Fast Retrieval of High-Dimensional Feature Vectors in P2P Networks Using Compact Peer Data Summaries

Wolfgang Müller
Department for Applied Computer Science 1
University of Bayreuth
Germany

wolfgang.mueller2@uni-bayreuth.de

Andreas Henrich
Department for Applied Computer Science 1
University of Bayreuth
Germany

andreas.henrich@uni-bayreuth.de

ABSTRACT

The retrieval facilities of most Peer-to-Peer (P2P) systems are limited to queries based on a unique identifier or a small set of keywords. The techniques used for this purpose are hardly applicable for content-based image retrieval (CBIR) in a P2P network. Furthermore, we will argue that the curse of dimensionality and the high communication overhead prevent the adaptation of multidimensional search trees or fast sequential scan techniques for P2P CBIR. In the present paper we will propose two compact data representations which can be distributed in a P2P network and used as the basis for a source selection. This allows to communicate only with a small fraction of all peers during query processing without deteriorating the result quality significantly. We will also present experimental results confirming our approach.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Data sharing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Search processes*

General Terms

Algorithms, Performance

1. INTRODUCTION

Peer-To-Peer networks (*i.e.* administration-free ad-hoc overlay networks in which each node has the same functionality) have gained popularity as a simple, low-overhead means for the dissemination and sharing of information (cf. section 2.2). In such networks users can search for items using a number of keywords, or simply by searching for a file name pattern, and then download them from the Peer-to-Peer network. In the present paper we will address the scenario where images are shared in a Peer-to-Peer (P2P) network. In this scenario each peer maintains its local image collection contributing to a large overall image collection.

The challenge in this scenario is to realize an efficient content-based image retrieval (CBIR) functionality. As usual, we assume that the CBIR functionality is based on feature vectors representing color, texture or shape properties of the images and that a CBIR query is processed as a similarity query on these feature vectors (cf. section 2.1). Two brute force approaches are conceivable for this purpose:

A first approach would be to broadcast the query in the whole P2P network. Then each peer has to perform the similarity query on his local data and to return it's best matches to the query issuer. Such flooding queries cause an unbearably high communication effort.

If we assume that queries are much more frequent than updates, an extreme alternative would be to replicate all index information (i.e. the feature vectors) for all images in the P2P network on each peer. In this case, queries can be processed locally and only updates have to be distributed in the P2P network. However, this would obviously lead to unacceptable storage requirements on each peer.

As a consequence, an intermediate approach balancing the storage overhead and the network load is desirable.

A promising approach in this respect is based on a distributed cluster algorithm: We determine a certain number of clusters for the whole image collection distributed over the P2P network. The centroids of these clusters are known to each peer. Then each peer distributes the information about the number of documents in each cluster to all other peers. Whenever a peer wants to issue a query, it will determine the most promising peers in advance based on the centroid information, and on the information about the frequency of documents in each cluster on each peer. Thereafter it will communicate its query to all peers maintaining images belonging to the respective clusters, starting with the peers containing the largest number of documents within the cluster. Fortunately, this approach is rather insensitive with respect to the number of clusters and to the accuracy of the cluster centroids. In other words, there is no need to update the cluster centroids with each new image. Instead, the need for a new calculation of the clusters arises only after a significant number of updates.

In the present paper we will present this approach together with an alternative and discuss their advantages and disadvantages based on experimental results. The paper is a contribution to the rather new field of CBP2PIR (content-

based peer-to-peer image retrieval) and a contribution to the more general field of distributed CBIR.

The remainder of the paper is organized as follows: Section 2 describes related work. Section 3 describes our work, speeding up CBP2PIR using compact representations of peer data. Section 4 describes the experiments we did for evaluating our approach, giving also some perspective for future work. Finally, we conclude in section 5.

2. RELATED WORK

Our approach for CBP2PIR is obviously related to work accomplished in two areas, namely content based image retrieval and retrieval in P2P networks.

2.1 Content-Based Image Retrieval

Content based image retrieval (CBIR) is about indexing images by their visual content. For achieving this task one usually extracts a multidimensional feature vector for each image, and maintains a multi-dimensional index structure on these feature vectors (cf. section 2.1.1). The index enables similarity search on the vectors: for a given vector $\mathbf{x}$ the k next neighbors can be found. The retrieval performance of CBIR systems is determined by image processing (see [24] for an overview), as well as computer learning aspects (like relevance feedback and browsing). Its time efficiency is largely determined by the dimensionality of the feature vectors, by the distance measure chosen [28], and the index structure used for indexing the feature vectors.

While issues of useful feature sets are clearly out of scope for this publication (we refer to [24]), in this paper we are concerned with indexing in a distributed environment.

2.1.1 Indexing of high-dimensional data

Indexing becomes hard in the context of CBIR, because — in contrast to text retrieval, where queries with one or two words are common — both documents *and* queries are represented by high-dimensional feature vectors. Typical image retrieval feature sets represent documents using feature vectors of $\mathcal{O}(100)$, if not $\mathcal{O}(1000)$, dimensions. Indexing of such vectors is very hard, due to the *curse of dimensionality*. The curse of dimensionality refers to the fact that even the closest data point within a collection is expected to be far away from a given query point, and one can rule out only little data on the way to finding it when increasing the dimensionality [1]. This fact makes a search with sublinear complexity a goal that is hard to attain.

Typical spatial index structures (*e.g.* [3, 5]) are conceived for low-dimensional spaces, and yield logarithmic complexity in the number of data items that have to be visited for dimensions up to ~ 10. Weber *et al.* [29] report experiments comparing a sequential scan to X-trees and R*-trees [3, 5] showing that for feature vectors of 10 or more dimensions, these spatial index structures yield very little advantage over a full sequential scan of the indexing data. Weber *et al.* suggest the VA-file, a structure performing such a scan in an optimized fashion. Using intelligent prefiltering of data a performance is achieved that is faster than both a simple sequential scan and spatial index structures. However, for the prefiltering, a short representation for each data point in the collection has to be analyzed, leading to linear complexity with a low constant. Hafner *et al.* [15] also suggest an approach that uses prefiltering, yet in a much more restricted setting.

Applying the lessons learned for high dimensional index structures to our scenario, it becomes obvious that tree-structured approaches are not well suited for a distributed setting in the high-dimensional case due to the curse of dimensionality. Also, approaches that involve a full scan of indexing data — such as the VA-file — are not desirable, because this would involve contacting all peers within the network.

2.1.2 Probabilistic Retrieval

The probabilistic view of retrieval and similarity has been shown to be very flexible and successful (*e.g.* in [28]). The peer content representation that we will describe below (cf. section 3.3) will make use of this probabilistic view of content-based image retrieval, as derived in [28].

Vasconcelos [28] describes image retrieval systems as maps from images to image classes. Let $\mathbf{x} \in \mathcal{X}$ denote an image feature vector and let $y \in \mathcal{Y}$, with $\mathcal{Y} = \{1, \ldots, M\}$ be the label of an image class. $\mathbf{X}, Y$ are the corresponding random variables. An image retrieval system is thus a mapping g from images to image classes:

$$g: \begin{array}{ccc} \mathcal{X} & \rightarrow & \{1, \ldots, M\} \\ \mathbf{x} & \mapsto & y \end{array} \tag{1}$$

The model assumes that for a query $\mathbf{x}$ the retrieval system should return all images falling into the class y with $g(\mathbf{x}) = y$. An image retrieval system now tries to minimize the retrieval error, *i.e.* the probability of retrieving images from a class different than that to which the query $\mathbf{x}$ belongs.

It is well-known that the retrieval error is minimized using the Bayes classifier

$$g^*(\mathbf{x}) = \arg\max_i P(Y = i|\mathbf{X}=\mathbf{x}) \tag{2}$$

$$= \arg\max_i P(\mathbf{X}=\mathbf{x}|Y = i)P(Y = i) \tag{3}$$

Here $\arg\max_i P(\ldots)$ stands for the i maximizing the probability $P(\ldots)$. $P(Y = i|\mathbf{X} = \mathbf{x})$ stands for the probability that i is the correct class for a given query $\mathbf{x}$. $P(Y = i)$ stands for the (prior) probability that any vector falls into class i and $P(\mathbf{X}=\mathbf{x}|Y=i)$ denotes the probability that the query $\mathbf{x}$ is an element of the given class i.

We will use this Bayesian view to select peers that probably contain searched result items. The details will be described in section 3.3.

2.2 Looking up Data in P2P Systems

Balakrishnan *et al.* define the lookup problem in P2P systems as follows [2]: "How do you find any given data item in a large P2P system in a scalable manner, without any centralized servers or hierarchy?" Unfortunately most approaches in this field address the search based on a unique key or a small set of keywords. We will summarize these approaches and first approaches towards content based retrieval in P2P systems in the following.

It is important to keep in mind the life cycle of a Peer in a P2P network:

Initial introduction: The initial introduction is the process of connecting to the P2P network *for the first time*. For connecting to the P2P network, one needs to know at least one peer to connect to. Classic ways of initial introduction are address lists that are either diffused in newsgroups, mailing lists, or on web pages.

Joining the network: On connecting to an already active peer (either initially or after an absence from the network), the new peer has to find its place within the network. On some networks (such as the initial Gnutella [8]) this phase is trivial, on others some work has to be done up front to maintain some structure invariant within the network. A compromise is to optimize the network during operation (see *e.g.* [17]).

Operation: During operation, the peer can issue and forward requests to other peers, and will process some requests locally. It typically performs some administrative work for optimizing the network or keeping up the structure invariant of the network.

Leaving the network: Typically it is not safe in P2P networks to rely on nodes to quit the network gracefully.

One of the challenges in creating P2P networks is to provide the right tradeoff between administrative overhead and query performance for a given purpose.

2.2.1 Looking up documents by their identifiers

Early P2P systems mainly performed search operations based on *identifiers* rather than content based information retrieval or even content based media retrieval. To this end, systems such as Gnutella performed so-called *flooding queries*. This means, a given peer issuing a query distributes the query to all its neighbours within the overlay (P2P) network [8]. Each neighbour processes the query and forwards it to all its neighbours (one such sending of a query is called a *hop*). When receiving the results from its neighbours it returns query results to the issuer of the query. Infinite loops of queries within the graph are avoided by giving each query a unique ID (peers never answer twice to queries with the same ID) and by defining a maximum number of *hops*, the *time to live (TTL)*. This method has numerous disadvantages: it wastes resources [22], and it cannot guarantee that all matches to a query present in the P2P network can be found within a query. The advantages of Gnutella are its simplicity, and the practically inexistent administration overhead. Some of Gnutellas problems are being solved using different classes of peers (super peers) which are out of scope of this paper.

FreeNet [7] (whose goal is efficient, anonymity-preserving publication) avoids wasting of resources by aggressive caching, and by a gradient ascent search algorithm with backtracking. Information about the gradient is provided using a routing table present in each peer. While this is already a considerable advance, FreeNet is not able to find out in a guaranteed number of hops, if a given item is present in the P2P network or not. Distributed hash tables (DHTs) such as CAN, Chord, and Pastry [21,23,26] provide guaranteed searches for items in the P2P networks, and are used as building blocks throughout the literature (a controversial example is [16]).

2.2.2 Content-addressable networks

CANs are concerned with similarity search on vectors. CANs were originally intended to provide simple DHT functionality, looking up objects by their multi-dimensional identifiers. In the original work, the identifiers were not required to carry any semantics. For space reasons we have to omit a full discussion which can be found in [21].

While CANs efficiently verify the *exact* presence or absence of a given data vector within a collection, due to the curse of dimensionality they fail if we want to find the nearest neighbours to a given data item in a vector space with high dimensionality.

Another weak point from our point of view is that CANs require sending a substantial amount of feature vectors (possibly all of them) to other peers when entering the network ([27], described in section 2.2.4, takes exactly this approach). In this publication, we seek for solutions in which the fact that a node enters the network causes only little administrative network traffic.

2.2.3 Information Retrieval

While most DHTs deal with one-dimensional IDs (CANs, as an exception use low-dimensional vector spaces for routing), there is now more and more research concerning information retrieval queries in P2P networks.

Joseph [17] proposes NeuroGrid, which (like some other current P2P approaches, *e.g.* [13]) permits keyword-based search for annotated items. NeuroGrid exploits the fact that each item is described by just a few keywords. As in FreeNet, routing tables are maintained in each peer. After a successful query the query issuing peer is connected directly to the provider of the match. Furthermore, the routing tables are updated, creating a link between the *keyword* and the provider of the match.

FASD [18] uses the same query routing scheme as FreeNet, however, to make the routing decisions it uses *term vectors* (*i.e.* the vector of words present/absent in a document) of full documents and the vector space model instead of document IDs. This approach is particularly effective in situations where the queries consist of only a few keywords. On multi-keyword queries, however, this scheme is likely to suffer from the curse of dimensionality.

Cuenca-Acuna and Nguyen [9] provide full-text information retrieval in their system PlanetP, which is inspired by GlOSS, and CORI [6, 14]. In PlanetP, the content of each peer is summarized using a *bloom filter*. A bloom filter is a bit array that represents a set A with fewer bits than $|A|$. In this case, the set represented by the bloom filter is the *set of terms* (words) present in the peer. Each peer distributes its bloom filter using a rumor spreading algorithm, and receives the bloom filters from other peers using the same mechanism. Query processing is done by choosing peers likely to contain matches, and then visiting peers starting with the most probable provider of a match.

PlanetP motivated our research, in that it is a viable solution for CBP2PIR, if a suitable peer data representation similar to the bloom filters used for the text scenario in PlanetP can be found for peers containing multimedia data.

2.2.4 Multimedia Information Retrieval in Peers

Tang *et al.* [27] present two approaches for text retrieval in P2P systems and suggest that these approaches can be extended to image retrieval as well. One of them (pVSM, peer vector space model) stores *(term, document id)* pairs in a DHT, and then retrieves such pairs to perform text information retrieval. According to the authors this approach can be handled due to the fact that terms are Zipf distributed, and one can thus limit the index for each document to a few strongly weighted terms. This approach seems to be hard to translate to image data, since [25] reports the need for a large number of features when using text information retrieval methods on image data.

The other approach suggested in [27], pLSI, is based on LSI [10]: Singular Value Decomposition is used to decrease the dimensionality of the feature vectors, and the reduced feature vectors are stored in a hierarchical version of the content-addressable network (eCAN). For each *document* of the collection, *(feature vector, document id)* pairs are stored in the eCAN. The curse of dimensionality is addressed by partitioning each feature vector into several, lower dimensional feature vectors, $((x_1, \ldots, x_n)$ will become $(x_1, \ldots, x_{n_1})$, $(x_{n_1+1}, \ldots, x_{n_2})$, $\ldots, (x_{n_{m-1}+1}, \ldots, x_{n_m}))$. Each of these vectors is said to be in a *plane*. The first of these lower-dimensional partial vectors is stored in a CAN_1, the second in CAN_2 and so on through CAN_m. A given query is then split into several queries of lower dimensionality. Together with aggressive pruning, Tang *et al.* manage to achieve impressive results on *textual* data: few nodes need to be accessed for achieving a high precision. The future will have to show if these results can be achieved also on image data.

While these results are impressive, they come at a price: To introduce a collection of N_x feature vectors of dimensionality n to a network of N_P peers, $N_x \cdot \sqrt[n]{N_P}$ messages have to be sent to the network. In the case of m planes, this becomes $m \cdot N_x \cdot N_P^{\frac{m}{n}}$. As a consequence, new peers with a collection of 10000 images will have to ship multiple megabyte to other peers when entering the network.

Ng and Sia [19] present the Firework Query Model for information retrieval and CBIR in P2P networks. In the Firework Query Model, there are two classes of links, normal *random* links, and privileged *attractive* links. A query starts off as a Gnutella-like flooding query. If a peer deems the query too far away from the peer's local cluster centroid, it will forward the query via a random link, decreasing the TTL of the query. Otherwise, it will process the query, and forward it via all its attractive links without decreasing the TTL.

One weakness of the DisCoVIR system presented in [19] is the limitation to one data cluster per node to check if a query is interesting for a server. This appears to have been solved recently [20].While DisCoVIR is similar to our work in some aspects, there are important differences: For one, Ng and Sia do not optimize the total number of messages sent, they rather optimize the number of times a peer has to actually process the query. Secondly, in their evaluation they use a scenario with a large amount of replication within the network and with very many peers. In our intuition, this makes some parts of the work simpler, as there is a larger chance of coming across a replication of an item by sheer luck. In any case, the work about peer data representation that we are presenting here would also be beneficial in a setting like the Firework Query Model.

3. CBP2PIR USING COMPACT PEER DATA REPRESENTATIONS

3.1 The Basic Scenario

In our scenario, we assume a P2P network in which users share their image data when they want to, that is, they cannot (and do not want to) guarantee high availability of their data. A user who wants to share her image data will first extract image feature vectors and index these vectors locally. She will then connect to the P2P network.

When connecting to the network, there are two possibilities: *Either* she is using a peer data representation that can be derived entirely locally. In this case her peer will send out the representation of the local peer data immediately after connecting to the P2P network. The network will diffuse this information using rumor spreading algorithms. *Or* the peer data representation needs some information about the P2P network's state. In this case, the peer will first request that information and then use it to derive its local representation before sending it out.

In any case, when connecting to the network the new peer will receive information about the peers present in the network, and will be able to query the network. Queries in our scenario are queries by pictorial example: the query is a feature vector, next neighbors to the feature vector are searched within the network. For processing a query, the peer will scan the peer data representations known to it, and will then contact directly the peers that are most likely to contain matches to the query, according to the peer data representations. The query processing efficiency is determined by the number of peers that have to be contacted in order to obtain the result, because the amount of data to be shipped (the query results) is small.

In this paper we focus on *compact* peer data representations, so that they can be easily transferred from peer to peer, representations that reduce the number of peers that have to be contacted for performing a query.

3.2 Distribution of Data over the Peers

When seeking to represent peer content, we have to make simplifying assumptions about the data present in each peer. This might become clear if we assume the data of each peer to be an independent identically distributed sample of the data within the full collection, and if we assume each peer collection to be large. In this case, we would expect the feature vectors within each peer to be distributed like the feature vectors within all other peers. It would be impossible to find a peer that is most likely to contain the best match without knowing the full indexing data present in the peer.

Fortunately there is some evidence that the distribution of the data over the peers will be non-uniform in many interesting application scenarios. In our scenario, we assume that people mainly share documents they have generated by themselves, *e.g.* scientific documents containing text and graphics, or photos from their last holidays. Furthermore, we assume that the data provided in one peer is provided by one person, or a small group of persons.

In such a scenario, we can assume that each peer contains data that can be partitioned into a small number ($N_{PeerClusters}$) of clusters.
For example, research documents provided by a research group usually can be partitioned into sets of documents, each belonging to one of a small number of research interests. In particular, we assume the number of nodes in the P2P network to be large compared to the number of clusters within a node: $N_{Peers} \gg N_{PeerClusters}$.

As mentioned before, optimizing P2P queries means limiting the number of network connections and the amount of data sent over the network for processing a query. To summarize the above assumptions, they mean that data in peers is sufficiently diverse among the peers, avoiding the worst case scenario, in which each and every peer has to be contacted to process a single query.

3.3 Compact Descriptions of Peer Content

As described in section 2.1.2, probabilistic image retrieval views the retrieval problem as a classification problem (equation 2). In our case, we are looking to find the peer (or peers) which are most likely to contain the feature vector $\mathbf{x}$ of the query document we are looking for. Let A be the random variable representing the peer containing a given feature vector. Then, equation 3 becomes:

$$g^*(\mathbf{x}) \quad = \quad \arg\max_a P(\mathbf{X}=\mathbf{x}|A=a)P(A=a) \qquad (4)$$

In this equation $P(\mathbf{X}=\mathbf{x}|A=a)$ is the probability that $\mathbf{x}$ is to be found in a given peer a, and $P(A=a)$ is the *prior probability*, which is proportional to the number of documents contained in peer a in our case.

Now, the goal of the peer content representation below must be to enable the estimation of the probability to find a given data vector $\mathbf{x}$ within a given peer $A=a$. So, we need a function f and a representation r_a of the data in peer a, such that: $f(\mathbf{x}, r_a) \approx P(\mathbf{X}=\mathbf{x}|A=a)$.

3.4 Statistical Independence and Maximum Likelihood

In this first approach we assume statistical independence of the dimensions of the feature space. For a query vector $\mathbf{x}=(x_1,\ldots,x_n)$ this leads to:

$$P(\mathbf{X}=\mathbf{x}|A=a) = \prod_{j=1}^{n} P(X_j=x_j|A=a) \qquad (5)$$

One way to represent $P(X_j=x_j|A=a)$, is to fit the probability density to a model, *e.g.* to the well-known Gaussian Mixture Model, used for example in [28]. For reasons of simplicity we limit ourselves to the approximation of $P(X_j=x_j|A=a)$ by a piecewise constant function, *i.e.* we generate a histogram of x_j for each peer. For each vector component x_j, we take a fixed number n_{bins} of bins with a variable width, such that $\min(x_i)$, $\max(x_i)$ are borders of bins. It is reasonable to encode each bin using a 16-bit word.

In this representation, the number of bits that describe a peer is given by $N_{Bits} = n_{Bins} \cdot n_{Dimensions} \cdot 16 + size(peerid)$.

3.5 Clustering

While in the previous model the compact representation of each node can be obtained independent from other nodes, in the following we will describe a method in which the nodes need to communicate in order to obtain the compact representation. We will analyze the communication cost at the end of this section.

We propose to divide the feature space into $N_{GlobalClusters}$ partitions, using a distributed clustering algorithm [11]. We will call these clusters "global" in order to distinguish them from the "local" clusters within each peer. We then use the frequency of documents belonging to a cluster as follows:

Let $\mathcal{C}$ with $|\mathcal{C}| = N_{GlobalClusters}$ be a set of global clusters. Let $c \in \mathcal{C}$ be a concrete cluster out of $\mathcal{C}$ and let C be the random variable denoting the cluster containing a given feature vector. Then we can write (given independence of X from A given C):

$$P(\mathbf{X}=\mathbf{x}|A=a) \quad = \quad \sum_{c\in\mathcal{C}} P(\mathbf{X}=\mathbf{x}|C=c)P(C=c|A=a) \qquad (6)$$

Here $P(\mathbf{X}=\mathbf{x}|C=c)$ stands for the *probability of membership* of x to cluster c and $P(C=c|A=a)$ stands for the *probability of documents in cluster c on peer a.*

The $P(C=c|A=a)$ can be approximated by the share of documents on peer a falling into cluster c. For the $P(\mathbf{X}=\mathbf{x}|C=c)$ various different approximations are conceivable. At present we simply find the cluster whose centroid is closest to $\mathbf{x}$ and set $P(\mathbf{X}=\mathbf{x}|C=c)$ to 1 for this cluster and to 0 for all other clusters, *i.e.* we assume that an $\mathbf{x}$ is strictly absent in all clusters except for the cluster $c_{\mathbf{x}} \in \mathcal{C}$ whose centroid is closest to $\mathbf{x}$. Applying this to equation 6 we get:

$$P(\mathbf{X}=\mathbf{x}|A=a) \quad = \quad P(\mathbf{X}=\mathbf{x}|C=c_{\mathbf{x}})P(C=c_{\mathbf{x}}|A=a) \qquad (7)$$

$$\quad = \quad 1 \cdot P(C=c_{\mathbf{x}}|A=a) \qquad (8)$$

As a consequence, peers with high values for $P(C=c_{\mathbf{x}}|A=a)\cdot P(A=a)$ will be contacted first, when processing a query $\mathbf{x}$. Technically, this can be realized efficiently by distributing the absolute numbers of documents per cluster for each peer. When a peer enters the network, it requests the cluster centroids and uses them to calculate for each document of its local collection to which global cluster it belongs. Each peer can then calculate a histogram over the global cluster labels $c \in \mathcal{C}$. These histograms are the $P(C=c_{\mathbf{x}}|A=a) \cdot P(A=a)$.

Given reasonable precision (*e.g.* 16 bits), each peer is represented by $N_{Bits} = N_{GlobalClusters} \cdot 16 + size(peerid)$.

Since the approach described above is based on the existence of global clusters, *i.e.* on data aggregated over the whole network, we have to show that the calculation of global clusters is not too expensive. This we regard improbable for two reasons: Firstly, because there are distributed variants of the k-means clustering where each peer has to communicate only its local means for each cluster centroid during each clustering run. Thus, the communication load of distributed clustering per peer is of the order $O(N_{GlobalClusters} \cdot n_{Dimensions} \cdot n_{Iterations})$, *i.e.* for a large number of documents per peer, the price of distributed clustering is still low compared to methods that would require sending over the network a large sample of all data stored within the peers [12].

The second argument is that there is no need to perform the clustering very often. As experiments (cf. secton 4.2) show, the performance found depends only weakly on the number of clusters, so we can assume that slightly outdated cluster centroids do not significantly affect the process.

4. EVALUATION

In our evaluation we assess the feasibility of our method for feature vectors derived from images. We measure if we can reduce the number of peers that have to be visited compared to a Gnutella flooding query, *i.e.* without the use of a peer content representation.

Experiments on CBP2PIR are hard: as there are no systems shipped to a wide public yet, there is no experience nor a snapshot about the data distribution over a whole P2P network. As a consequence, we had to create a synthetic data distribution over the peers that reflects our assumptions concerning the data distribution, namely that peers contain data concerning a small number of areas of interests. With respect to all other aspects we tried to make the experiments as hard as possible for the system, to obtain results that are rather low bounds on the performance given

Algorithm 1: Assigning images to peers

Data : $\mathcal{I}$ // all images in the collection
Data : $\mathcal{I}^*$ // all images not yet assigned to a peer
Data : $\mathcal{A}$ // a set containing all peers
Result : $\mathcal{M} : \mathcal{A} \rightarrow 2^{\mathcal{I}}$ // mapping peers to image sets

$\mathcal{I}^* \leftarrow \mathcal{I}$;
for $a \in \mathcal{A}$ /* each peer */ **do**
  $\mathcal{M}[a] \leftarrow \varnothing$;
  for $1 \leq b \leq N_{PeerClusters}$ /* each peer cluster */ **do**
    i $\leftarrow$ **random element from** $\mathcal{I}^*$;
    $\mathcal{M}[a] \leftarrow \mathcal{M}[a] \cup \{\mathbf{i}\}$; $\mathcal{I}^* \leftarrow \mathcal{I}^* \setminus \{\mathbf{i}\}$;
    for $\mathbf{x} \in \left\{ \frac{n_{Docs/Peer}}{N_{PeerClusters}} - 1 \right\}$ **best matches to i do**
      $\mathcal{M}[a] \leftarrow \mathcal{M}[a] \cup \{\mathbf{x}\}$; $\mathcal{I}^* \leftarrow \mathcal{I}^* \setminus \{\mathbf{x}\}$;

the said assumption and the use of synthetic data.

One aspect of making experiments hard for the system was to assume no replication. In many current file-sharing systems identical data items are present on a large number of peers. We assumed each data item to be present in exactly one peer.

4.1 Test environment

Our experiments were performed in four steps: (1) The feature data has been extracted from a set of images. (2) The images together with their feature vectors have been distributed to the peers in the network. (3) For each peer the peer data representation has been created. (4) A number of test queries have been performed, yielding statistics about the efficiency of the peer data representations. Each of these steps is described in more detail in the following.

4.1.1 Feature extraction

For our evaluation we used 162-dimensional color features derived from the HSV color space. We used color histograms and partitioned the color space into 18 intervals in the *hue* dimension, into 3 intervals in the *saturation* dimension and into 3 intervals in the *value* dimension.

4.1.2 Peer data distribution

As said above, we distributed the data over the peers according to our model that each peer contains data that belongs to a small number ($N_{PeerClusters}$) of areas of interest, the peer clusters or local clusters (as opposed to global clusters). As one of our peer data representation methods involves clustering, we tried to assure that we do not produce artifacts by finding local clusters and peer clusters by the same method.

The box named *Algorithm 1* presents the algorithm used to distribute the data over the peers. $n_{Docs/Peer}$ denotes the number of documents assigned to each peer. We kept this number constant for all peers as intuitively this should be harder than varying $n_{Docs/Peer}$, as peers with very little data would have a very accurate peer data representation, and peers with a large data collection would have a large prior probability to contain searched items. To each peer cluster we assigned $\frac{n_{Docs/Peer}}{N_{PeerClusters}}$ images.

Obviously, the peers considered early in the described distribution procedure will have rather coherent peer clusters whereas the peers considered later will have more inhomogeneous peer clusters.

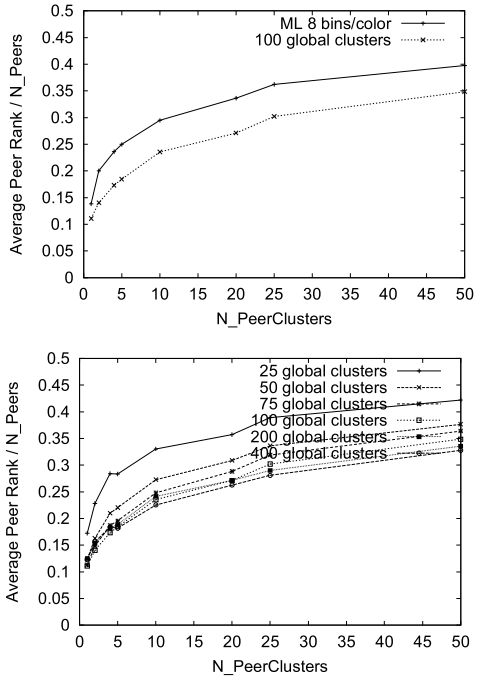

Figure 1: Top: The average peer ranks for the peers containing the M_{20} images for varying numbers of $N_{PeerClusters}$. Bottom: Varying $N_{GlobalClusters}$ from 25 to 400.

4.1.3 Create peer data representation

We used the peer data representations described in section 3.4 and 3.5. For the calculation of the global clustering needed for the latter representation we used k-means clustering.

4.1.4 Performing and evaluating test queries

We performed test queries in a simulation analogously to the PlanetP scenario. Since PlanetP assumes peer data representations to be present in every peer of the network and all queries to be transported directly from the query issuer to the query processor, PlanetP is simple to simulate using a single-threaded program on a single-processor machine, if we assume that the rumor spreading processes to distribute the administration information have been finished.

To assess our approach we considered the question: Does a consideration of the peers in decreasing order of $P(\mathbf{X} = \mathbf{x}|A=a)P(A=a)$ yield a better performance than random techniques and which of the proposed peer data representations performs better?

To this end, we chose the results of a centralized query engine as ground truth and asked ourselves questions such as: *How many peers do we have to visit on average to find the top 20 results?* or *On which peers do we find how many of the top 20 results with respect to the ordering according to our approximation of* $P(\mathbf{X} = \mathbf{x}|A=a)P(A=a)$*?*

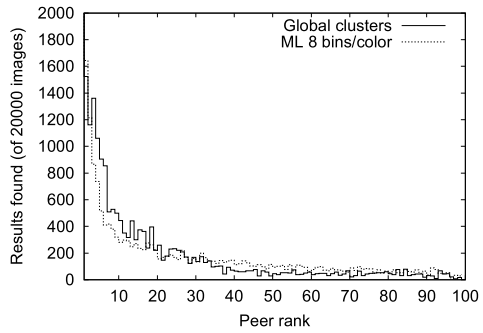

Figure 2: Number of images out of M_{20} found per peer rank over 1000 experiments

We evaluated our peer content representation schemes using 162-bin HSV color histogram data extracted from 10000 images provided by the Benchathlon CBIR benchmarking initiative [4]. In these experiments, each peer contained 100 images ($n_{Docs/Peer} = 100$).

We proceeded as follows: (1) We randomly chose a query image $\mathbf{i}$ from the image collection. (2) We did a centralized query for $\mathbf{i}$ on the collection, obtaining the 20 best matches M_{20}. (3) We queried the peers one peer after the other starting with the peers most likely to contain result images according to the peer content representations. This access order of the peers gave us a ranked list $R = (r_1, r_2, \ldots, r_{N_{Peers}})$ of all N_{Peers} peers. (4) We did some statistics on the number of the images out of M_{20} found on the peers with rank r_j.

4.2 Experimental Results

For each peer data representation method, 1000 test queries were performed and evaluated.

First we scanned the ranked list of peers R, and noted the peer ranks of the peers that contained each of the M_{20} images, obtaining a list L. For example, when the query image $\mathbf{x}$ itself was found in the 4th-ranked peer the pair $(\mathbf{x}, 4)$ was added to L. Then the mean over the peer ranks in L was calculated. Obviously, in the case of a random ranking of the peers, the average peer rank for peers containing an image out of M_{20} would be $\frac{N_{Peers}}{2}$.

These measurements were done for varying $N_{PeerClusters}$ and varying peer content representations. Figure 1 (top) shows the achieved average peer ranks for the peer data representation described in section 3.4 (ML 8 bins/color) and for the peer data representation described in section 3.5 (100 global clusters).

The figure depicts that the piecewise constant approximation of maximum likelihood under the assumption of statistical independence yields a worse performance than the approach based on 100 global clusters. The advantages over a random access order for the peers (which would yield a value of 0.5) are higher for small numbers of $N_{PeerClusters}$. The reason for this effect is that a peer with a small number of clusters has a more coherent image collection which can be better represented by the peer data representation.

The storage space requirements for both representations differ significantly. For the approach based on 100 global clusters we yield a representation of $2 \cdot 100 + 16 = 216$ bytes per peer, an order of magnitude less than the $162 \cdot 8 \cdot 2 + 16 =$ 2608 bytes used for the ML representation.

As mentioned in section 3.5, with the use of global clusters, an issue of concern is how often the centroids of the global clusters have to be updated *i.e.* how sensitive this method is with respect to changes of the clustering. In our experiments we evaluated this sensitivity by varying $N_{GlobalClusters}$ from 25 to 400. Figure 1 (bottom) clearly shows that the number of clusters does matter, however, even with the extremely small number of 25 global clusters a rather good result can be achieved.

To gain deeper insights into the considered techniques, we have counted how many images out of M_{20} were found on each peer rank. To this end, we fixed $N_{PeerClusters}$ to 5 and plotted the number of found M_{20} images per peer rank. Since, we performed 1000 test runs, a total of 20000 images was to be found and a random strategy would have values of $\frac{20000}{N_{Peers}} = 200$ for all ranks.

Figure 2 shows that the peer data representation based on a global clustering gets more result images on the peer ranks up to rank 20. Thus, this experiment also suggests that the cluster based representation outperforms the representation based on the assumption of statistical independence of the features.

The above results suggest that especially the cluster based peer representation can be used during query processing in a P2P network in the following way: Given a query $\mathbf{x}$ we determine the peers with the highest probability of containing good matches based on the compact peer data representations stored locally in the querying peer. Then only the most promising peers are queried to yield a result which will not be the optimal result but a good approximation obtained with a small fraction of the communication cost. Here the number of peers to be contacted can be chosen depending on the desired result quality and the communication cost one is willing to pay.

5. CONCLUSION

Doing efficient query routing for CBIR applications in P2P networks requires knowledge about the data in each peer. This problem becomes hard because of the high dimensionality of the data represented. Within this paper we have presented two methods for peer data representation, that permit to reduce the number of peers that have to be visited when processing a content-based image retrieval query. This is both motivating the work and giving room for improvement. We see the following directions of research:

Usage scenarios and evaluation: As in the whole area of P2P research, there is the question if the usage patterns will change, or if what we see is already the final usage pattern: people sharing much data they have not generated by themselves, and a high degree of replication of each item being a realistic scenario.

However, we see the necessity for algorithms supporting different scenarios, otherwise, P2P IR will be only usable for people looking for mainstream data, as non-replicated data will be found only by chance.

Research will have to find interesting usage scenarios, algorithms to cater for the needs of users in those scenarios, and ways to evaluate them in order to advance the field.

Peer data representations *vs.* distributed indexes: Deeper, real-world testing needs to be done to compare distributed indexes to source selection techniques based on peer

data representations. Fault resilience will be an important issue in this respect.

Peer data representations and distributed algorithms: The work presented in this paper motivates further research on useful peer data representations. Obviously, mixture models are an interesting topic in this respect. Other classification algorithms have to be evaluated for the possibility to learn the classifiers in a distributed (P2P) environment with bearable communication overhead.

6. REFERENCES

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *8th Intl. Conf. on Database Theory*, 2001.

[2] H. Balakrishnan, M. F. Kasshoek, D. Karger, R. Morris, and I. Stoica. Looking Up data in P2P Systems. *Communications of the ACM*, 46(2):43–48, February 2003.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *Proc. ACM SIGMOD*, Atlantic City, NJ, 1990.

[4] The Benchathlon Network. URL: `http://www.benchathlon.net`.

[5] S. Berchtold, D. A. Keim, and H.-P. Kriegel. The X-Tree: An Index Structure for High-Dimensional Data. In *Proc. 22nd VLDB*, San Francisco, CA, 1996.

[6] J. P. Callan, Z. Lu, and W. B. Croft. Searching Distributed Collections with Inference Networks. In *Proc. 18th ACM SIGIR*, Seattle, Washington, 1995.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, 2000.

[8] Clip2. The Gnutella Protocol Specification v0.4. URL: `http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf`, 2000.

[9] F. M. Cuenca-Acuna and T. D. Nguyen. Text-Based Content Search and Retrieval in ad hoc P2P Communities. Technical Report DCS-TR-483, Department of Computer Science, Rutgers University, 2002.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[11] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining, LNAI*, 2000.

[12] M. Eisenhardt, W. Müller, and A. Henrich. Classifying documents by distributed P2P clustering. submitted for publication, 2003.

[13] Gnunet homepage. URL: `http://www.gnu.org/software/GNUnet/`.

[14] L. Gravano, H. Garcia-Molina, and A. Tomasic. The Effectiveness of GlOSS for the Text Database Discovery Problem. In *SIGMOD Conf.*, Minneapolis, MN, USA, 1994.

[15] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack. Efficient color histogram indexing for quadratic form distance functions. *IEEE Transactions on PAMI*, 17(7):729–736, 1995.

[16] M. Harren, J. M. Hellerstein, R. Huebsch, B. T. Loo, S. Shenker, and I. Stoica. Complex Queries in DHT-based Peer-to-Peer Networks. In F. Kaashoek and A. Rowstron, editors, *1st Intl. Workshop on Peer-to-Peer Systems*, MIT, Cambridge, MA, USA, 2002.

[17] S. Joseph. Adaptive routing in distributed decentralized systems: Neurogrid, gnutella and freenet. In *Proc. of workshop on Infrastructure for Agents, MAS, and Scalable MAS, at Autonomous Agents*, Montreal, Canada, 2001.

[18] A. Z. Kronfol. A Fault-tolerant, Adaptive, Scalable, Distributed Search Engine. URL: `http://www.searchlore.org/library/kronfol_final_thesis.pdf`, May 2002. Final Thesis, Princeton.

[19] C. H. Ng and K. C. Sia. Peer clustering and firework query model. In *Poster Proc. of The 11th Interational World Wide Web Conf.*, Honululu, HI, USA, 2002.

[20] C. H. Ng, K. C. Sia, and C. H. Chang. Advanced peer clustering and firework query model in the peer-to-peer network. poster in WWW2003, *(accepted)*, 2003.

[21] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. 2001 Conf. on applications, technologies, architectures, and protocols for computer communications*, San Diego, CA, United States, 2001.

[22] J. Ritter. Why Gnutella Can't Scale. No, Really. URL:`http://www.darkridge.com/~jpr5/doc/gnutella.html`, February 2001.

[23] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Proc. 18th IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, 2001.

[24] A. W. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content based retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, 2000.

[25] D. M. Squire, W. Müller, H. Müller, and J. Raki. Content-based query of image databases, inspirations from text retrieval: inverted files, frequency-based weights and relevance feedback. In *11th Scandinavian Conf. on Image Analysis*, Kangerlussuaq, Greenland, 1999.

[26] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proc. ACM SIGCOMM Conf.*, San Diego, CA, USA, 2001.

[27] C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information retrieval in structured overlays. In *First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, NJ, 2002.

[28] N. Vasconcelos. *Bayesian Models for Visual Information Retrieval*. PhD thesis, MIT, June 2000.

[29] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. Intl. Conf. on VLDB*, New York, USA, 1998.