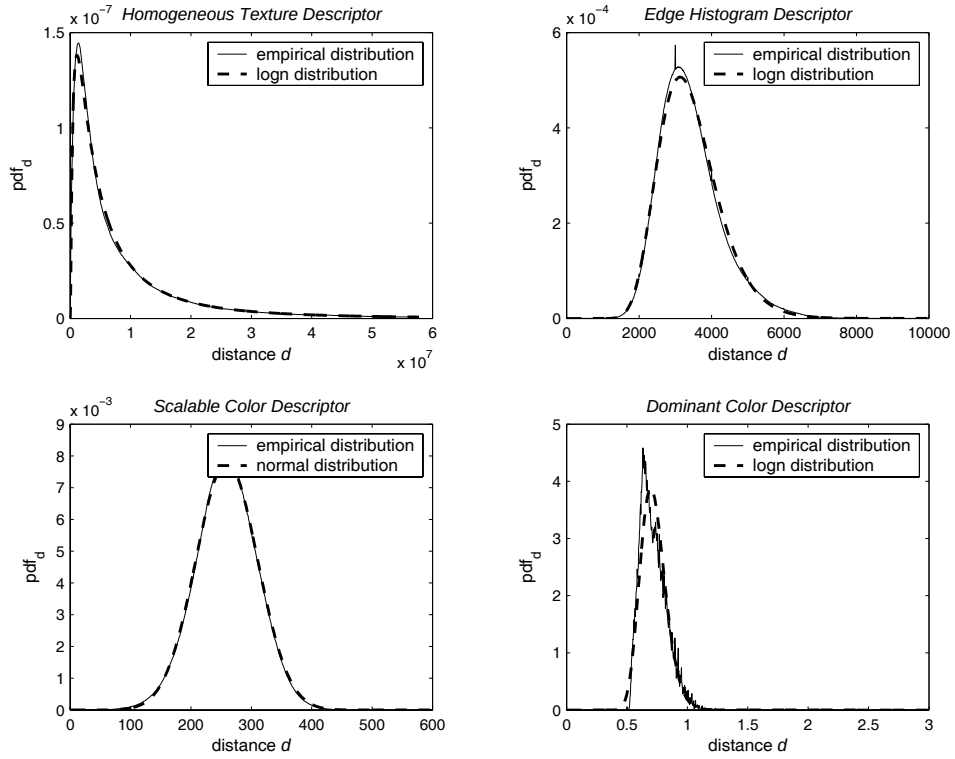

Figure 2.5: Pdf of the distances for each descriptor type along with the best matching distributions

types might be a better feature. These are general assumptions, in the specific case it depends on the user's intent. Usually a user enters the system with query-by-keyword and then refines the search with relevance feedback. In chapter 3 we suggest an approach to set the initial values based on information collected from the dataset keyword. For the moment, consider an initialization with each weight set to 0.25. The weights are then changed in the relevance feedback process by the information obtained from images selected by the user as relevant. This is described in section 2.3.

The descriptor type with the highest weight is chosen as the *primary descriptor*. $k-NN$ search is done in the clusters of this feature type. Now, consider that the EHD is our primary descriptor, we search for images that are close to the query. Since the EHD is the primary descriptor we search the EHD cluster of the closest centroid to the query. If we just did the same for each secondary-feature cluster, and then took the intersection of the results, there might be very few images (if not none) that are in the $k-NN$ for all the descriptor types. As an approximation we search only the images that are in the cluster of the primary descriptor type. This means, however, that all the other descriptors, the *secondary descriptors*, need to be stored along with the primary descriptor clusters — for each descriptor type. Otherwise the gain achieved by clustering is lost.

2.2.3 Text Feature Space

To implement a search in the textual feature space one can choose from many options. They include the standard vector space model [19], boolean queries on an inverted index or more sophisticated methods like LSI [20]. In spite of these additional options, to our knowledge large-scale web-retrieval systems usually rely on boolean queries on an inverted index - starting with the earliest [21] up to the currently most successful [1]. We decided to follow the same path at least for the initial keyword search, since not only the boolean queries on an inverted index are a well-established instrument but also because the focus of this work shall be on image retrieval rather than text retrieval. We implemented a *ranked boolean OR query*: Query words separated by whitespace are implicitly converted to an OR query. The rank is set based on the common $tf * idf$ measure:

$$w_{i,j} = tf_{i,j} * log(\frac{N}{n_i}) \tag{2.4}$$

where $tf_{i,j}$ is the number of times term i appears in document j, n_i is the number of documents containing the term i and N is the total number of documents. tf is the term-frequency, and $idf = log(\frac{N}{n_i})$ is the inverse document frequency. The rationale behind this is that with tf a document is ranked higher if the term i appears often in the document, idf is a measure for the term importance, i.e. how often term i appears in the overall collection of size N.

An inverted index is built with the keywords extracted as explained in the last section. For each keyword and each document, there is an entry in the inverted index which maps the keyword to the document along with the frequency it appeared. Our inverted index is implemented as a table in a relational database. The essential columns of our inverted keyword index are shown in 2.1. Note that the keywords are represented by their id `KeyID`. This is crucial for performance, since indexing and accessing a column with numeric values is much faster than using a column with alphanumeric values, i.e. the keywords themselves. The relation between the keyword ids and the keywords is established in another table. Also, we have two columns for the keyword frequency: `altfreq` if the keyword appeared in the `alt` tag of the image, `keyfreq` for the text around the image. This way they can be weighted differently, i.e. the keywords in the alt tag can be given a higher weight since they are "closer" to the image. From the layout of the inverted index it becomes clear why boolean queries on such an index is so widely used in large-scale applications: The representation as an inverted index is very compact for *high-dimensional yet sparse* data: As each document can be treated as a point in the term space, i.e. each possible keyword is a dimension in this space, vectors to represent the documents are of dimensions between 10^4 and 10^5 in uncontrolled collections. However, our documents, i.e. the words assigned to an image, are in the order of 10^1 to 10^2.

A common data-structure for such indices are B-Trees. We chose a relational database to implement our inverted index. Relational databases usually offer standard functionality to create B-Tree indices on table columns. Moreover, boolean queries on the inverted index can be formulated in the Structured Query Language (SQL):

```
SELECT ImageId, Sum(keyfreq) as Total_Score,Count(keyfreq)
as Matches FROM imagekeys WHERE keyid IN(<keyids>)
GROUP BY ImageId ORDER BY Matches DESC, Total_Score DESC
```

KeyID	ImageID	altfreq	keyfreq
12993	1456	0	2
134564	1456	1	0
12283	1456	0	1
35691	1457	1	0
22983	1457	0	3
43563	1457	0	1
...	...	...	...

Table 2.1: An Inverted Index

This is a simplified version of the query which takes only the column `keyfreq` into account. The query returns image ids, jointly ranked by the number of keyword matches (`Matches`) and the total number of appearances of the query-keywords with the image (`Total_Score`). If we want to include the $tf*idf$ measure from equation (2.4) and both `keyfreq` and `altfreq` a query would look like

```
SELECT ImageId, Sum(0.3*keyfreq+0.7*altfreq)*importance
as Total_Score,Count(keyfreq) as Matches FROM imagekeys,
keywords WHERE imagekeys.keyid IN(<keyids>) and
imagekeys.keyid=keywords.keyid GROUP BY ImageId
ORDER BY Matches DESC, Total_Score DESC};
```

The `Total_Score` is replaced with a weighted sum of `keyfreq` and `altfreq` which corresponds to tf and multiplied by the column importance from the keyword table which contains the *idf* value.

Note that the inverted index can get quite large: for our $3 * 10^6$ images we have an inverted index table with about $60 * 10^6$ rows and we collected about $6 * 10^5$ different keywords!

The number of keywords is large indeed. One possibility to reduce it is to take only words that are listed in a common thesaurus. One of the most-used thesuari in the electronic text processing community is WordNet [22] from Princeton University. In the latest release it contains about 150 000 words along with their semantic relationships. We made some experiments, but decided to rely on our large keyword collection for the moment. One drawback is that many product or company names are not contained in WordNet.

2.3 Relevance Feedback on Visual Features

Relevance feedback is information about query results, given by a user to a retrieval system. In the context of image retrieval the user usually chooses the images that are relevant (positive feedback) and sometimes also the wrong matches (negative feedback). Our system *Cortina* includes positive relevance feedback on the visual descriptors. The user can mark images that are relevant for a given query and the visual feature vector information is used to adapt and refine the query. How text information could contribute to the relevance feedback is discussed in chapter 3. Another kind of relevance feedback is long-term relevance feedback, i.e. information for each query is stored and revoked when another user enters the same query. This is also discussed in chapter 3.

The specific choices of methods for and implementation of relevance feedback for *Cortina* were done in an affiliated work [23]. Here, we will discuss the general concepts and methods to be able to discuss the enhancements which will be introduced in chapter 3.

When a user selects relevant images we automatically obtain more information about his intents. Most importantly, if the user selects multiple images we can determine the best descriptor type for the query and set the weights in equation (2.3) accordingly. The basic concept is simple: If a descriptor type describes the query very well, all the vectors of the relevant images lie close together. To formulate this quantitatively: For each feature type f the average distance d^f_{avg} between all the relevant images is calculated:

$$d^f_{avg} = \frac{\sum_{i=1}^{M}\sum_{k=1}^{M} d^f_{ik}}{M^2}, \quad f \in \{EHD, HTD, SCD, DCD\}$$

where i, k index the relevant images and d^f_{ik} is the distance between image i and k with the distance measure defined for feature type f. The d^f_{avg} are added and normalized to the range $[0, 1]$, such that the weights can be set, e.g. for the EHD

$$\varepsilon = \frac{1 - d^{EHD}_{avg}}{(1 - d^{EHD}_{avg}) + (1 - d^{HTD}_{avg}) + (1 - d^{SCD}_{avg}) + (1 - d^{DCD}_{avg})}$$

Not only are the weights adapted, but also a new query vector is constructed based on the relevant images. A linear combination of the vector elements

from all the relevant images was chosen:

$$r_k^f = \frac{\sum_{i=1}^{M} r_{ik}}{M} \tag{2.5}$$

Where $f \in \{EHD, HTD, SCD, DCD\}$ is the feature type, r_k^f is the component k of the feature-vector of type f, and i indexes the relevant images.

The above is true for the first round of relevance feedback, i.e. after a query-by-keyword. In round n of relevance feedback the new query vector $\mathbf{r_n}$ is also combined with the query vector of round $n-1$ and the weights are adapted accordingly. The details of this process are not relevant for this work. The interested reader is referred to [23].

While constructing a new query vector for the EHD, HTD and SCD can be done as described above, the DCD needs special treatment again. The MPEG-7 standard does not include a particular procedure for relevance feedback on the DCD, so we proposed an approach ourselves.

All the the dominant colors from all the query images are sorted by their percentage and put into a source list $S = \{(\mathbf{c}_s, p_s)\}$. Starting with the color at the first entry (i.e. the one with the highest percentage) each color is selected and put into a list of selected colors $L = \{(\mathbf{c}_s, p_s, N_s)\}$, where $\mathbf{c}_s$ is a 3D color vector, p_s is a percentage and N_s is a counter. For this newly added color, all the images selected for relevance feedback are visited, i.e. their descriptors DCD_q are examined

$$DCD_q = \{(\mathbf{c}_{qi}, p_{qi})\},\ i = 1, 2, \ldots, N_q \tag{2.6}$$

If a color $\mathbf{c}_{qi}$ of DCD_q is similar to $\mathbf{c}_s$ (i.e. their Euclidean distance is smaller than T_d as it was used for equation 2.1), its percentage p_{qi} is added to p_s and the counter N is increased. The the next color is removed from S and appended to L — until S is empty. Then each percentage p_s in L is divided by its counter N_s, i.e. normalized. The k entries in L with the highest normalized percentages p_s form the new query vector.

This being a simple, greedy algorithm, the result is dependent on the order of the values in S. Ordering by percentage is reasonable, but does not deliver the best possible results. It would be better to look at the distribution of all the colors in S and define "bins". It turned out that such an approach has been proposed very recently in [24].

Chapter 3

Semantics: Combining Visual Features and Text

This chapter describes how the joint use of textual and visual features can be exploited to improve the results of a web-image retrieval system. It is the theoretical core of the present work and it introduces some new approaches to semantics in large-scale, web-based image retrieval.

3.1 The Semantic Gap

The basic challenge in the design of image retrieval systems is known as *the semantic gap*: Visual features alone are usually not suited to describe a semantic concept. Often, not even single words can describe a semantic concept. As an example consider the concept(s) *apple*. On a high level, there exist at least two semantic concepts for *apple*: The fruit and the computer manufacturer. Both of them can be split into many sub-concepts. For the moment consider the apple being a fruit. If we were asked to describe an apple visually, we might think of some slightly distorted round shape — but even the color can differ from green to yellow to red. The low level image-features can store some of this information, e.g. the color or the texture as in our system. However, the apple might appear in front of different backgrounds, might hang on a tree or even be part of an apple-pie. These problems are usually referred to as *polysemy* (i.e. a word or concept with multiple meanings) or *synonomy* (i.e. the same concept described by different words or features). Since we use global visual features in our

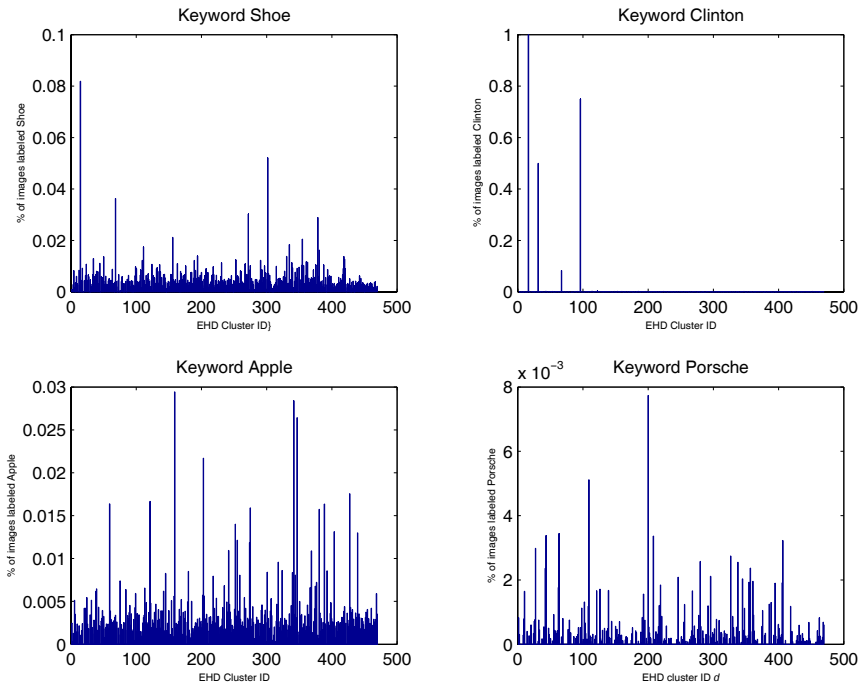

Figure 3.1: Distribution of images matching several keyword queries over the EHD clusters

system, i.e. no segmentation is done, this problem gets even harder to solve. However, remember that by collecting images from the WWW we were able to assign both text *and* several visual features to them in an unsupervised manner, as presented in the preceding chapters. The idea is to exploit these co-occurrences between text and different visual features to improve the search on a semantic level, i.e. to "bridge" the semantic gap by combining all the information we have. To illustrate that there really exists a connection between the keywords and the low level image features, consider the histograms in figure 3.1. It shows the distribution of the images labeled with different keywords over the image feature clusters - in this particular case for the EHD. It can be seen that images with particular keywords appear more frequently in some clusters than in others. Indeed, from figure 3.1 it seems that EHD cluster fourteen describes the images matching the keyword "shoe" very well. Other keywords show more and lower peaks for the edge histogram descriptor. Similar relations can be found for the keywords and the other visual feature descriptors.

3.2 Related Work

Numerous publications have introduced techniques to find relationships between keywords and images. This section gives an overview of some of these techniques and examines the applicability to our problem. Basically there are four approaches to the problem: Mapping to some joint, low dimensional spaces, model-based approaches, classification and data mining. Several works have applied a method known as Latent Semantic Analysis (LSA) or Latent Semantic Indexing (LSI) which has been introduced in [20] originally for text retrieval. LSI maps documents and queries to a lower dimensional concept space in which the retrieval takes place. The power of this technique is that it is able to identify semantic concepts rather than rely on individual keywords only. To look into LSI as a possible solution to our problem of large scale web image retrieval a short summary of the theory behind LSI is given as follows: The low dimensional concept space mentioned beforehand is created from a $terms \times documents$ matrix A by Singular Value Decomposition (SVD). An entry a_{ij} represents the number of occurrences of term i in document j. SVD decomposes A into three matrices of dimensions $terms \times m$, $m \times m$ and $m \times Documents$, respectively. m defines the size of the concept space and ranges typically from 50-350. Several works have applied LSI to image retrieval or combined text and image retrieval. In [25] visual features are given as additional "terms" to the $terms \times documents$ matrix and thus combines the visual features and textual features in the same concept space. In this particular work the technique was applied to some 3000 newspaper photographs with associated text. The method is described to perform well, however the set of images is very small. As interesting s the theory of LSI is, it has some drawbacks: the complexity of the SVD algorithm can be shown to be $O(N^2m^3)$, where N is the number of terms plus documents. In our system we have numbers of documents in the order of millions, for the terms only the number of keywords is in dimensions of 10^5. While of much smaller dimension but yet not without contribution, the dimensionality of visual features comes on top of that. This means that the SVD is nearly infeasible in a reasonable amount of time for our problem, if applied to the whole dataset. Also, the LSI process has to be redone if the document collection changes significantly, which in the dynamic environment of the WWW can happen very often. As LSI and related methods map the information to a lower dimensional semantic concept space, other approaches try to find

probability *models* that jointly describe text and visual features. Barnard and Forsyth have published numerous works on retrieval based on matching words and images, including [26, 27, 28, 29]. Most of their work is based on a model introduced by Hoffman [30]. Their model is a generative hierarchical model in form of a tree structure, i.e. each image with text is thought to be generated by following the branches in the tree model. The search process is based on the probability of each candidate image emitting the query items. This approach is described to work well with some several thousand images from the Corel data set [26] and some several thousand images from a museum database [27]. Very recently [29] the method and some variants were compared to each other also based on some ten thousand images from the Corel Data set and a text vocabulary of 155 words. The results appear to be quite good, but clearly, the amount of data tested is still very small. In addition, the authors mention that the system might be sensitive to very noisy words. According to [26] the computational requirements are "a few thousand images in a few hours" to train the model which would mean for some millions of images (as in our case) a few thousand hours if we suppose linear complexity in the best case. Other statistical models were introduced by Wang et al. in [31]. Here, hidden Markov models were use to train some hundred semantic concepts also based on the Corel data set and used to auto-annotate other images from the same Corel classes with words.

A different approach — the one we found the most practicable after investigating all the afore-mentioned approaches — is data mining: The goal is to find patterns in the data *where they exist*. i.e. it is possible that for a large amount of abstract concepts it will be hard to find patterns, however, if some concepts are well-described by some connection between features and keywords we believe that they can be identified by data mining. Traditionally data-mining literature is mostly concerned with business-data, census data or similar. Recent publications mine multimedia data for *visual* relationships in multimedia datasets, for instance in [32] the spatial relationships between texture classes in aerial images are explored. In [33] characteristics of multimedia documents like size, file-type and strongly compressed (pivoted) color and texture information were represented in data cubes — keywords where simply appended to these data cubes.

3.3 Data Mining for Semantic Clues

Our approach is the following: We mine our data to find interesting patterns and try to find semantic *rules* from these patterns to improve the search process. Semantic rules can be based on associations between text and visual features or between different types of visual features for a given concept. This choice is motivated mainly by the fact that our dataset is large, diverse, dynamic and noisy. Model-based approaches would fail because no model can be found to represent the whole dataset. Mapping to a lower-dimensional semantic space suffers from the same problem and in particular the original dimensionality and size of the dataset. The techniques we apply are *frequent itemset mining* and *association rules*. These are well-studied methods, known to scale very well. After an introduction to frequent itemset mining and association rules we will investigate how they can be applied to find semantic rules from our data.

3.3.1 Introduction to Frequent Itemset Mining and Association Rules

The concept of association rules was first introduced in [34] as a means of discovering interesting patterns in large, transactional databases. Let $I = \{i_1 \dots i_k\}$ be a set of k elements called *items*. A is a k-itemset if it is a subset of I with k elements, i.e. $A \subseteq I, |A| = k$. A transaction is a pair $T = (tid, A)$ where tid is a transaction identifier and A is an itemset. A transaction Database D is a set of transactions with unique identifiers $D = \{T_1 \dots T_n\}$. A transaction T *supports* an itemset A if $A \subseteq T$. The *cover* $cover(A)$ of an itemset A is the set of transactions supporting A.

Definition 3.1 *The support of an itemset $A \in D$ is*

$$support(A) = \frac{|\{T \in D | A \subseteq T\}|}{|D|} \tag{3.1}$$

An itemset A is called *frequent* in A if $support(A) \geq minsupp$ where $minsupp$ is a threshold for the minimal support defined by the user. An association rule is an expression $A \rightarrow B$ where A and B are itemesets (of any length) and $A \cap B = \varnothing$. Usually [1] the quality of a rule is described in the *support-confidence framework*.

[1] Several extensions have been made to association rules. Most noteworthy maybe [35] where the proper mathematical commonalities and differences between association rules

TID	Items
1	Bread, Milk
2	Beer, Diaper, Bread, Eggs
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Bread, Diaper, Milk

Table 3.1: Example: Transactions from a store

Definition 3.2 *The support of a rule*

$$support(A \rightarrow B) = support(A \cup B) = \frac{|\{T \in D | (A \cup B) \subseteq T\}|}{|D|} \tag{3.2}$$

measures the statistical significance of a rule.

Definition 3.3 *The confidence of a rule*

$$confidence(A \rightarrow B) = \frac{support(A \cup B)}{support(A)} = \frac{|\{T \in D | (A \cup B) \subseteq T\}|}{|\{T \in D | A \subseteq T\}|} \tag{3.3}$$

Confidence is a measure for the strength of the implication $A \rightarrow B$.

Note that the confidence can be seen as a maximum likelihood estimate of the conditional probability that B is true given that A is true.

Example 3.1 *The classic application for association rules is market basket data analysis. In this context, an itemset refers to a set of products. A transaction is the set of products bought by a particular customer. Consider the transactions in table 3.1. Suppose we want to find support and confidence of the famous rule* $\{Diaper, Milk\} \rightarrow Beer$*:*

$$support = \frac{support\{Diaper, Milk, Beer\}}{|D|} = \frac{2}{5} = 0.4$$

$$confidence = \frac{support\{Diaper, Milk, Beer\}}{support\{Diaper, Milk\}} = 0.66$$

The problem of finding association rules boils down to finding frequent itemsets as a first step. "Good" rules are identified in a second step, for instance

and correlation are outlined. A chi-squared test is suggested as an alternative to the support-confidence framework

all rules that have higher confidence than than the predefined threshold $minconf$.

Frequent itemsets are subject to the monotonicity property: all k-subsets of frequent $k+1$-sets are and must be also frequent. The well known APriori algorithm introduced in [36] takes advantage of this property. As discussed

Algorithm 1 APriori

```
1: i ← 1, L ← ∅
2: C_i ← {{A} | A item of size 1, A ∈ D}
3: while C_i ≠ ∅ do
4:    L_i ← ∅
5:    database pass:
6:    for A ∈ C_i do
7:       if A is frequent then
8:          L_i ← L_i ∪ A
9:       end if
10:   end for
11:   candidate formation:
12:   C_{i+1} ← sets of size i + 1 whose all subsets are frequent
13:   C_i ← C_{i+1}
14:   L ← L ∪ L_i
15: end while
16: return L
```

in [37] the complexity of the APriori algorithm is $O(\sum_i |C_i| np)$ where np is the size of the data consisting of n transactions and p items. i indexes the size of the itemsets. Note that the algorithm needs k passes through the data. Many improved algorithms for frequent itemset mining have been developed since, an overview of state-of-the-art algorithms with performance measures can be found in [38]. Most of the algorithms reduce the passes over the data to only 2 and further speed up the computations by using data structures, that make it easier to find frequent itemset candidates. We chose an implementation described in [39], which uses an array-based extension of FP-trees to achieve outstanding performance. The details of the algorithm can be found in the publication, a thorough discussion is out of scope of this work. The code was obtained from the authors. With all the improvements achieved by better algorithms the search space is still huge and a frequent

itemset of size k implies the existence of $2^k - 2$ non-empty subsets of the itemset. Instead of mining all frequent itemsets it has been suggested to mine only *maximal frequent itemsets* (MFI's) and *closed frequent itemsets* CFI's. An itemset is maximal frequent if it has no superset that is frequent. The problem with MFI's is while we know the support s of the MFI itself, we know that the support of all its subsets is at least s, but we don't know the exact value. Therefore a CFI is defined to be a frequent itemset without a frequent superset with the same support. We mined for MFI's in most cases.

3.3.2 Exploring Frequent Itemsets for Semantic Association Rules

Identifying semantics in our data set means finding correlation between text and different types of low-level image features such as color and texture. In this context each multimedia-document[2] is a transaction which consists of several keywords and some information on the visual features. While the use of keywords as items in a transaction is straight-forward the visual features need some treatment before they can be added to the transaction. The main problem is that the visual feature vectors are very high dimensional such that the number of possible item-values would be huge if each feature vector would be taken as an item, and the chance of an item occurring often i.e. being frequent would be very small. Thus we need some kind of quantization. The clustering of the low level features as described in section 2.2.1 achieves exactly this; it reduces the dimensionality and quantizes values in some range to a cluster identifier which can be added to the transaction easily. Some examples of such transactions are given in table 3.2. Note that we stored two cluster id's per visual feature type and transaction for EHD,HTD and SCD: The actual cluster the image is assigned to and the next closest. This is done to increase the frequency of the visual feature items and to increase the number of co-occurences of text and visual features. Each image appears in several clusters for the DCD because of its concept or layout, as explained in chapter 2.2. As mentioned in the introduction and quantified in the last section, our choice of frequent itemset mining and association rules as a tool

[2]Here we usually speak about images and collateral text. But we believe, that the techniques presented can be applied to video and speech or other multi-modal multimedia datasets in the same manner.

ImageID	Keywords	EHD	SCD	HTD	DCD
129977	shoe, leather, high, heel	223,14	413,555	2,399	12,44,321,4,7
129978	shoe, brown, formal	223,37	455,25	16,246	67,97,33
129979	shirt, linux, penguin, nerd, shop	21,56	23,67	46,2	87,231,222,34
129980	suit, formal, tuxedo, brioni	88,271	123,321	265,333	345,23,233,123,56
...	...	...	...	...	...

Table 3.2: Example: Images as transactions

to identify semantics is motivated by the comparably low computational complexity, i.e. the scalability to large datasets as ours. But the method has additional, beneficial characteristics:

- Association rules can be analyzed by humans in most cases. Unlike neural networks or the semantic concept space of LSI, which are very hard to interpret.
- Rules can be added or deleted: Rules for additional data can be added, human users can remove incorrect rules etc.
- Instead of editing rules manually, the information obtained by humans can be integrated from long-term relevance feedback: Stored usage data (i.e. which images were selected for certain queries) can be analyzed to add additional rules or to weight existing rules.
- It is possible to implement a system which does not need to re-analyze the whole dataset when new data are added.

Like every method association rules have some disadvantages which should be mentioned, too:

- It is important to note that association and correlation is not the same. In particular, association rules with the support/confidence framework can find only positive correlations. A thorough discussion of this subject can be found in [35].

- If we want to discover interesting rules with low support the dataset has to be analyzed with a low minimum support threshold which takes long and leads to the generation of many uninteresting rules [3].

While mining the transactional data as in table 3.2 we met several challenges. The first is given by the distribution of the data over the low-level feature clusters. Figure 2.3 depicts this distribution for the EHD, HTD and SCD descriptor types respectively. The distribution is very uneven and has a range from a few images per cluster to over 10^5 images per cluster. (We believe this is the result of two factors: First, the stopping criterion for the K-Means algorithm was not zero and maybe too high. But the lower the stopping criterion the longer the clustering process. Second, samples of the data and not the whole dataset was used to determine the centroids for the clusters). Clearly, this distorts the information obtained from frequent itemset mining: If a cluster is very large, the a-priori probability that an image for any semantic concept is located in this cluster is high, which leads to strong but uninteresting rules. On the other hand, if a cluster is very small it gives us close to zero information. To overcome this problem, large clusters were re-clustered by individually giving them as an input to the k-means algorithm. This way each cluster larger than $2 * 10^4$ was split into smaller clusters of about 8000 images per cluster. This is obviously a crude approximation, but as it can be seen from figure 3.2 it helped to lower the variance of the cluster sizes, in particular the new clusters with ids larger than 400 are nearly evenly distributed.

Another problem was the strength of associations within text versus text and low level feature clusters: It turned out that in our data the associations between keywords are much stronger than the associations between keywords and the low level feature clusters. That means, to discover patterns consisting of text and visual information the frequent itemset mining

[3] However, in [40] methods are described to efficiently find multiple level association rules, i.e. rules on higher and lower semantic levels. The drawback is that the methods supposes some predefined hierarchy-structure between high-level and low-level concepts. In our database such a hierarchy is not available.

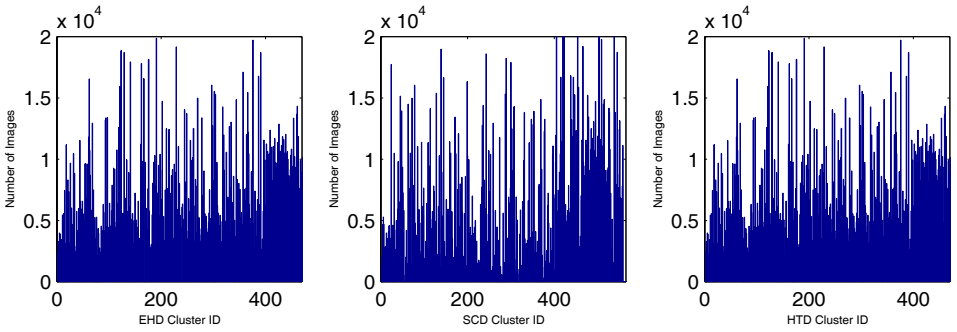

Figure 3.2: Large clusters were re-clustered into smaller ones.

has to be done with extremely low minimal support threshold, which affects runtime and scalability.

Thus, we redefined the problem: For the moment we are interested in rules that contain keywords and visual feature clusters. In particular, we want to have rules that consist of one keyword and some visual feature cluster ids. Thus, we can do the frequent itemset mining per keyword, i.e we load all the images which match a given keyword and assign the association rules found from this data to the keyword. We have the information on which images are matched for each keyword in the inverted keyword-index discussed in section 2.2. The transactions can be obtained from this table and a table that lists the cluster identifiers for each image very easily. In addition, using this approach we find also rules for keywords that appear rarely — if we did the mining just over the whole dataset, the information how often a keyword appears in the whole dataset would influence the frequent itemsets, i.e. they would contain mostly information we are not interested in.

Table 3.3 gives an overview over the characteristics of the frequent itemsets we found for Maximal Frequent Itemsets (MFI), respectively. By doing frequent itemset mining per keyword as described above, the number of transactions increases to about $41 * 10^6$. However, the transactions are short, in fact the maximal length is either 7 or 15 depending if the DCD is used or not (1 keyword id followed by the clusterids). Keywords, that generated less than 300 transactions (i.e. less than 300 images are labeled with the respective keyword) are disregarded. This way 12407 keywords where analyzed. The minimal support thresholds are set per frequent itemset mining *per keyword*. The column "long rules" stands for itemsets that

min. support	# of rules	# long rules	avg. abs. supp.	runtime [s]
0.01%	21 818 132	13 458 731	2.48	3 541
0.1%	8 582 929	3 714 033	11.24	2 769
1%	2 726 716	79 685	24.28	1 688
10%	135 336	986	37.03	1 493

Table 3.3: Frequent itemset mining results. MFI per keyword, for 41 781 266 transactions in total.

contain several types of low-level feature clusters, e.g. EHD and SCD (and obviously the keyword, since that is appended to every frequent itemset). I we applied CFI instead of MFI it would result in more frequent itemsets, since the MFI are per definition a subset of the CFI.

If we successfully found association rules which join text and visual features, it turned out that many of them are semantically useless: In particular the DCD, since applied to whole image, was distracted by background colors. It turned out that in the DMOZ shopping category, where we collected most of our data from, there are many objects on a white background. Thus, the system identified many strong rules between objects like "shoe" etc. from shopping pages with the DCD cluster of dominantly white color. These images could have been segmented quite easily but since for the moment we rely on global feature, we just had to exclude the DCD in most of the association rule discovery.

From Frequent Itemsets to Association Rules

When we want to discover semantic relationships in our data, the most interesting associations are those that connect (key)words and visual features. Suppose we found an MFI consisting of a word and image features. We can either take the word as antecedent of our rule, or we can take the image features as antecedent.

Example 3.2 *One of the very strong MFI we found in our database was* $\{EHD249, shirt, EHD310, SCD493\}$*, which appeared in 2160 transactions. We can either define a rule* $\{shirt \rightarrow EHD249, EHD310, SCD493\}$ *or* $\{EHD249, EHD310, SCD493 \rightarrow shirt\}$*. Support and confidence are*

$$support = \frac{support\{EHD249, shirt, EHD310, SCD493\}}{|D|}$$

$$= \frac{2160}{3 * 10^6} = 0.00072$$

$$confidence = \frac{support\{EHD249, shirt, EHD310, SCD493\}}{\#\ images\ labeled\ with\ "shirt"}$$
$$= \frac{2160}{43040} = 0.05$$

and

$$support = \frac{support\{EHD249, shirt, EHD310, SCD493\}}{|D|}$$
$$= \frac{2160}{3 * 10^6} = 0.00072$$

$$confidence = \frac{support\{EHD249, shirt, EHD310, SCD493\}}{\#\ images \in \{EHD\,249 \cap EHD\,310 \cap SCD\,493\}}$$
$$= \frac{2160}{2485} = 0.86$$

The results from example 3.2 are visualized in figure 3.3 along with another MFI $\{shoe, EHD14\}$. The antecedent and consequent of the word-image-feature rules are shown in the x-y-plane. The z-axis represents the support of each rule, and the colors of the bars show the confidence. Noteworthy are

1. The low support of all the rules: Most of our rules have support lower than 0.1%!

2. The high confidence for the rule $\{EHD249, EHD310, SCD493 \rightarrow shirt\}$. Rules like that could be used to auto-annotate images, i.e. assign keywords to images based on their features.

3. There are also rules $shirt \rightarrow EHD14$ and $EHD14 \rightarrow shirt$. These are only shown to complete the picture. Their support is too low to be discovered by frequent itemset mining in general. The potential rules $\{EHD249, EHD310, SCD493 \rightarrow shoe\}$ and $\{shoe \rightarrow EHD249, EHD310, SCD493\}$ do not exist, i.e. they have support zero.

Point 1 can be further commented: If we consider the large number of possible keywords in our collection and the relative short "documents" consisting of only some 10-50 words, it becomes clear why the support cannot be high:

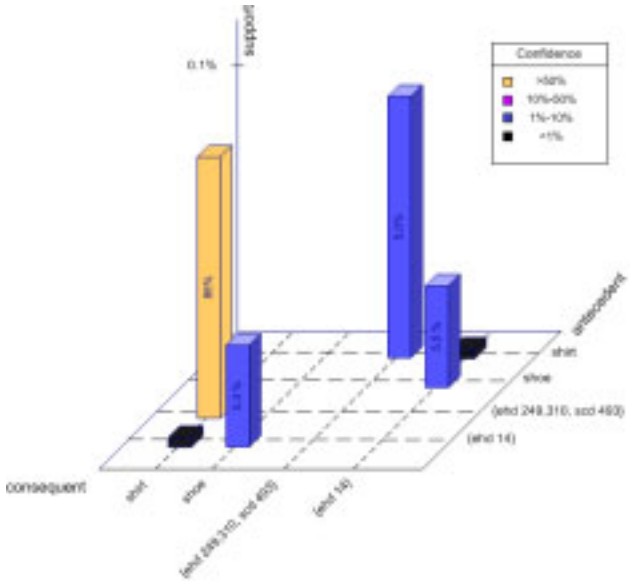

Figure 3.3: Several rules deduced from MFI in our database and their support and confidence

Let's say we consider only 150000 Words listed in WordNet [22] significant and we know that our inverted keyword index contains $60 * 10^6$ rows then this makes in average about 400 images (transactions) per keyword — which is compared to the total number of transactions of $3*10^6$ already very small, namely 0.25%. If we want to increase the number of relations between image features and keywords we could increase the number of keywords per image by collecting more, but probably the precision of the keyword annotation would decrease. Besides the support, the confidence needs to be considered, too. In table 3.4 the confidence for the values from table 3.3 is shown. Only rules $keyword \rightarrow \{featureclusters\}$ were analyzed. The columns list how many rules have a confidence higher are higher than the respective value. More illustrative than the absolute values are the percentages of the number of frequent itemsets (equivalent to rules in this case), which shown in figure 3.4. There is a trade-off between lower support and longer rules or higher support which results in higher confidence for our setup, since the antecedent (the keyword) stays the same, the consequent changes to shorter but more frequent items for higher support. It is interesting, that the 1% series seems to have more items with higher confidence than 20% and 40%, the 10% series seems to be shiftet to the left from that, i.e. more rules have

min. supp.	# conf. >1%	# conf. >5%	# conf. >20%	# conf. >40%
0.01%	504	0	0	0
0.1%	6531	600	0	0
1%	1 149 307	147 048	99 100	77 071
10%	58993	7124	340	33

Table 3.4: Support "distribution" for the frequent itemsets from table 3.3

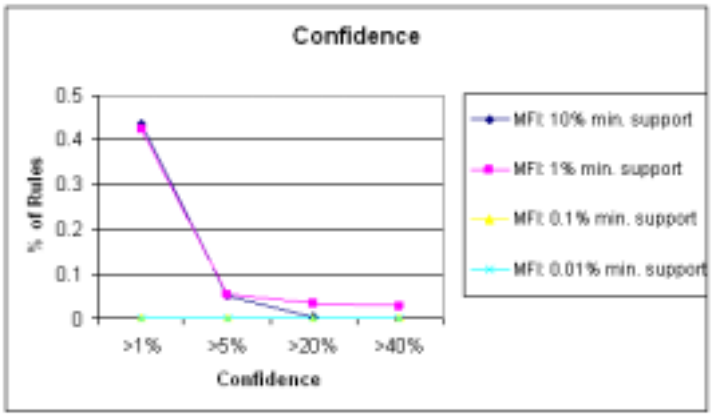

Figure 3.4: Confidence for different minimal support thresholds.

confidence around 1%. This needs to be investigated further, which was not possible because of the limited time.

The most important question in association rule discovery is how to judge the interestingness of a rule. Criteria for interestingness can be either purely statistical measures or can rely on background knowledge about the data items. The latter we included already in the process of rule discovery and even itemset mining: Remember that in the previous section, we decided to do frequent itemset mining per keyword, since we are mostly interested in rules that connect keywords and image features at the moment. Another question is the length of the rules: Are we interested in long rules (which have lower support by the nature of the problem) or are we interested in shorter rules with higher support and confidence? The long rules certainly give more information about the semantic relations in that they associate keywords with texture and color. But the question remains if this information is significant enough to build on. Lastly, it depends also on the application. Purely statistical criteria can be used to identify rules with high information

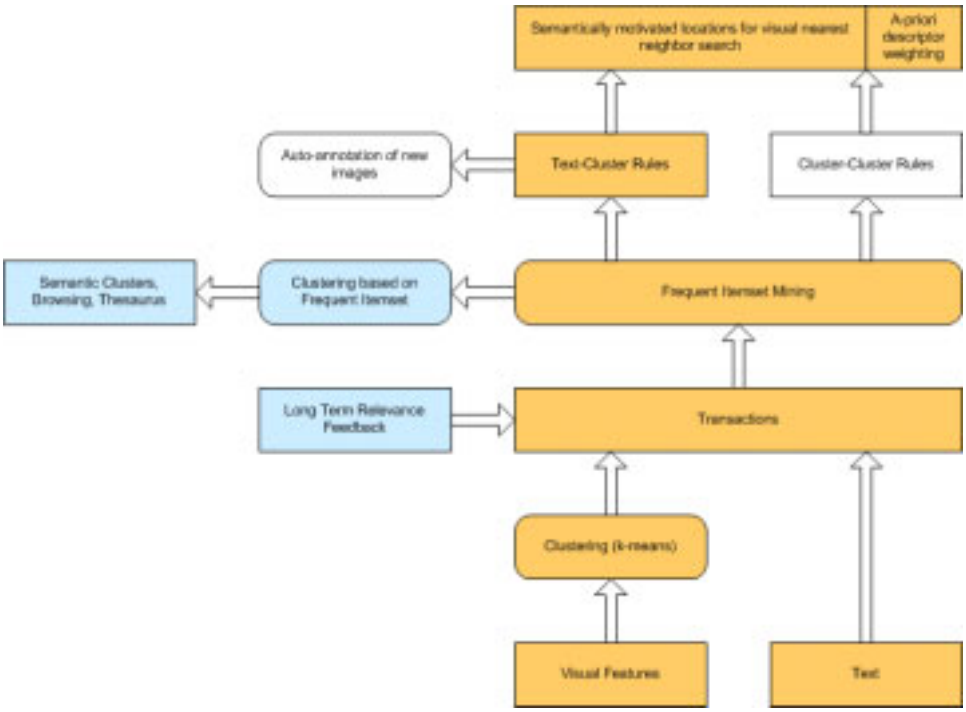

Figure 3.5: How frequent itemset mining can be applied to gain insights into semantics from different perspectives

content independent from the specific application. Given the time limits, we used our background knowledge on the data and the support of an itemset to chose our rules. In chapter 7 we will make a suggestion on a statistical measure to be used to refine the rule selection. In the next section we will discuss how these types of rules can be used in web image retrieval systems.

3.3.3 Exploiting the Semantic Rules

The different kinds of semantic rules can be exploited in several ways to obtain improvements for image retrieval systems. Some of them options are depicted in figure 3.5. Elements in orange color are discussed and implemented in the system *Cortina*, elements in blue are discussed but not implemented in the current version and elements with white background are further options that are not discussed in detail in this thesis.

Identifying Interesting Low-Level Clusters

In our system, an initial keyword search is followed by a visual nearest neighbor search with relevance feedback. Obviously, nearest neighbor search that relies on visual feature information only, cannot succeed in returning exclusively semantically relevant images. Consider the results from a pure visual nearest neighbor search in figure 3.6. The query image is the one on the top left, the initial keyword search was "shoe". Two things are noteworthy: While the first couple of results are shoes indeed, the other ones are not shoes and appear only because of their visual similarity. The immediate idea is to return only images which are visually similar *and* contain the keyword that was searched for ("shoe" in this example). However, our experiments have shown, that since the keyword annotation is automatic and unsupervised, often the matching keyword does not appear with the semantically and visually closest images. This is illustrated in figure 3.6 where the keywords for the 10 visually closest neighbors for the image on the top left as query are shown. Obviously objects that aren't shoes don't have the keyword shoe, but note that 3 out of 6 images depicting a shoe don't have the keyword "shoe" assigned to them either. Eliminating images lacking keywords of the original query would thus eliminate many correct matches.

As an alternative we introduce an approach which points the system to places where the chance of finding images for a certain semantic concept is high. Then the results are re-ranked not only on their visual similarity but also on the amount of keywords they share. Currently, we consider a single keyword as the semantic concept and link it to the low level clusters based on simple association rules of the form

$$\begin{aligned} \{Keyword\} \quad \rightarrow \quad & \{\{EHDCluster_1, \ldots, EHDCluster_n\} \\ & \{SCDCluster_1, \ldots, SCDCluster_m\}, \\ & \{HTDCluster_1, \ldots, HTDCluster_k\}\}, \\ & |n|, |m|, |k| \geq 0 \end{aligned} \tag{3.4}$$

Remember that in chapter 2.2.1 clusters were made highly overlapping to capture the nearest neighbors in case they were spread over the borders of the original clusters. The problem with this approach is, that it is not very scalable. Now, consider the situation depicted in figure 3.7.

The feature vectors for a group of images belonging to a semantic concept are spread over several clusters. With association rules as presented above,

Figure 3.6: The 10 visually closest images for the image on the top-left as the visual query after an initial keyword search for "shoe". (Columnwise, i.e. left column contains k-NN 1-5, right column 6-10.)

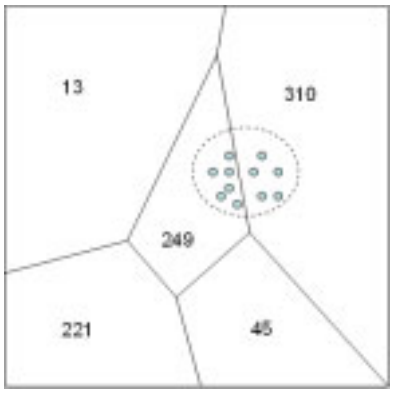

Figure 3.7: Images belonging to one semantic concept spread over two clusters

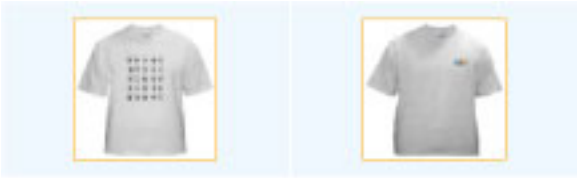

Figure 3.8: A shirt from EHD cluster 249 (left) and one from EHD cluster 310 (right)

there is a chance that we discover this situation and can take both clusters into account for visual nearest neighbor search without the need to create large overlapping clusters. In fact, it turned out that this is really the case if there is a strong enough connection between a keyword and some low level visual feature clusters. For example frequent itemset mining returned the itemset $\{EHD249, shirt, EHD310\}$ which as a rule can be $\{shirt \rightarrow EHD249, EHD310\}$. It turned out that EHD clusters 249 and 310 really contained a lot of similar images of t-shirts, as shown in figure 3.8.

To sum it up, the steps for similarity search are the following:

1. Look up the association rules for the original query-keyword.
2. Calculate the distances from the query-image to the centroids of the set of clusters C_A given by the association rules.
3. Load the n closest clusters from C_A and the cluster closest to the query vector if it is not already given by the association rules. (We usually set n to 3).
4. Do a similarity search on all these clusters.

As mentioned in section 2.2.1 the selection of clusters in high dimensional spaces based on the minimal feature vector distance might not always give correct results, since high dimensional spaces show differences in behavior compared to three dimensional spaces. The method to select candidates from the clusters based on their semantic similarity obtained from meta-data could be a general, pragmatic method to avoid some of the problems in high-dimensional spaces.

In our case, the approach works well when there is a good connection between a single keyword and low level features, i.e if the semantic concept

keyid	ehds	scds	htds	support
24782	14,			870
24782		421,		837
24782		418,		678
24782		536,		606
24782	302,			496
24782		501,		457
24782		453,		418
24782	379,			398
24782	335,			271
24782	272,			268
24782	419,			247
24782	418,			235
24782	355,			229
24782	417,			181
24782	421,			180

Table 3.5: Top 15 maximal frequent itemsets for the keyword "shoe". (In total, 15639 images are labeled with the word "shoe")

is well-described by this single keyword and the semantic concept can be expressed by some low level feature characteristics. For instance "shoe", "shirt" and similar objects can be described this way, more abstract concepts like "clinton" or "holiday" can not be expressed by low level features in general, which means that association rules for these cases will be less precise in general.

A-priori Weights for Relevance Feedback

The association rules collected for each keyword can not only be exploited to define the search space, they also contain information on how well a visual descriptor type describes the keyword or the semantic concept related to it. For instance, table 3.5 lists the top 15 maximal frequent itemsets found for the keyword shoe (id 24782). Most of the rules connect the keyword with an EHD cluster, some of them connect the keyword with an SCD cluster and no connections seem to exist for the keyword and the HTD clusters. From this one can argue that the images labeled with "shoe" in our database

are best described with the edge histogram descriptor or the scalable color descriptor. A look at these images in our database actually shows that they are most often catalog-like images with a shoe on a white background. This is described well by the edge histogram since in this case the distinctive shape of shoes is captured by the descriptor, the white background gives the coherence for the scalable color descriptor. This means we can preset the weights for formula 2.3 for the combination of the individual descriptors for the first round visual nearest neighbor search - we call this the *a-priori weight*. In this case we would set the EHD the highest, some medium value for the SCD and a very small value for the HTD.

If only one image is selected for the first round of visual nearest neighbor search, the weights can be taken as obtained from the frequent itemsets. If several images are selected we have to combine the a-priori weights and the weights obtained from the information of the selected images. We just initialize the weights for iteration 1 of relevance feedback with our a-priori weights, as if there had been an iteration 0 of relevance feedback.

Re-Ranking Visual Results with Text

Showing the results of a visual nearest neighbor search ordered only by their visual distance is suboptimal. The results should be sorted based on their overall or semantic similarity, i.e. the keywords should be considered, too.

As it has been shown in the preceding sections (see also figure 3.6) not all the images in the result-set of a visual nearest neighbor search are labeled with the original query keyword(s). The introduction of association rules to identify clusters which contain many images labeled with the original query keyword(s), and the expansion of the search to these clusters increases the probability of finding images which are labeled with the query keyword(s) within the firs k retrievals. However, the semantically close results might still be visually farther from the query than some wrong matches. To rank these results higher, we rely on the keywords. We do not only take the original query keyword(s) entered by the user, but also all the keywords assigned to the image selected by the user as input for relevance feedback. This gives us a set of ranking keywords $\mathbf{R}$. Then we simply divide the visual distance by the number of common keywords of the query images $\mathbf{q}$ and the

retrieval candidates c_i, i.e.

$$d_{sem}(\mathbf{q}, c_i) = \frac{d_{vis}(\mathbf{q}, c_i)}{|keywords(c_i) \cap \mathbf{R}|} \quad (if \ \ |keywords(c_i) \cap \mathbf{R}| \neq \emptyset)$$

where $d_{vis}(q, c_i)$ is the visual distance between the query vector $\mathbf{q}$ and the retrieval candidates c_i.

This improves the correct ordering of the results significantly — however only if the query and the candidates share common keywords. In fact, a query and a candidate might share some common keywords such as "size" which can appear in many contexts and describes no clear semantic concept. Another candidate might have no common keywords with the query but a different keyword which describes the correct concept (polysemy) — the wrong image gets ranked higher in this case. As an example consider figure 3.6 again: The image on the top right is labeled with "walk" which is clearly semantically close to the query ("shoe") but does not appear in the list of query keyword(s).

As one solution to this problem we suggest to use semantic distances between keywords instead of direct matches. These distances can be obtained from a thesaurus like WordNet [22]. In fact, there exist very recent proposals [41] and even publicly available implementations to find these semantic distances between words.

In general one might also further investigate the possibility to include the keyword ranking somewhat tighter into the relevance feedback process.

Semantic Clusters from Association Rules

Suppose we want to sort our documents in a semantically meaningful way based upon the combination of textual *and* visual features. The benefit could be a organization of the data which allows semantic browsing, or another "layer" of clustering on top of the visual clusters which incorporates bot visual and text information. If we had such a clustering, the search space could be limited to these semantic clusters and visual nearest neighbors would only be retrieved from a set of images that are semantically relevant.

We want to achieve this with the frequent itemsets found in the previous sections, i.e. we want to follow the option of *semantic clustering* as shown in figure 3.5. We discovered that Ester et al. recently introduced several algorithms to cluster text data based on frequent itemsets or *frequent terms*, as the authors call them in the context of text documents [42, 43]. As an

experiment we wanted to see how well the method performs if we applied it to our dataset which means also to extend the context of the method from pure text clustering to multi-modal data clustering. In [42] two simple clustering algorithms are introduced, one of them builds a hierarchy the other just independent clusters. In [43] the method is improved with better algorithms. Since we are interested in the general applicability of the method we chose the simplest algorithm.

All the algorithms rely on a frequent itemset mining of the dataset. Each frequent itemset is a candidate for the "core" of an additional cluster. The clusters are built under the restriction that the *overlap between them is minimized*, which is the only criterion for clustering. Note that thus no distance measure needs to be defined, which is very useful since defining distances (e.g. what is the "semantic" distance between two images which includes both text and visual feature information?) for multi-modal datasets can be very difficult.

Let $F = \{F_1, \ldots, F_k\}$ be the set of all frequent itemsets in our database $|D|$. Let f_j bet the number of all frequent itemsets supported by document D_j

$$f_j = |\{F_i \in R | F_i \subseteq D_j\}|$$

where R is a subset of F, the subset of *remaining frequent itemsets*. Since each $F_i \in F$ is a candidate for the creation of a new cluster C_i, R is the set of F_i which have not yet been defined as a cluster. The *entropy overlap* of cluster C_i with the other clusters is defined as the distribution of the documents in C_i over all the remaining cluster candidates:

$$EO(C_i) = \sum_{D_j \in C_i} -\frac{1}{f_j} * log(\frac{1}{f_j})$$

$EO(C_i)$ is zero if no document in C_i supports any other frequent term set, i.e. $f_j = 1 \; \forall \; D_j \; \in \; C_i$. It increases monotonically with increasing f_j.

A greedy algorithm FTC is defined in [42], and shown in algorithm 2. In each iteration the algorithm selects the candidate F_i from the set of remaining frequent itemsets R, whose cover has the minimum overlap with the other clusters, and defines it as a new cluster C_i. Then the documents in the cover $cover(F_i)$ are assigned to C_i.

The algorithm FTC was applied to a subset of our data, namely 80000 images from the DMOZ clothing category. Frequent itemset mining was

Algorithm 2 FTC: Frequent Term-based Clustering

1: $SelectedItemSets \leftarrow \{\}$
2: $n \leftarrow |D|$
3: $R \leftarrow DetermineFrequentItemsets(D, minsup)$
4: **while** $cover(SelectedItemSets)| \neq n$ **do**
5: **for all** $F_i \in R$ **do**
6: Calculate overlap for F_i
7: **end for**
8: $BestCandidate \leftarrow F_i$ with minimum overlap
9: $SelectedItemSets \leftarrow SelectedItemsets \cup BestCandidate$
10: $R \leftarrow R \setminus BestCandidate$
11: Remove all documents in $cover(BestCandidate)$ from D and from $cover(F_i \in R)$
12: **end while**
13: return $SelectedItemsets$ and $cover(F_i) \; \forall \; F_i \in SelectedItemsSets$

again performed with the fpmax* algorithm [39]. As in the previous sections it turned out that the associations within the text are stronger than associations between text and visual features. We thus chose only the frequent itemsets as input for the FTC algorithm, which consisted of keywords and visual feature information. It turned out that the most frequent combinations of text and low level features contained words words like *price, item, size, color* etc., which are semantically meaningless. Removing these words gave some benefit, e.g. we found many semantic clusters for "shirt" which appeared in different contexts, i.e. different low level feature clusters. However, in general the associations were not strong enough to enable a semantic clustering or browsing hierarchy. I believe that the method might bring good results for other multi-modal datasets, e.g. biological images with meta-data. Thus I suggest further research into applying Ester et. al.'s text-algorithms for multi-modal multimedia datasets.

3.3.4 Summary of Semantic Improvements for Visual NN Search

Several semantic improvements to visual nearest neighbor search have been suggested in the preceding sections. The improved visual nearest neighbor search process in our system can be summarized as follows:

1. Look up the association rules for the initial query-keyword.

2. Calculate the distances from the visual query-vector q to the centroids of the set of clusters C_A given by the association rules.

3. Load C_s: The n closest clusters from C_A and the cluster closest to the q_v if it is not already given by the association rules. (We usually set n to 3).

4. Set the a-priori weights of the descriptors according to the information given by the association rules.

5. Do a similarity search on all these clusters.

6. Re-rank the results based on the initial query-keyword and the words obtained from the selected images

These improvements lead to semantically more relevant results for visual nearest neighbor searches, based on text assigned to the images. This will be quantified in chapter 5. In the next section we suggest a method which achieves the opposite: The improvement of keyword-searches with visual features.

3.4 Improving Text-Based-Search with Visual Features

Since we have both textual and visual features it would be interesting to see if the visual features can be used to support query-by-keyword. I suggest a re-ranking of the query-by-keyword results based on visual feature information. To my knowledge, no web image retrieval system exists that uses visual features as a direct support for query-by-keyword.

3.4.1 The Visual Link

In this section, a new, graph-based method is described which achieves a ranking of query-by-keyword results based on visual features *and* text as opposed to the commonly used text ranking. The basic concept is the following: Let $\mathbb{I}(Q)$ be a set of images matching a keyword query Q. Let d_{ij}^f be

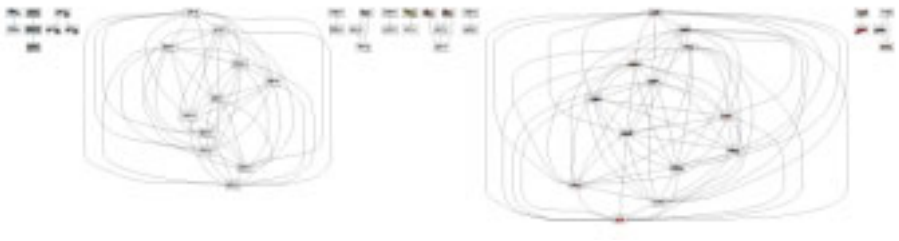

Figure 3.9: A typical graph with visual links.

the (euclidean) distance between the feature vectors i and j of visual feature type f. Then an undirected graph $G(V, E)$ can be defined as follows:

$$\begin{aligned} V &= \{i \mid i \in \mathbb{I}(Q)\} \\ E &= \{\{i,j\} \mid d_{ij}^f < d_{thresh}\},\ f \in (SCD, EHD) \\ w_{ij} &= \frac{1}{d_{ij}^{EHD} * d_{ij}^{SCD}}\, if\, \{i,j\} \in E \end{aligned} \tag{3.5}$$

where d_{thresh} is a minimal distance threshold which is defined based on the distribution of the distances for feature type f and w_{ij} is the weight assigned to edge $\{i, j\}$. In other words: images that are separated by a distance smaller than d_{thresh} are connected by an edge which can be thought of as a *visual link*. The edge weight is the inverse of the product of the distances. Links can be created based on several feature vector types f. In our system one color descriptor and one texture descriptor were chosen, namely the Scalable Color Descriptor and the Edge Histogram Descriptor. Note that the number of distance calculations is

$$\frac{N * (N-1)}{2}, \qquad N = |\mathbb{I}(Q)| \tag{3.6}$$

Recall the following concepts of basic graph theory: A graph is *connected* if there is a path from any point in the graph to any other point. Otherwise the graph is said to be *disconnected*. A group of vertices in a disconnected graph that is reachable from one another is called a *connected component*. In our case, a choice $d_{thresh} < d_{max}$, where d_{max} is the maximum distance between two feature vectors in V, results in a disconnected graph G. The number of connected components increases with decreasing d_{thresh} and reaches a maximum of $|V|$ for $d_{thresh} = 0$ if all the elements in $|V|$ are distinct. Each connected component $C \in G$ consists of a set of visually similar images, based on the feature types f. An example of a Graph can be seen in figure

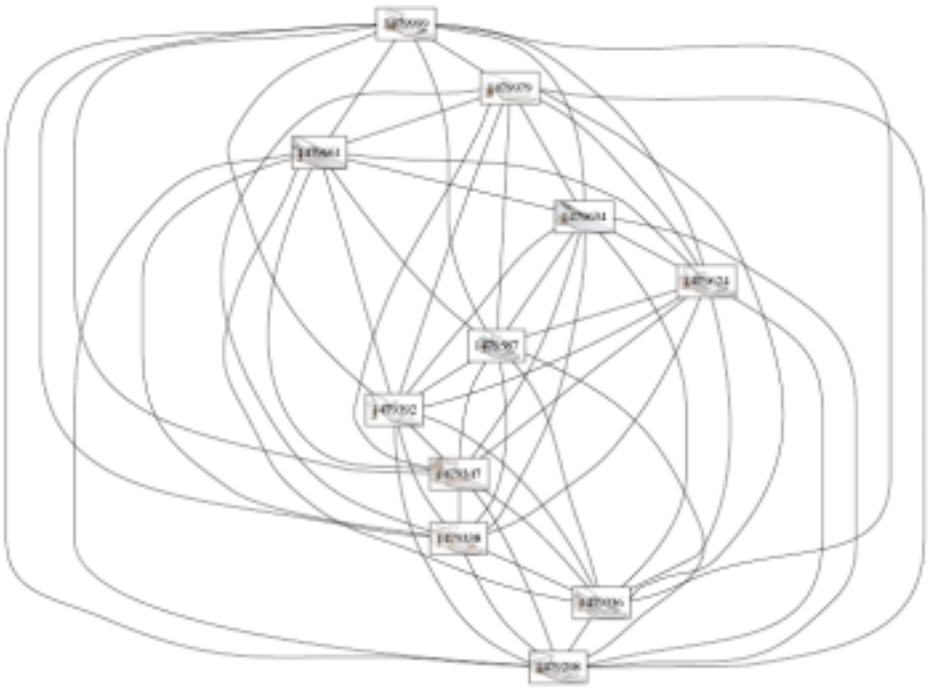

Figure 3.10: A closer look at a connected component

3.9. (The example was built with a reducet set of results from a keyword query. The actual graphs are too large to visualize them easily.) A closer look at one of the connected components is given in figure 3.10. It contains a particular set or concept of shoes, namely white woman's shoes.

An analysis of the graph G will now be done under the following assumptions:

- If a component C is large, i.e. it consists of *many* similar images for the keyword query Q, it represents the semantic concept of the query Q better than a very small component.

- The inherent average similarity of the component C is a measure for its quality or consistency.

- One or several representatives r_C can be chosen to summarize the concept described by the component C.

This leads to the following simple algorithm for a semantic analysis of G:

Algorithm 3 Find Representatives

```
1: for each connected component C ∈ G do
2:    r_C ← 0
3:    for all v ∈ {C, V} do
4:       if weight(v) > weight(r_C) then
5:          r_C ← v
6:       end if
7:    end for
8:    {r_C} ← r_C
9: end for
10: return {r_C}
```

The weight function $weight(v)$ is defined as follows:

$$weight(v) = \left(\sum_{e \in edges(v)} \frac{weight(e)}{degree(v)} \right) * keywordscore(v)$$

The weight depends on the normalized similarity of each node to it's neighbors, i.e. a node which has a high similarity to all of it's neighbors is thought to be very "central" for this component. In addition, the original keyword score of the image (the node) is taken into account, since we don't want to loose this valuable information.

Before the list of representatives $\{r_C\}$ is returned to the user, we suggest a ranking of the representatives:

$$rank(r_C) = f(size(C), avgdegree(C), avgkeywordpos(C))$$

A larger component ($size(C)$) is more important than a smaller one, the average degree $avgdegree(C)$ is a measure for the similarity within the component and the $avgkeywordpos(C)$ is the average keyword-ranking of all the documents in component C. For f we chose a simple, weighted linear combination of $size(C)$, $avgdegree(C)$ and $avgkeywordpos(C)$.

3.4.2 Approximations for Faster Graph Construction

While the approach as presented above is very compelling, it is hard to run it fast enough to present the results to the user within a reasonable amount of time. In fact, a keyword query can return several thousand matching

ImageID	ClusterId	2ndClusterId	ReldDist
28821	33	12	0.6
28822	245	23	0.2
28823	1	6	0.5
28824	222	444	0.9
28825	3	199	0.3
...	...	...	...

Table 3.6: Relational DB table to approximate distance calculations

images and the number of distances to calculate gets very high as given in equation (3.6). Computationally demanding are in particular

1. The loading of the feature vectors. Since the images are not all visually close, the clusters as presented in chapter 2 can not be used. Looking up and loading the individual feature vectors from the database is required. The image identifiers of the feature vector table are obviously indexed but matching them with the image table and loading the vectors (from different locations on the hard-drive) is very time-consuming.

2. The calculation of the distance itself.

Thus, the following approximation was introduced, which reduces the computational time to a minimum. A table as shown in 3.6 was added to the relational database.

Instead of checking if the distance between two images falls below the threshold d_{thresh} as introduced above, we first check if the images are in the same cluster using the column $ClusterId$. If they are in the same cluster, we check if their relative distance to the cluster's centroid is in the same range r. The relative distance in column $RelDist$ is the distance to the cluster's centroid divided by the distance to the cluster member that is the farthest from the centroid - which is a measure for the size of the cluster. As shown in figure 3.11 we capture all the vectors within a layer of thickness r in the hypersphere around the centroid of the cluster.

Further improvements where achieved by setting r not uniformly but based on the distribution of the relative distances to the cluster centroids $RelDist$: Consider figure 3.12 which shows this distribution. (Note that

Figure 3.11: Candidates C within a layer of thickness r around the query Q, and centered at the cluster centroid are considered similar, i.e. linked.

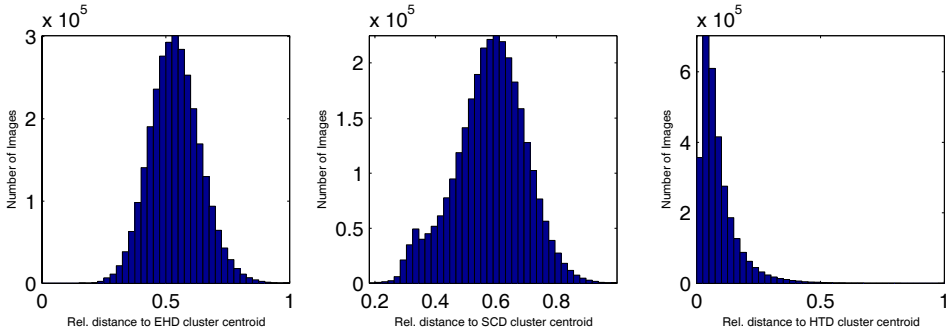

Figure 3.12: Relative distances of feature vectors to their cluster centroids

the distributions show the same form as the overall distance distribution in figure 2.5). We chose a small r if the $RelDist$ values of both images to compare were within the peak area of the histogram and some larger value if at least one of the images was at the border of the histogram.

To further improve the approximation one could take the angle within the layer r into account. We can obtain it from the cluster that is the next closest for the given image, which is stored in the column $2ndClusterId$. Some characteristics of this approximation are noteworthy:

- We don't find all the links that should exist since images that lie at the borders of a cluster may have very close neighbors in the adjacent

clusters.

- The use of the next-closest cluster to approximate the angle is a rather strict measure, especially if the query is close to the perpendicular bisector between tow nieghboring clusters. Also, it is questionable how good a measure of direction this is in high-dimensional spaces.

- In contrast, within the cluster the relative distance range r can be adapted to capture distances within any desired range. However, without the restriction of the angle it introduces "wrong" links to vectors on the "opposite side" of the hypersphere.

The required table can be calculated off-line rather easily. It can even be integrated into the clustering procedure at nearly no extra cost.

We observed that in very rare cases too large connected components are created whose inner similarity is not strong enough any more. A simple solution to this problem would be to calculate the minimum spanning tree and the (feature-)distance between the images at the end and start points, respectively. If the distance is larger than a threshold value, the component could be either split or several representatives along the minimum spanning tree could be chosen instead of only one.

Some figures are presented to illustrate the results. In figure 3.13 the first 16 images for the result of the query "shoe" are given. In figure 3.14 the result for the same query but this time with our semantic search as introduced above is shown. It can be seen that the noisy, pink images are reduced to one image. 14 out of 16 images are shoes. The system increases diversity and significance on the first page. In figure 3.15 a search for "apple" is shown that produces only noise on the first page of results. This happens when many identical images with the same keyword but from different locations are collected. In figure 3.16 the same query summarizes the noisy results to one image. Note that on the first page both the oncepts "Apple Macintosh" and "Apple the fruit" are present. One could argue that images that appear very often and look the same could just be removed from the database. But it is difficult to do that in an unsupervised manner and for every possible keyword. Also, often the same image appears in different contexts, i.e. on different pages with different additional keywords which could be a potentially interesting information about semantics.

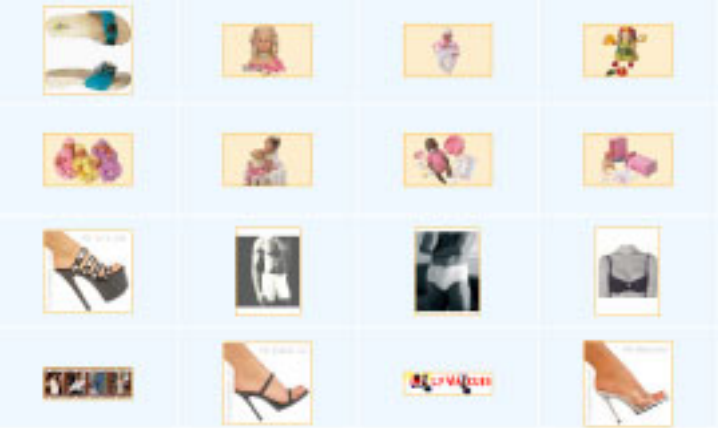

Figure 3.13: First 16 results for query "shoe" before connected component analysis

Figure 3.14: First 16 results for query "shoe" after connected component analysis

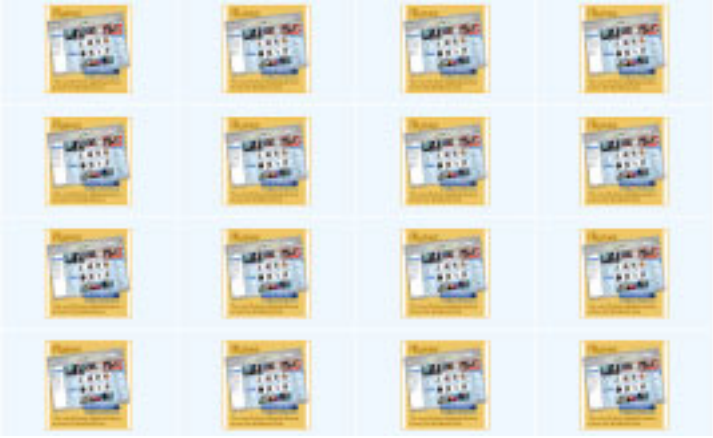

Figure 3.15: First 16 results for query "apple" before connected component analysis

Figure 3.16: First 16 results for query "apple" after connected component analysis

Chapter 4

Software Design

A substantial part of our work consisted of implementing the system and methods described in the previous chapters as a working system, which could be scaled to even larger dimensions than it is now. This chapter describes how such a system can be designed from the software and computer system point of view. Figure 4.1 shows the hardware setup. Processing and web-serving were done on a Apple G5 Dual 2GHz computer with 2GB RAM and OS X as an operating system. A Terra-byte disk-array was used as storage for the data collected from the WWW. Software was written in C++ and Perl only. The following sections discuss the challenges met and decisions made to design a large-scale image retrieval system.

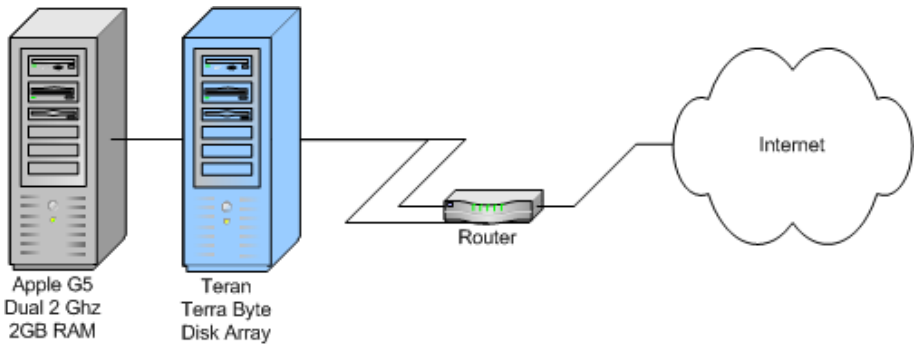

Figure 4.1: Hardware setup

4.1 Collecting the Data

The DMOZ open directory project was taken as a starting point for our data collection. To be specific, from the data DMOZ offers as a RDF [1] (Resource Description Framework) file we took URLs from the *Shopping* and *Recreation* categories. The data was entered into our relational database. From there, a web-crawler written in Perl took over. It takes the URLs and analyzes the web page for links and images. Links within the same directory of the URL are added to the queue of URLs to visit. Image tags are analyzed and the obtained information is entered into a table in the databases. Namely the image URL, its dimensions and the URLs of the web-site and the document the image was embedded in. Most importantly, the words from the image's ALT-tag and maximal 20 words above and 30 words below the image were stored, too. Also the category information from the DMOZ open directory was stored, however, in our work we never used it, except to organize storage. An additional Perl script downloaded all the images from the web and deposited them on the storage server. Note that one can easily reach file system limitations in that large a system: For instance most operating systems allow only some ten-thousand files per directory and our intend was to store millions rather than thousands. We decided to deposit the images in subdirectories corresponding to the DMOZ directories.

4.2 Extracting Features

Feature extraction was implemented in C++. Low level access to images was done with the C++ API to ImageMagick [2]. The feature vectors were stored in binary format: As BLOB (Binary Large Object) field in a relational database table and as a individual files on the Terra-byte disk array. This way a backup was generated on the fly. The software loads a *set* of images from the Terra-byte storage, calculates the features while keeping the results in memory and writes them to storage only as the last image of the set has been processed. In our final, highly optimized version of the code we were able to extract EHD, HTD, DCD and SCD of about $2 * 10^5$ images

[1] http://www.w3.org/RDF/

[2] ImageMagick is a collection of tools and libraries to read, write, and manipulate an image in many image formats. http://www.imagemagick.org

in 24 hours on the G5 2*2GHz platform. This means that the process of extracting features for our $3 * 10^6$ images takes 15 days with optimal code and full processor load!

4.3 Software Layers

Figure 4.3 shows the layered approach we took while designing our system. The design philosophy was motivated by

- finding an appropriate technology for each task
- at the same time keeping the number of programming languages small
- using standardized code and freely available applications

Software was written in Perl and C++. C++ was mostly used for the low level image feature related code since here speed is crucial and C++ offers the possibility to design appropriate data structures. Only ANSI C++ and standard libraries were used, which makes the code portable. As a compiler GNU gcc for OS X was used. Perl was mostly used for WWW related tasks such as collecting and processing the textual data, since it offers fantastic capabilities when it comes to handling text strings. Perl was also used as a middleware between the client (i.e. web-browser) and the image retrieval components written in C++. Fortunately, the CPAN [3] repository offers a lot of useful Perl libraries which are extremely helpful to rapidly deploy a system as ours. Perl allows fast prototyping of code which can then be transformed into web-applications without much effort. All the libraries used are released under some kind of Open-Source license. This means all of them are free for research purposes and costs for commercial use are very low.

4.4 Relational Database

As mentioned on several occasions a relational database is at the heart of our system. We chose such a system, because it offers simple methods to build and access structures like inverted indices or just to store tabular information like all the meta data that comes along with our images. The

[3] http://www.cpan.org

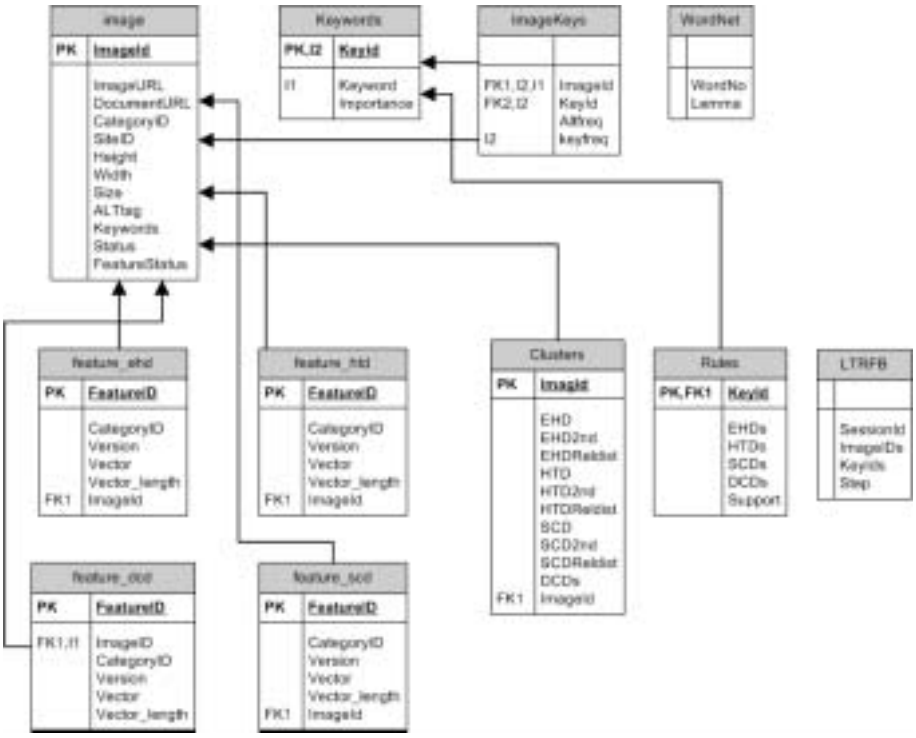

Figure 4.2: Simplified Entity Relationship Diagram (ERD) for our mySQL Database

product we operate with is mySQL. It is freely available [4]. mySQL showed good performance, for instance the inverted index consists of a table with about $60 * 10^6$ rows. However, understanding how mySQL works and using the right parameters (in particular setting the correct indices on the proper columns) turned out to be crucial. Figure 4.2 shows a structural diagram of the database. The table `image` is at the heart of the structure. It contains all the information for each image collected from the WWW, such as size, url, etc. The field `status` is set depending on if the image has been downloaded. The field `featurestatus` is set if the features were successfully extracted. Note that only information associated with each image ist stored in this table, i.e. the images themselves are stored on the an external storage server. The location of the file on this server is identified by the imageid and categoryid. The tables `feature_*` contain the feature vectors for each feature type. A keyword query is done on the inverted index `imagekeys` after looking up the `keyid` for the entered `keywword` in the table `keywords`.

[4] http://www.mysql.com

The table `keywords` is indexed on the column `keyword`, the table `imagekeys` on the column `keyid`. Listing the individual keywords (character data) in an extra table and indexing them there increases performance significantly since the number of rows in `keywords` is much smaller than the number of rows in `imagekeys`. Thus we need only an index on a numeric column in the large table `imagekeys`. The table `clusters` lists to which clusters each image belongs. The table `rules` contains the association rules sorted by keyword. The table `ltrfb` is used to store long-term relevance feedback, i.e. for each user that enters the system the keywords he enters and the images that are selected as relevant are stored. The table `wordnet` contains the words we extracted from the WordNet [22] database. These are only the tables that are significant for the implemented system, some of them contain also more columns than shown. In reality we used about twice as many tables for experiments with different types of association rules, restricted amounts of keywords etc. For the interested reader some parameters: We used mySQL version 3.23 for OS X. Table types are MyISAM. Access to mysql was done with the MySQL C API from C++ and DBD::mysql from Perl. We used version 3.3.2 of mySQL. The newer version 4.0 would enable us to achieve even better performance. E.g. in version 4.0 it is possible to load certain indices to memory manually and keep them in memory persistently. This would allow us to speed up the search process significantly.

4.5 Middleware

The connection between the low level application layer written in C++ and the Perl middleware was established using Inline::CPP. Inline::CPP allows the embedding of C++ code in Perl and in particular C++ functions can be called from Perl. The C++ code was compiled with gcc. Libtool was used to create a dynamic library from the object files. The classes and functions within this library can then be accessed through Inline::CPP which takes the library as a parameter. For this purpose a section in the Perl script is reserved where wrapper functions establish the connection between Perl and C++.

As a web-server Apache [5] was chosen. Apache calls Perl via the Common Gateway Interface (CGI). For an application with many users instead of

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

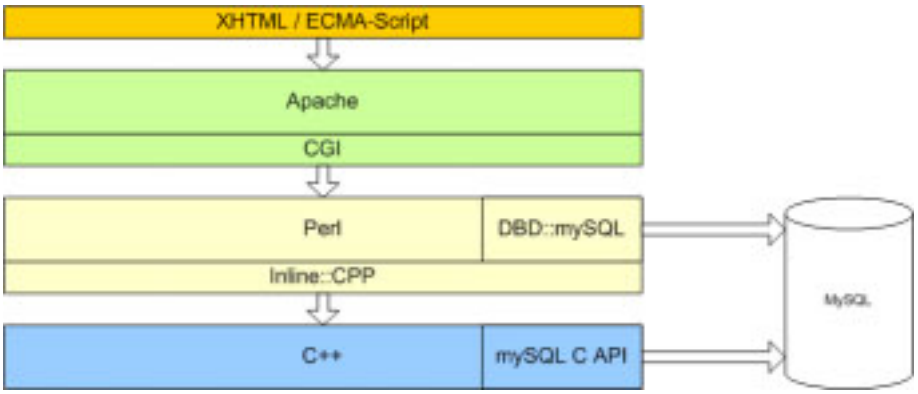

Figure 4.3: Software Layers

using CGI we recommend to use modPerl. modPerl is a persistent Perl interpreter embedded in the Apache web-server. This means that time-consuming calls to the external Perl interpreter through CGI are obsolete. For the relevance feedback variables for each user need to be stored across several steps of relevance feedback. Since HTTP is a connection-less protocol we needed to store the data in sessions. Session-functionality is provided by the Apache web-server and was accessed from Perl via CGI::Session.

Further noteworthy technologies include the Boost Graph Library and GraphViz. The Boost Graph Library [44] was used to implement the connected component analysis in chapter 3.4. It is claimed to have some of the fastest implementations of algorithms and graph access and is a candidate to be included in the C++ standard libraries. GrahpViz is a tool from AT&T which allows visualization of graphs. It can be called from the command line and this way be included into applications. Its SVG export capabilities allow display of graphs in web browsers. This could be the base for a more immersive, interactive, yet standards-compliant interface to multimedia retrieval applications on the web.

4.6 Presentation Layer

The interface with the user is an important element of every multimedia application. For applications on the web certain constraints like accessibility, portability and possibly low bandwidth have to be taken into account. We decided to implement an interface which displays and allows in every common web-browser without the need of additional plug-ins like Flash or

Figure 4.4: Cortina WWW interface

	EHD	HTD	SCD	DCD
QVMO	100	100	100	100
QVMW	0	0	0	0
Weight	9	2	66	20

Figure 4.5: Cortina WWW interface detail

Java applets - which may not be installed at all at the client or need extra bandwidth. Thus, the interface was designed in pure, standards-compliant XHTML and interactivity was enriched with the help of ECMA-Script (former Java-Script). The XHTML markup was separated from the underlying Perl code by using a template mechanism. In particular the `Template` library from CPAN was used. Figure 4.4 shows the web-interface to the application. The interface shows also information about the query and how it was handled by the system: for instance, in figure 4.5 an element of the interface is shown where the weighting of the descriptors is presented to the users.

Chapter 5

Discussion and Further Results

The first, prominent result of this work is the contribution to the implementation of *Cortina*, our large-scale, content-based image retrieval system for the World Wide Web. To our knowledge it is the largest, content-based image retrieval system available to date, with over 3 Million [1] images. The reader is encouraged to try [2] the system.

In the following, the system and and its performance shall be discussed in detail, in particular we focus on:

- The concept of a high-level keyword search followed by a visual nearest neighbor search with relevance feedback in distinction to either pure text-based search or pure query-by-visual-example.
- The improvements of this process based on the semantic analysis of the data, and the scalability of the methods suggested to exploit them.

5.1 Measuring the Quality of Search Engines

An imminent question in the information retrieval community is how to measure the quality of a search method or search engine. In general, when methods for information retrieval are proposed, they are tested on a small set of data, the exact composition of which is known to the researcher. This

[1] We are currently collecting more images and hope to increase the size of the collection to over 10 Million

[2] Try http://sobel.ece.ucsb.edu or http://www.cortina.ch

dataset is commonly called *the ground truth*. To measure the quality of a method the precision and recall measures are applied [45]. The desired results D for a query are known from the ground truth, A describes the actual query results and within A there is a set of correct results C. Precision P and recall R are

$$P = \frac{|C|}{|A|}$$

and

$$R = \frac{|C|}{|D|}$$

For large scale experiments as ours, there is no such ground truth and we can definitely not measure the recall R. However we can get some notion of the precision with some slightly modified concept for P, as will be shown below.

As a side note: There seems to be the need for some benchmark to compare retrieval systems. The TREC conferences [46] offer such benchmarks for text retrieval. Similar, large-scale multimedia datasets would be needed to compare the performance of retrieval systems. Several efforts are underway to create such datasets, e.g. at the Viper [47] project at the University of Geneva. In particular a large-scale benchmark dataset consisting of images and keywords would be needed to compare multi-modal retrieval techniques.

We decided to take the precision based upon results from user questioning as a measure for the performance of our system.

5.2 Overall Precision of the Image Retrieval System

The precision of the system was measured with user questioning. The participants had to complete the following steps

1. Enter a keyword query, count correct results.
2. From the results, select one image. Count how many very similar images there are and how many conceptually similar in the first 8 and first 16 positions. Do a similarity search with this image as input.
3. Count how many very similar images and how many conceptually similar there are in the first 8 and first 16 positions. From the results,

select all the very similar images and do another round of similarity search.

4. Repeat step 3. once.

Steps 2.–4. where repeated for three different images per keyword query. Five persons participated in the tests, the total number of queries was $11*3 = 33$. The keywords were

```
shoe, bike, mountain, porsche, bikini, flower,
donald duck, bird, sunglass, ferrari,
digital camera
```

The terms "visually very similar" and "conceptually similar" where explained to the user with an example: If the original keyword query was "shoe" and one had chosen a black shoe as visual query, every result that contains a black shoe is visually very similar. Other shoes are conceptually similar, images not related to "shoes" are wrong matches.

The precision was measured with

$$P = \frac{|C|}{|A| \in \{8, 16\}}, \tag{5.1}$$

where

$$C \in \{visually\ very\ similar, conceptually\ similar\}$$

i.e. it was measured how many correct matches were returned in the first 8 and first 16 results respectively. C ("correct results") was defined as "visually very similar" in the one case and as "conceptually close" in the other. It is evident that these measures contain some degree of subjectivity, but by having several persons and several visual queries per concept we hope to minimize these effects. A sample sheet as given to the persons questioned is shown in appendix C.

The graphs in figures 5.1 and 5.2 show the results of our precision experiments. The precision as in equation (5.1) is shown on the y-axis, along the x-axis are the several steps of the experimental queries, starting with a keyword-query and followed by three steps of relevance feedback on the visual appearance. Four series of results are shown per graph: Once for the search without the improvements discussed in chapter 3 (labeled "Normal"), and once with the improvements based on the semantics in the data (labeled

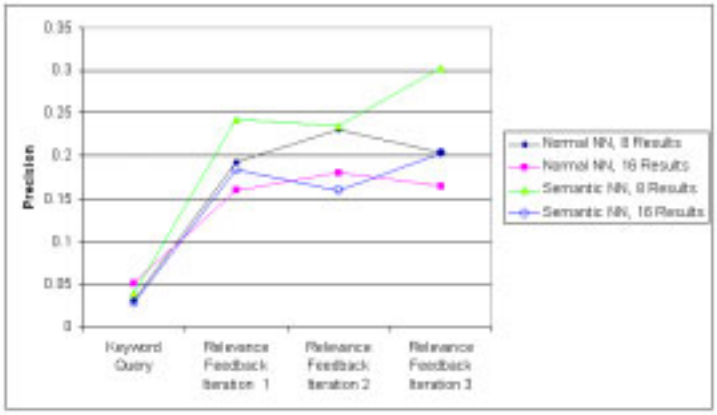

Figure 5.1: Precision curve for visually very close images.

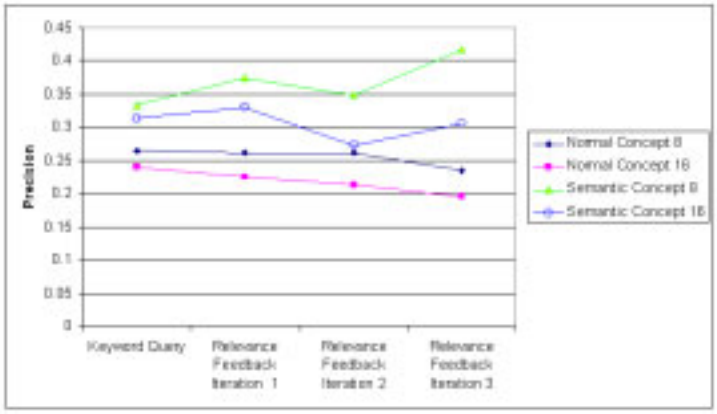

Figure 5.2: Precision curve for conceptually close images.

"Semantic"). For each experiment the precision based on the first 8 and first 16 query results is shown.

It can be seen that visual nearest neighbor search and relevance feedback improve the precision compared to pure keyword query results. In the case of close visual neighbors (figure 5.1), especially the improvements from the initial results to the first round of relevance feedback are significant. The improvements for the conceptual search are not as good as for the close visual neighbors, which was to be expected, since words describe higher level concepts much better than visual features. But when it comes to the precise look of an image the visual features take over. These results clearly show the benefits of an initial keyword query to identify the higher level semantic concepts, followed by a visual nearest neighbor search with relevance feedback to redefine the query based on visual appearance, i.e. the lower level semantic concepts.

Both graphs clearly show the positive effect of the improvements which were the focus of the thesis at hand: The semantically improved version of the search outperforms the original version. In particular it performs significantly better in the case of conceptual close images, where the precision is about 15% higher. This is due to the introduction of clusters identified by association rules into the visual nearest neighbor search: More images matching the higher level semantic concept, but perhaps not situated in the same visual feature cluster as the query, are "injected" into the results. The improvements are amplified by re-ranking the results based on their keyword similarity. This can also be deduced from the curves for the eight nearest neighbors: The re-ranking of the results lets some lower ranked results "jump" into higher positions. In general, it turned out that the semantically improved retrieval was equal or better to the normal version in nearly all cases and performed weaker very rarely. This has to be considered in relation to the scalability: As listed in Appendix A the size of the non-overlapping clusters for the semantic search is 7.3 GB - the size for the overlapping clusters is 53 GB, roughly a factor 7!

If one compares the exact values for the precision in both figures, it can be seen that the normal version of the relevance feedback saturates roughly between 0.2 and 0.25 in both figures, which means that the conceptually similar and visually very similar images are in fact more or less the same, i.e. there are no additional, conceptually similar images to the visually very

Figure 5.3: Query images selected for relevance feedback. Initial keyword query was "sunglass"

close images. But the semantic version reaches over 0.4 in figure 5.2 and about 0.3 in figure 5.1, which means it adds 10% conceptually similar images. The drop of the semantic version in the second step of relevance feedback is noteworthy: It is mostly caused by one sample in the experiments which gave comparably bad results for this step. By averaging over more queries the curve might become flatter at this place. However, it is very important to see that the curve goes up for the third step again. Since the semantic version introduces more conceptually close images to the results and not only visual nearest neighbors, the user has the possibility to select images which bring the system back "on track", which is not the case with pure visual nearest neighbor search - here, one is often stuck with the few correct results and one has no way to adapt the query. In fact, this can also be seen from the relevance feedback in the normal version: the curve drops for the third step of relevance feedback, no improvements are made from there. It would be interesting to see if an even larger dataset would improve this situation: often there seem to be just no more semantically and visually very similar images available. In a pure scientific dataset with a ground truth this could be measured with the recall values, but in our setup this is impossible.

The results are also illustrated with a visual example: In figure 5.4 results for the normal version of relevance feedback is shown. The images in figure 5.3 were selected as a query. The results for the same query but with the semantic version are shown in 5.5. The examples show that the semantic version sometimes achieves significant improvements over the normal version. In this example, the improvements result probably from the difficult query vector: The combination of the (texture) feature vectors of the sunglasses in figure 5.3 moves probably from both individual images, since the orientation

of the two query objects is different. The color feature vectors are sensitive to the white background and some gray/silver mixture in the query. The semantic version is not affected as strongly, since it loads clusters with many sunglasses — the normal version loads only the (overlapping) cluster with the centroid closest to the combined query vector. The improvements with the semantic version are not always that good — in most cases the results are comparable or only slightly better than the normal version, but with better scalability.

The results in the graphs 5.1 and 5.2 suggest that the semantic version of the search outperforms the normal version. However, it must be stated that the associations between keywords and image feature clusters were not very strong in general, such that sophisticated methods like auto-annotation or image understanding are overambitious with dataset as it presents itself at the moment, i.e. the uncontrolled character of the WWW data poses extraordinary challenges.

5.3 Frequent Itemset Mining and Semantic Rules

To explore the underlying semantics in the data, frequent itemset mining has been identified as a very scalable method to identify relations between keywords and image feature (clusters). The complexity of the method is far lower than the methods discussed in related work. Figure 5.6 shows the runtime of the frequent itemset mining (done per keyword) for our roughly $40 * 106$ transactions (over 12407 keywords) with different minimal support thresholds. The mining takes about half an hour for the transactions generated from our $3 * 10^6$ images. The difference between MFI and CFI is not very significant. Also, the curve for the runtime is rather flat in the depicted range — this could be explained by the rather short transactions. Since the relation between the number of keywords and the number of images is linear, the method should also scale linearly with larger datasets.

The values from tables 3.3 and 3.4 are the source for figures and 5.7 and 5.8. It can be seen that the number of "long" frequent itemsets is somewhat lower than the overall amount of frequent itemsets and that the support is low in general. (Remember, that an itemset has been defined as "long" if it associates a keyword with several low-level clusters of different feature types). The comparison of figures 5.7 and 5.8 shows again the trade-

Figure 5.4: Results after one step relevance feedback “normal”

Figure 5.5: Results after one step relevance feedback “semantic”

off between how many frequent itemsets one collects and the confidence of the deduced rules. A value between 0.1% and 1% for the minimal support might be the best choice. However, methods to increase the support would be needed. One method might be to identify first keywords that appear frequently together as a concept and then mine per concept instead of mining per single keyword. I.e. a multistage process which applies frequent itemset mining to keywords only, first, and based on these results is applied to keywords and visual features. This would make sense conceptually, since one would identify high level associations before mining for low-level associations. The method of keyword-clustering discussed in section 3.3.3 could be used to find these high level concepts, also. Questions that remain are the influence of the uneven distribution of the low-level feature clusters or the interestingness of rules based on pure statistical measures. While a lot of improvements can still be done, the results in figures 5.1 and 5.2 are quite encouraging already.

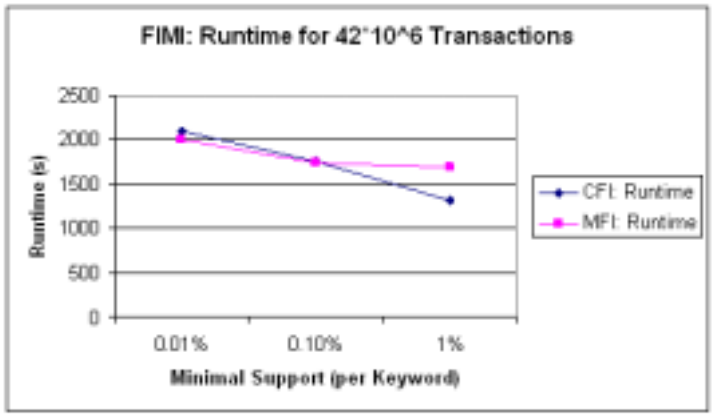

Figure 5.6: Runtime of frequent itemset mining algorithms.

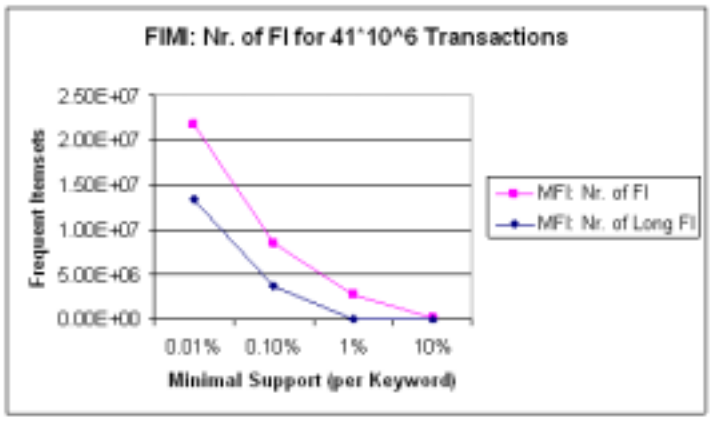

Figure 5.7: Number of frequent itemsets and long frequent itemsets.

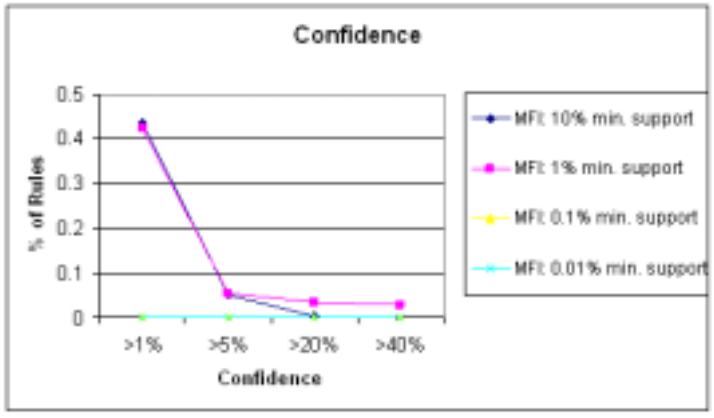

Figure 5.8: Confidence for different minimal support thresholds.

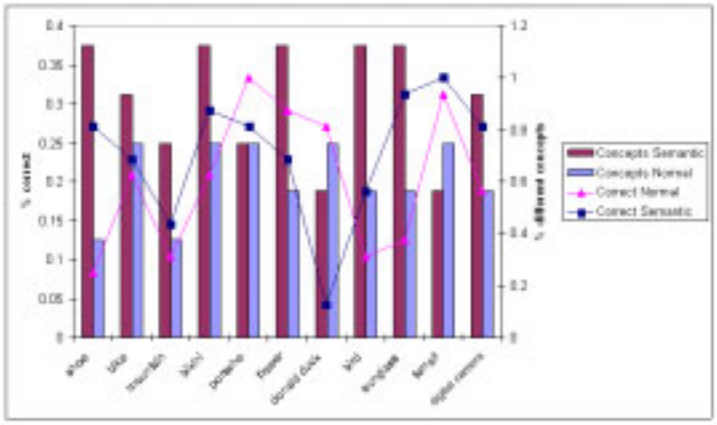

Figure 5.9: Semantic vs. Normal keyword search

5.4 The Keyword Search

Ranked boolean queries on an inverted keyword index were implemented as a base for text-queries. Text ranking of the results was done with the *td∗idf* measure. In distinction to existing image retrieval systems on the web, ranking was improved by including the visual appearance of the candidates into the ranking process. In figure 5.9 measurements regarding the result of the keyword search with ranking based on visual content, as introduced in section 3.4 are shown. It is visible that in average the precision increases somewhat, but in particular the number of different semantic concepts on the first set of results increases. This integrates perfectly into our concept of a keyword query followed by a visual nearest neighbor search: Offering a more diverse selection of results on the first page gives the user the chance to choose between more "directions" for the visual nearest neighbor search. At the same time, since connected components of images where ranked higher when they are large, which not only increases the semantic relevancy of the first results, but also the probability that there are some visually *and* semantically similar images to be found by the following nearest neighbor search. Visual examples for the results of this keyword query have been shown in chapter 3 in figures 3.13 through 3.16.

Chapter 6

Conclusions

In this thesis a concept and a system design for large-scale, content-based image retrieval on the WWW have been introduced and implemented.

Content-based image retrieval has been taken to a new level of scale by searching a collection of images in the order of Millions. To achieve this large-scale experiment, a large dataset consisting of over 3 Million images and collateral text was collected from the World Wide Web. Several low-level MPEG-7 image feature types were extracted. Keywords were indexed as an additional high-level feature. A complete system which enables access to the whole dataset was implemented and made available to the public on the WWW.

The suggested retrieval-concept starts with a query-by-keyword as a first step, which identifies high-level semantic concepts for this query. The diversity of these concepts and their relevance were increased by using a new, graph-based method which re-ranks and summarizes the images for a given keyword-query based on their visual content. At the same time, (semantic) noise introduced by images that are either visually far from the semantic concept or appear very often within the first few results was reduced. The method was made scalable to large datasets with a new way for approximating distance calculations.

From this summarized representation, in a second step, the user can select images which match the semantic concept he was looking for. The results of this visual nearest neighbor search are refined with relevance feedback.

Several proposals such as LSI or statistical models were discussed as a possibility to find relations between keywords and image features. Frequent

itemset mining was identified as one of the few (if not the only) existing *scalable* methods to gain insight into the semantics in the dataset, given by the associations between keywords and images. Transactions were defined in the context of web-image retrieval as a series of keywords and low-level feature cluster ids. The itemset mining process was done per keyword to scale the method and identify interesting rules. This way, the multi-modal characteristics of the data could be exploited in several ways. A method was introduced which selects low-level feature clusters for nearest neighbor search based on rules deduced from the semantic patterns identified. Initial weighting for the combination of several feature vectors was deduced from these patterns, too. The relevance feedback process was extended and improved by re-ranking the results of a visual nearest neighbor search with keyword information. The required keywords were obtained by keeping the initial query-keyword(s) in memory and by adding the keywords assigned to the query-images for visual relevance feedback.

The retrieval-concept of a keyword-query followed by relevance feedback on visual appearance was shown to result in more precise results than plain keyword search. The suggested improvements based on semantic rules deduced from frequent itemset mining and the graph-based summary of the keyword-query results were shown to increase the precision even more, while maintaining scalability. In particular, low-level feature clusters had not to be made overlapping, which reduced the required amount of data by a factor 7.

Several suggestions and experiments were made to even further exploit frequent itemsets and association rules in multimedia datasets. They include an approach to semantic clustering based an frequent itemsets or the incorporation of rules obtained from long-term relevance feedback. While our system for image retrieval on the WWW could not yet benefit from all these suggestions, we propose that the methods are also applicable to other, more controlled multimedia datasets, where they are believed to show good results.

Chapter 7

Outlook

While the system as introduced in the previous sections has shown to perform very well, we discovered many avenues which could lead to improved versions:

Starting with the low-level feature extraction, we suggest the implementation of some unsupervised segmentation which is applied where the data are suitable, in particular on images with uniform backgrounds, the number of which in our dataset we underestimated. In fact, in [6] it has been shown that segmentation increases the performance of image retrieval system. However, the question to what extent the additional computational efforts affect extraction, storage and retrieval of the objects obtained from segmentation needs to be considered.

There are alternative methods to combine the different visual MPEG-7 features, one very promising has bee suggested very recently in [48], where instead of a weighted distance, a decision making framework based on fuzzy theory is used.

The K-Means clustering algorithm was chosen for its fast processing, but did not deliver high-quality results. In particular the uneven occupation of the clusters lead to problems in scalability and in the quality of frequent itemset mining results. Recently, in [49] a new adaptive indexing structure was suggested. The suggested method is based on VA-Files (Vector Approximation). The problem with most existing VA-based methods is that they suppose an uniform distribution of the data, which is not the case usually. In fact, in [17] the feature-vector data from our *Cortina* dataset has been analyzed and the distribution of each dimension is Rayleigh rather than uniform for the HTD as an example. Based on this information, each of the

feature vector dimensions is partitioned into bins such that each bin contains approximately an equal number of objects. The method is very scalable and results in uniformly populated index bins, which matches our requirements perfectly. This method being an indexing rather than a clustering method, to mine association rules as suggested in this thesis, instead of using cluster identifiers in the transactions the hypercubes given by the index could be used.

Exploiting multi-modal characteristics of data-sets should be further investigated, not only for web-image retrieval. Frequent itemset mining and association rule discovery for this task can be taken much further. It seems to be one of the few methods which is applicable to large datasets. In particular the idea of building semantic clusters based on frequent itemsets which reflect the semantic associations between different modes of the data should be looked into for more controlled datasets.

Statistical measures for the interestingness of a rule should be considered. The so called *J-measure* as introduced in [50] might be a suitable tool.

The process of frequent itemset mining could be further adapted to the task. For instance, one could probably define special algorithms which find only relations between words and low-level feature clusters and disregard certain other associations. Once the size of the dataset grows even more, parallel implementations of frequent itemset mining algorithms can be used — this has been an active area of research in recent years.

More data-sources could be included into the search process. We started to collect long-term relevance feedback, i.e we stored usage data. Unfortunately we could not collect enough data yet, that could be included in the search process. But once enough information has been gathered, frequent itemset mining is suggested to analyze these data in the same manner as introduced for our images. The information gained from this could be combined with the existing data to improve semantic knowledge and, based on that, the search process. Improvements made for text-retrieval on the web, such as PageRank [2] could be implemented. The semantics within the collection could be further analyzed by integrating thesaurus-data like WordNet [22], in particular the semantic distance between keywords, as mentioned in the appropriate chapters in this work.

The basic system has been shown to be scalable to large amounts of data. To implement even larger systems, issues like parallelization, operating

system limits etc. come into play. One of the most urgent improvements is a real database server. Our database runs on a regular high-end PC together with many applications. For database tasks the disk access is the bottleneck, thus a database server with fast and parallel organized hard-drives is needed.

An issue of very different nature is copyright: The collection and use of images by the system user raises some questions. Some of them have been addressed in [51].

Appendix A

Cortina in Numbers

The system *Cortina* can be characterized with the following numbers as of 25 April 2004.

Number of Images	3'006'660
Number of Keywords	680'256
Size of Keyword Index	50'260'345 lines
Size of mySQL Database	23 GB
Size of Image Collection	111 GB
Size of non-overlapping feature vector clusters	7.3 GB
Size of overlapping feature vector clusters	51 GB
Size of Transactions File (1 keyid, ehd, scd, htd)	1.33 GB
Size of Transactions File (1 keyid, ehd, scd, htd, dcd)	2.63 GB

Appendix B

About the Name Cortina

Cortina is Latin and means caldron or tripod. The term is used in context for the ancient oracle of Delphi. In his famous work *Aeneid*, the poet Vergilius (70-19 b.C.) speaks of the "Phoebi Cortina", the caldron in the temple of Apollo at Delphi. Cortina as a synonym for the ancient oracle of Delphi inspired my choice for the name of our system. The well-known quote "Neque te Phoebi Cortina fefellit" from the Aeneid translates to "Apollos caldron did not deceive you".

Latin: Aeneid 6:340-351

Hunc ubi vix multa maestum cognovit in umbra, sic prior adloquitur: "quis te, Palinure, deorum eripuit nobis medioque sub aequore mersit? dic age. namque mihi, fallax haud ante repertus, hoc uno responso animum delusit Apollo,qui fore te ponto incolumem finisque canebat venturum Ausonios. en haec promissa fides est?" ille autem: "neque te Phoebi cortina fefellit, dux Anchisiade, nec me deus aequore mersit. Namque gubernaclum multa vi forte revulsum, cui datus haerebam custos cursusque regebam, praecipitans traxi mecum [...]

English Translation (ed. Theodore C. Williams)

Aeneas now Discerned his sad face through the blinding gloom, And hailed him thus : 0 Palinurus, tell What god was he who ravished thee away From me and mine, beneath the o'crwhelming wave? Speak on! for he who ne'er had spoke untrue, Apollo's self, did mock my listening mind, And chanted me a faithful oracle That thou shouldst ride the seas unharmed, and touch Ausonian shores. Is this the pledge divine? Then he, 0 chieftain of Anchises' race, Apollo's tripod told thee not untrue. No god did thrust me down beneath the wave, For that strong rudder unto which I clung, My charge and duty, and my ship's sole guide, Wrenched from its place, dropped with me as I fell.

Appendix C

User Questioning Form

A sample of the forms used for questioning user about the precision is shown in figure C.1.

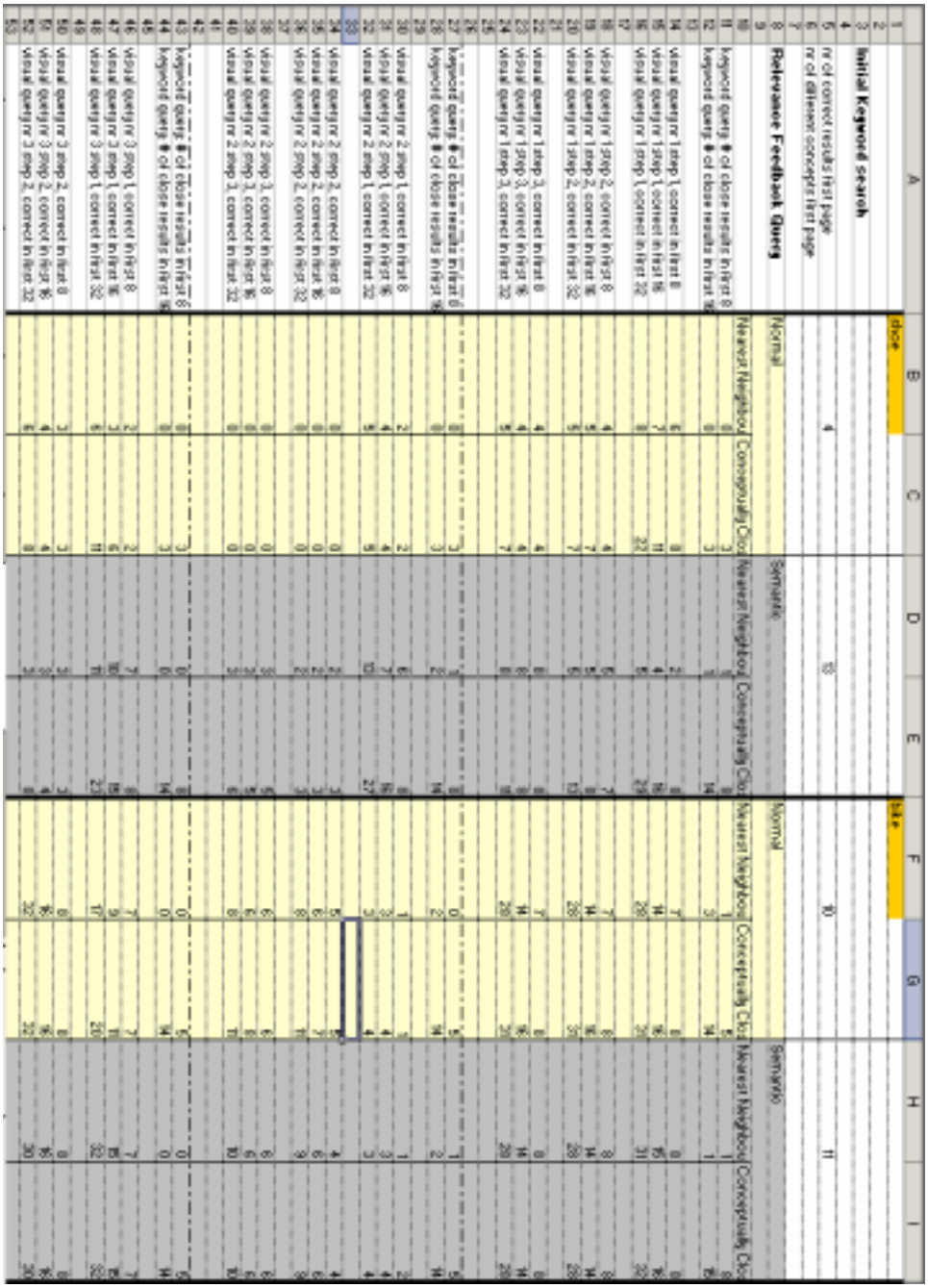

Figure C.1: Sample Questionnaire

Appendix D

CD ROM

Bibliography

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[2] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[3] Wei-Ying Ma and B. S. Manjunath. A texture thesaurus for browsing large aerial photographs. *Journal of the American Society of Information Science*, 49(7):633–648, 1998.

[4] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petković, David Steele, and Peter Yanker. Query by image and video content: The qbic system. *IEEE Computer*, 28(9):23–32, September 1995.

[5] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third International Conference on Visual Information Systems*. Springer, 1999.

[6] Wei-Ying Ma and B. S. Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[7] Ingemar J. Cox, Matthew L. Miller, Thomas P. Minka, Thomas Papathomas, and Peter N. Yianilos. The bayesian image retrieval system, pichunter: Theory, implementation and psychophysical experiments. *IEEE Transactions on Image Processing (to appear)*, 9(1):20–37, January 2000.

[8] Yong Rui and Thomas S. Huang. A novel relevance feedback technique in image retrieval. In *Proceedings of 7th ACM International Conference on Multimedia (MM)*, pages 67–70, 1999.

[9] M. Swain, C. Frankel, and V. Athitsos. Webseer: An image search engine for the world wide web, 1997.

[10] Shawn Newsan, Baris Sumengen, and B.S. Manjunath. Category-based image retrieval, 2001.

[11] Open Directory Project. http://www.dmoz.org.

[12] B.S. Manjunath, Philippe Salembier, and Thomas Sikora, editors. *Introduction to MPEG-7*. John Wiley and Sons Ltd., 2002.

[13] Yining Deng, B. S. Manjunath, Charles Kenney, M. S.Moore, and H.Shin. An efficient color representation for image retrieval. *IEEE Transactions on Image Processing*, 10(1):140–147, January 2001.

[14] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[15] Getty Images Inc. getty images, April 2004.

[16] Corbis. Corbis, April 2004.

[17] Jelena Tešić. *Managing Large-scale Multimedia Repositories*. PhD thesis, University of California at Santa Barbara, 2004.

[18] Ullrich Moenich. Studienarbeit, April 2004.

[19] Gerald Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[20] 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.

[21] Brian Pinkerton. Finding what people want: Experiences with the webcrawler. In *The Second International WWW Conference*, October 1994.

[22] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. Five papers on wordnet. *International Journal of Lexicography*, 1991.

[23] Lars Thiele. Studienarbeit, April 2004.

[24] Ka-Man Wong and Lai-Man Po. Mpeg-7 dominant color descriptor based relevance feedback using merged palete histogram. May 2004.

[25] T. Westerveld. Image retrieval: Content versus context. In *Content-Based Multimedia Information Access, RIAO*, 2000.

[26] Kobus Barnard and David Forsyth. Learning the semantics of words and pictures. *International Conference on Computer Vision*, 2:408–415, 2001.

[27] Kobus Barnard, Pinar Duygulu, and David Forsyth. Clustering art. *Computer Vision and Pattern Recognition*, II:435–439, 2001.

[28] K. Barnard, P. Duygulu, D. Forsyth, N. de Freitas, D. Blei, and M. Jordan. Matching words and pictures, 2003.

[29] Kobus Barnard, Pinar Duygulu, and David Forsyth. Exploiting text and image feature co-occurrence statistics in large datasets. to appear as a chapter in Trends and Advances in Content-Based Image and Video Retrieval (tentative title), 2004.

[30] Thomas Hofmann and Jan Puzicha. Statistical models for co-occurence data, 1998.

[31] Jia Li and James Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1075–1088, 2003.

[32] Jelena Tešić, Shawn Newsam, and Bangalore S. Manjunath. Mining image datasets using perceptual association rules. In *Proceedings of SIAM Sixth Workshop on Mining Scientific and Engineering Datasets in conjunction with the Third SIAM International Conference (SDM)*, May 2003.

[33] Osmar R. Zaïane, Jiawei Han, Ze-Nian Li, and Jean Hou. Mining multimedia data. In *Proceedings of CASCON'98: Meeting of Minds*, pages 83–96, November 1998.

[34] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993.

[35] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: generalizing association rules to correlations. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, pages 265–276, 1997.

[36] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994.

[37] David J. Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*. MIT Press, 2001.

[38] Bart Goethals and Mohammed J. Zaki. Advances in frequent itemset mining implementations. In {*FIMI'03*} *IEEE IDCM Workshop on Frequent Itemset Mining Implementations*, November 2003.

[39] Gosta Grahne and Jianfei Zhu. Efficiently using prefix-trees in mining frequent itemsets. In {*FIMI'03*} *IEEE IDCM Workshop on Frequent Itemset Mining Implementations*, November 2003.

[40] J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In *Proc. of 1995 Int'l Conf. on Very Large Data Bases (VLDB'95), Zürich, Switzerland, September 1995*, pages 420–431, 1995.

[41] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity - measuring the relatedness of concepts. In *Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, July 2004.

[42] Florian Beil, Martin Ester, and Xiaowei Xu. Frequent term-based text clustering. In *Proceedings 8th International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, 2002.

[43] Martin Ester Fung Benjamin, Wang Ke. Large hierarchical document clustering using frequent itemsets. In *Proc. SIAM International Conference on Data Mining 2003 (SDM 2003)*, May 2003.

[44] Jeremy G Siek, Lie-Quan Lee, and Andrew Lumsdaine. *The Boost Graph Library, User Guide and Reference Manual*. Addison-Wesley, 2002.

[45] Vittorio Castelli and Lawrence D. Bergman, editors. *Image Databases - Search and Retrieval of Digital Imagery*. John Wiley and Sons Ltd., 2002.

[46] Donna Harman. Overview of the first text retrieval conference (trec-1), 1992.

[47] Viper project benchmarking website. http://viper.unige.ch/research/ benchmarking/index.html, visited April 21 2004.

[48] Azadeh Kushki, Panagiotis Androutsos, Konstantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. Retrieval of images from artistic repositories using a decision fusion framework. *IEEE Transactions on Image Processing*, 13:277– 292, March 2004.

[49] Jelena Tešić, Sitaram Bhagavathy, and B. S. Manjunath. Issues concerning dimensionality and similarity search. In *Proceedings of 3rd International Symposium on Image and Signal Processing and Analysis (ISPA)*, September 2003.

[50] Padhraic Smith and Rodney M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Date Engineering*, 4(4):301–316, August 1992.

[51] Beat Hangartner, Lukas Hohl, Tobias Koch, and Till Quack. Neue medien - stand der technik - stand des rechts. Seminararbeit in Urheberrecht Uni/ETH Zürich, June 2002.