Sample-based Creation of Peer Summaries for Efficient Similarity Search in Scalable Peer-to-Peer Networks*

Daniel Blank Soufyane El Allali
Wolfgang Müller Andreas Henrich
Faculty of Information Systems and Computer Informatics
Chair of Media Informatics
University of Bamberg
D-96045 Bamberg, Germany
first-name.last-name@wiai.uni-bamberg.de

ABSTRACT

In this paper we introduce a simple yet experimentally convincing approach in the research field of source selection for content-based similarity search in P2P networks or, more concretely, in summary-based P2P systems. In these systems, summaries are used for data source selection when performing k-NN queries on distributed collections of documents represented by feature vectors.

We introduce a new type of cluster-based summaries for source selection that can efficiently and cheaply be calculated and distributed in P2P networks. For the summaries generation, a very large number of sample points is used. Each peer in the network assigns its indexing data to their corresponding closest sample points and publishes its constructed summary. We evaluate the quality of these summaries when changing the number of sample points used in experiments on real-world image feature data obtained from a large crawl of the flickr web photo community and show that for higher numbers of sample points we achieve a better retrieval performance. Our experiments show that the proposed summaries yield four times better performance with respect to previous methods. Intuitively, there are some disadvantages to this approach due to the large size of the generated summaries. We show experimentally, that these disadvantages can easily be overcome due to the sparse nature of the generated summaries by simple compression techniques.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process, Selection process; H.3.5 [**Online Information Services**]: Data sharing

This work is partially funded by the Deutsche Forschungsgemeinschaft DFG HE 2555/12-1

General Terms

Algorithms, Experimentation, Performance

Keywords

P2P, CBIR, Source selection

1. INTRODUCTION

User-generated data is at the center of the Web 2.0 revolution. Some Web 2.0 community services such as *YouTube.com* or *flickr.com* are used by millions of people. Users provide parts of their self-generated content by uploading their data to servers and annotating it so that others can retrieve, browse, and/or download it. As this kind of service is quite attractive for both users and companies (flickr.com and YouTube.com recently have been bought by *Yahoo!* and *Google* respectively), this motivates enabling similar services in a peer-to-peer (P2P) environment [11]. Furthermore, current shortcomings of Web 2.0 systems[1] suggest that the collaborative annotation ("tagging") of content items has to be complemented with content-based search techniques.

Our vision is a large-scale P2P network that allows for the sharing of user-generated data, and that provides both tag-based and content-based search. Different methods for content-based search have been proposed [27] most of which boil down to feature vector search. There are three main classes of approaches that have been proposed in this regard of similarity search on feature vectors:

1. DHT-based distributed indexing structures [10] seek to specialize nodes on regions in feature space. Each node "knows" only a small region in feature space. Joining the network is expensive, but search is efficient.

2. Unstructured approaches such as super-peer networks[2] consist of super-peers, which act like servers, and normal peers. Such networks are flexible and powerful. However, super-peers know very little about other peers outside of their local super-peer network.

[1]According to a representative crawl of 666,909 flickr images only about 60% of the images at flickr are tagged, and the average number of tags of an image with one or more tags is 4.0.
[2]Super-peer networks without summaries, we later propose the use of super-peer networks *with* summaries.

3. Summary-based methods such as [6, 19] seek to distribute data summaries to other peers. These summaries are good for efficient retrieval *and* they are able to provide an overview of the collection.

All the above approaches have their merits and sometimes, depending on the type of application in hand, a relatively similar retrieval performance (see for example [8]). However, when we try to combine retrieval performance with the knowledge about the properties of an entire network's data collection, for instance in the case of content-based browsing applications, summary-based approaches are the ones that are most suitable for such applications. Summary-based approaches offer an overview of the network's global data distribution, which is useful in browsing applications. For example, porting PicHunter [5], which offers user feedback capability for content-based browsing of images, to P2P systems becomes feasible given the large number of sample points that represent the global state of the network's data collection.

In this paper we extend earlier work on how to build summaries for efficient source selection in summary-based P2P systems. We follow the approach of summary-based source/peer selection in PlanetP-like networks [6]. In PlanetP every peer knows the summary of every other peer's data, enabling each peer to choose the most interesting peers given a query. Scalable variants of this idea exist [29, 19]. A detailled description of our scenario can be found in Section 2.

Usually content-based indexing is realized by indexing d-dimensional feature vectors. In our PlanetP setting, every peer knows a set of k centroids. In order to summarize its local data, a peer assigns every feature vector of its local collection to the closest centroid, i.e. the documents are clustered using the centroids. A peer's local data summary will be represented by a *count* of documents belonging to each cluster, constructing therefore what we refer to as a *cluster histogram* of each peers indexing data. In previous work Eisenhardt et al. [7] showed experimentally using real-world data that a choice of centroids that are computed by an elaborate clustering process brings only small retrieval performance improvements when compared to a randomly chosen set of centroids from the underlying collection. We will refer from now on to this randomly selected set of centroids by *sample points*. When performing cheap selection and distribution of these sample points, we are able to choose a very high number of globally known sample points to all peers in the network. This makes the constructed summaries very fine-grained and thus they perform much better than the ones described in [7] when using a smaller number of well-chosen centroids.

Intuitively, one would suppose that, given the large number of sample points used to generate the summaries, this method has the following efficiency problems. (i) We would expect the summaries to be of large size. (ii) We would expect that, given that the summaries are fine-grained and precise, we have to adapt the sample points (and the summaries) very often to changing collections.

In contrast to the above assumptions, according to the experiments presented in this paper we show that (i) The summaries are very sparse and can thus be compressed to very small sizes. (ii) The set of sample points used for obtaining the summaries depends only very weakly on the collection and can thus be hard-coded into the indexing software. This makes shipping the sample points with the indexing software a viable possibility. Consequently, the performance costs that are induced by mechanisms that seek to cluster the network's data collection in order to determine the cluster centroids are negligible in our case. In addition to these aspects we show that query costs, i.e. the number of peers to be contacted for processing a query, is reduced by a factor of four when compared to the previously used summaries in [7].

The remainder of this paper is organized as follows: In Section 2 we present the P2P architecture and the data source selection approach used together with a detailled description of the peer summaries. Section 3 explains our experimental setup, the data acquisition, the performance measures, and the results obtained from our experiments. In this Section we furthermore compare our results with results from our previously used Gaussian Mixture Model based approach [9] as well as a distributed indexing structure based on Content-Addressable Networks (CANs) [10]. Section 4 takes a closer look at related work and Section 5 finalizes our paper with a conclusion and an outlook on future work.

2. SUMMARY-BASED SIMILARITY SEARCH

2.1 Summary-based P2P architecture

PlanetP and extensions: Our considerations are based on PlanetP-like networks. As stated in the previous section, in PlanetP [6] each peer knows the summary of every other peer participating in the network. This makes routing simple and the network extremely robust. As a downside, this approach does not scale. Depending on the churn rate (i.e. the number of peers joining and leaving per minute related to the total number of peers in the network) and the type of summaries used, PlanetP starts to fail at a couple of thousands of peers. When the number of nodes in the network and the churn rate are too high, the peers are mainly busy forwarding summaries and the network is not able to process queries anymore.

However, Rumorama [19], a scalable variant of PlanetP has been proposed. Rumorama builds hierarchies of networks that are accessible by an efficient multicast. Its leaf networks behave like PlanetP. Therefore, while we examine the summary-based ranking of peers in PlanetP-like middle-sized networks, we can easily extend this to large-scale Rumorama-like networks.

On the other hand, while we are using the approach in a PlanetP-like setting, we think that it can also be applied to a super-peer network similar to [12] in order to enable similarity search. In this case, each (normal) peer creates a summary of its data and transfers it (i.e. the *summary*) to the responsible super-peer. A query originating from a (normal) peer can be processed by forwarding it to the corresponding super-peer, which sends the query to all remaining super-peers in the network and afterwards collects and merges the result sets obtained from the super-peers (i.e. *flooding at the super-peer level*). Alternatively, the super-peers themselves could summarize their data computing aggregated summaries and perform PlanetP-like search at the super-peer level. Every super-peer itself has to determine which peers to contact in its sub-network in order to find the most similar documents.

PlanetP architecture for summary-based similarity search on images: The original PlanetP implementation is designed for text documents. Extensions of PlanetP proposed cluster-based summaries for image data using a distributed k-Means clustering and an approach that randomly selects sample points out of the data [7, 8]. Within this paper we further develop the approach based on random selection of sample points and show that this leads to significantly better results.

PlanetP-like networks hold the *original data* (i.e. the images) together with the *indexing data* (typically a high-dimensional feature vector representing different properties of an image, such as color distribution, texture, or shape) on the same peer. As opposed to many distributed indexing structures, no local indexing data resides on remote peers; indexing data remains at the peer that contains the original data. This brings several benefits to the network: At first, when a peer joins the network no indexing data needs to be transferred to remote peers. Secondly, when a peer leaves the network, it takes all its original *and* indexing data with it. It is not possible that indexing data of a peer that has already left the network is still present on a remote peer. Only summaries remain in the network. Expiring them is simpler and cheaper than expiring full indexing data. Situations in which peers leave the network, containing indexing data of images that are still present in the network and therefore should still be searchable, are avoided. All these issues become important as P2P networks are highly dynamic. Therefore peers joining and leaving the network impose a main cost factor to a P2P network.

In our P2P scenario *peer data summaries* are used in order to compactly aggregate the documents of a single peer. Only these short summaries are distributed within the network. Queries are routed to peers potentially containing interesting data. Each peer that is contacted during query execution will process the query locally, contributing to the query result. The query processing consists of identifying peers that probably contain relevant data (i.e. ranking the peers), forwarding the query to them, and afterwards collecting and merging the results. The query performance is determined by the *quality of the summaries* and the *peer selection method*. We will strive to use summaries that are i) *simple to generate*, ii) *cheap to distribute* and iii) *selective*.

As the creation of peer data summaries is based on a Large number of Sample Points (see Section 2.2) and the peer selection mechanism uses StableSortRanker (see Section 2.3), we will refer to the method presented within this paper as *LSP-SSR*. The issues of summary creation and peers selection are addressed next and solutions are presented in the following.

2.2 Creation of summaries for efficient peer selection

A set of centroids is usually used for computing the summary of a peer. We use in our approach a set of sample centroids $C = \{c_i | 1 \leq i \leq k\}$ that we refer to as *sample points*. Each peer's data is represented by a *sample histogram* s of size k. A peer assigns every document o_j, according to a certain distance function, to the closest sample point $c_i \in C$ and afterwards increments the absolute frequency $f_i \in \mathbb{N}$ of the specific sample histogram bin i. The summary s_α of a peer p_α is therefore represented by $s_\alpha = \{f_{i,\alpha} | 1 \leq i \leq k\}$ the set of absolute frequencies $f_{i,\alpha}$ that indicate how many documents of peer p_α are closest to a certain sample point c_i. Let us quickly review the properties of the summaries obtained:

Simple to generate: Eisenhardt et al. [7] investigated the utilization of centroids that are computed using different methods, namely distributed k-Means and showed that the use of randomly selected sample points as centroids has little influence on overall retrieval performance and can be used instead of distributed k-Means for finding centroids (although there is small loss in performance). The only task that a peer needs to perform for computing the summary is to assign every single document to one of the k sample points that have been provided to the peer in advance; no other collection-wide information or distributed computation (e.g. clustering) is needed.

Cheap to distribute: On entering the P2P network, each peer needs to know the k sample points. As these sample points can be provided during software installation or update, only sample histograms need to be exchanged. There is no longer a need to distribute the sample points as we are not using any distributed clustering. Another positive side effect is that the need for recomputing the cluster centroids as new images are added to the peers' collections is eliminated. As new images are added to the peers' image collections they are simply assigned to their closest sample points, which takes place only at indexing time. As a consequence, this allows for high values of k. At first glance, this seems to induce a linear growth to the size of the peer data summaries. However, we will show that higher values of k result in more sparse peer data summaries, and therefore applying data compression yields a highly sublinear increase in the effective size of the peer data summaries.

Selective: As our results given in the following sections show, the summaries can be a good basis for selecting the most promising peers. By increasing k, the number of sample points, we gain more selective summaries and therefore overall performance in query processing. How to rank peers according to the information given by the summaries is described in the following.

2.3 Retrieval using compact peer descriptions

Retrieval in our P2P scenario works as follows: First of all, a user states a query (in our case an example query document) locally. Afterwards the peers are ranked w.r.t. the query based on their summaries and C, the set of sample points. The querying peer can easily determine which of the k sample points is the closest to the query according to a certain distance function. Since the querying peer knows all the summaries, it can determine which peers are most likely to contain documents relevant to the query and rank the peers by their likelihood to hold relevant documents. *StableSortRanker* [7] determines a ranking (i.e. an ordered list) of peers in order to answer a certain query. The rank of a peer in this ordered list is called *peer rank*. The querying peer will afterwards contact the remote peers in ranked order up to a certain point and obtain lists of relevant documents it can merge.

StableSortRanker is a ranking mechanism that makes its decision based on $L_{samples}$, a list representation ordering all $c_i \in C$ in ascending order w.r.t. their distance to the query. The first element of this list therefore always corresponds to the sample point that is closest to the query. $L_{samples}$ is processed starting with the first element, i.e. the

sample point closest to the query, until it is either possible to make a decision or the end of $L_{samples}$ has been reached. Peer p_α is ranked higher than peer p_β if $f_{i,\alpha} > f_{i,\beta}$. If peer p_α and peer p_β both have the same amount of documents assigned to a certain sample point ($f_{i,\alpha} = f_{i,\beta}$), *StableSort-Ranker* chooses the next element c_j out of $L_{samples}$ and recursively compares peer p_α and peer p_β according to $f_{j,\alpha}$ and $f_{j,\beta}$. This process consumes more processor time for higher values of k. However, the source selection process is performed only once per query. In large networks, the processing power spent for source selection will vanish against the processing load incurred by handling queries from other peers.

3. EXPERIMENTS

In the following we describe the data that we use in our experiments as well as the method that is used for measuring retrieval performance before we present our results. At the end of this experimental section we compare our results to other methods.

3.1 Data acquisition & distribution

In our experimental setting we use a real world data set. The data obtained is a subset from a crawl of flickr.com, a web-based community portal to store, share and search images. Each user can store an arbitrary number of photographs and other pictures in his/her account. We take a randomly chosen subset of 10,961 user accounts so that the total number of images in these accounts is 250,000. We assign each user account to one peer to simulate flickr.com in a P2P setting, which results in $|P| = 10,961$ peers for our scenario where P is the set of all peers. The feature set used to index the images is a 36-dimensional color histogram based on the HSV color model as used in [8]. Obviously there exists other feature sets for describing an image, but within the scope of this paper we are showing experimentally the efficiency of our summary-based query technique and we will address the issue of quality of retrieved images by using more elaborate features in our future work.

3.2 Defining a performance measure

In our experiments we always search for the top-20 images matching a certain query. We are interested in how many peers are to be contacted in order to retrieve a fraction of the 20 most relevant (i.e. closest) documents w.r.t. Euclidean distance. We measure the *Average Peer Rank* $APR_{20}(n)$. It describes how many peers need to be contacted *on average* to find at least n of the top-20 matches for a given query, where the query is the feature vector of a randomly selected image from the data collection. In our experiments the APR is averaged over 1,000 queries in order to minimize the influence of outliers on the results. We calculate the APR as a percentage of the total number of peers $|P|$ in the network ($|P| = 10,961$ in our case) to make results more comparable. We choose this measure since contacting other peers during query processing, sending the query and receiving the result sets of the other peers are the main query cost factors in our P2P information retrieval scheme.

The top-20 matches are computed based on the global document collection. This is done prior to executing the query in the P2P network and can be seen as a baseline against which to compare the P2P retrieval system.

3.3 Random selection of sample points

As we plan to distribute C, the set of sample points, a priori together with the installation software of our P2P system, we first examine if this strategy is applicable as the peers' image collections change over time. In the following we therefore investigate three different strategies. At first we select the sample points out of the underlying data collection and therefore use sample points that reflect the current data collection. Secondly, we use a disjoint collection of flickr images. Lastly, we choose an image collection with images being independent of flickr.com as a worst case collection being independent of the application domain of our P2P scenario.

Selecting sample points from the underlying Flickr collection (flickr250k): The strategy that is the easiest to simulate is simply choosing the sample points from the underlying data collection. This strategy is useful, as regions out of the feature space that contain many feature vectors will be represented by many sample points. The strategy can be implemented in the protocol that is used for communication between super-peers. One simple way could be that super-peers, when executing queries of (normal) peers, simply track the query documents (and maybe additionally the result sets they obtain from remote peers) that constitute C.

Selecting sample points from a disjoint flickr collection (flickr50k): In another strategy we pick the sample points from a second, disjoint collection of flickr images and apply these sample points on the collection that has been described in Section 3.1. Therefore we used a second crawl of 50,000 flickr images from which we picked the sample points. It is important that there is *no overlap* between these two collections. With this, we test if large samples within photo communities are so much alike that they describe any large randomly-picked subset of images in such a photo community.

Selecting sample points from a collection of non-flickr images (www9908): To show the robustness of our approach we use a third strategy. If it is not possible to track query documents and there is no chance to crawl images from the collection indexed in order to use their features as sample points, one has to think of an alternative way, i.e. a worst case strategy of how to obtain the sample points anyway. We therefore used random images out of a collection made up of 9,908 images [16, 28] representing a web crawl.

The application of the three presented strategies is shown in Figure 1 for $k = 4096$, where the performance of *flickr250k* and *flickr50k* nearly coincides. The performance of the third strategy, where the sample points are chosen independently of flickr.com performs not as well as the other strategies but still leads to much better results than described in earlier work [7, 8] (see Table 1 for $k = 256$). Note that we can easily exploit the fact that the performance of the other strategies only marginally differs: We can deduce that the sample points have to be replaced by better-adapted sample points only very rarely, so we can handle them as one would handle a P2P software update.

The performance of all these strategies can be further improved by increasing k, which is comparatively low in Figure 1 ($k = 4096$), as we describe in the following. In our experiments, unless stated otherwise, we apply the *flickr50k* strategy.

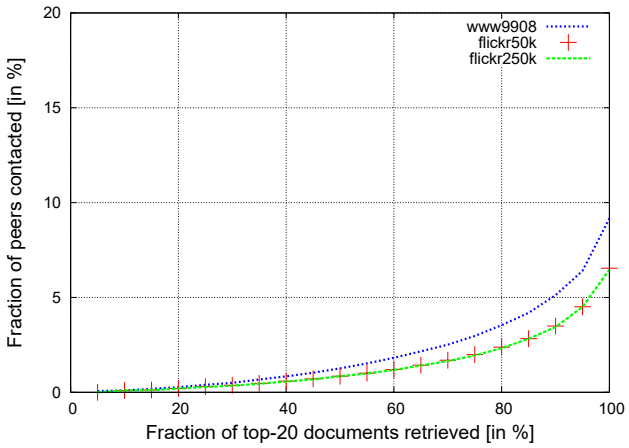

Figure 1: Different sources for sample points

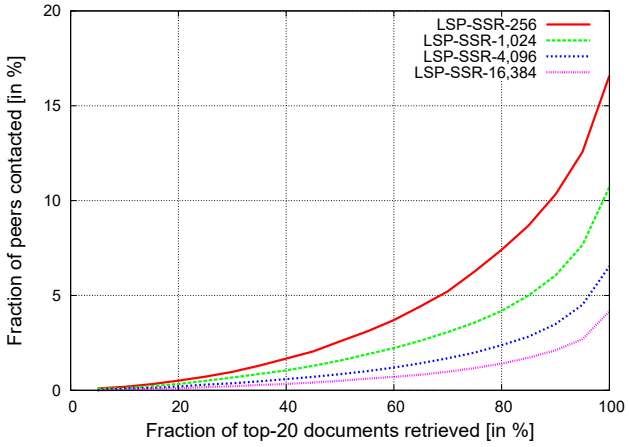

Figure 2: Peers contacted with increasing k

k	60% top-20	80% top-20	100% top-20
256	3.7%	7.4%	16.6%
1,024	2.2%	4.2%	10.7%
4,096	1.2%	2.4%	6.5%
16,384	0.7%	1.4%	4.2%

Table 1: Peers contacted for top-20 [in %]

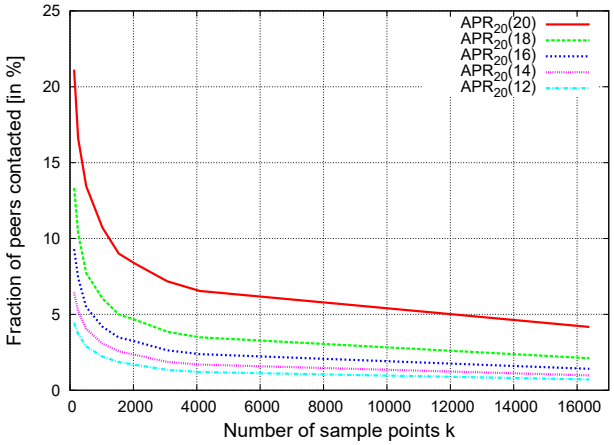

Figure 3: Different APRs with increasing k

3.4 Boosting performance by increasing k

Obviously one important parameter in our approach is k, the number of sample points used for computing the summaries. Higher values of k increase the size of the summaries and therefore the amount of data that needs to be transferred throughout the network but they promise more fine-grained peer selection decisions. Figure 2 shows that we obtain better results the higher we choose k. If we use 16,384 sample points, it is possible to retrieve, on average, all of the top-20 images by contacting approximately 4% of the peers, a 4 times improvement with respect to using 256 sample points. We show later that the size of the summaries can be reduced by compression (see Section 3.5).

Typically a user is satisfied when retrieving not all of the top-20 documents. Table 1 shows that if a user is interested in e.g. 16 out of the top-20 documents only 1.4% of the peers need to be contacted with $k = 16,384$. This is less than 1/5 of the peers that are to be contacted when choosing k equal to 256.

Figure 3 visualizes different APRs w.r.t. k. It shows that the decision to increase k is much more important if k is not too big ($k < 4,000$ in this case), especially if the person who states the query is interested in high precision (e.g. $APR_{20}(20)$ or $APR_{20}(18)$). For bigger values of k the APR decreases more smoothly, still indicating that a bigger k always leads to an increase in overall performance while the cost of creating the bigger summaries is not hindering this performance since (i) every peer builds his summary individually and (ii) as new images are added to the peer's collection only the peer summary's affected bins are changed once per added image.

3.5 Compressing the summaries

When $k \gg N_{p_\alpha}$, where N_{p_α} is the total number of documents of a peer p_α, this results in sparse (many summary bins without any assigned documents) histograms/summaries with a high potential for applying compression techniques. Even if k is comparatively small, compression is still beneficial, as peers/users usually tend to store similar images according to the features (because people have e.g. taken a picture of the same location under the same lighting conditions from the same or a similar point of view several times).

As compression algorithm we applied run-length encoding (RLE) and then zipped the resulting RLE compressed summaries. RLE is easy to implement and quick to use and can be seen as a baseline for more elaborated compression techniques. Figure 4 shows the gain when compressing the summaries. The size of the summaries grows only logarithmically with increasing k. Even if we think of more than 16,000 sample points/summary bins, the average size of a summary in our scenario is approximately 110 bytes. This is for example much less than the amount of data that is transferred using Bloom filters [3] for text data as described in [6] and the size of uncompressed summaries with k being equal to 256 as used in [7, 8].

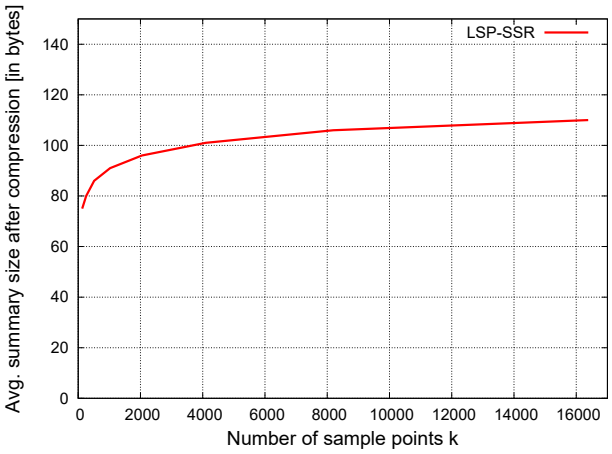

Figure 4: Compressing the summaries

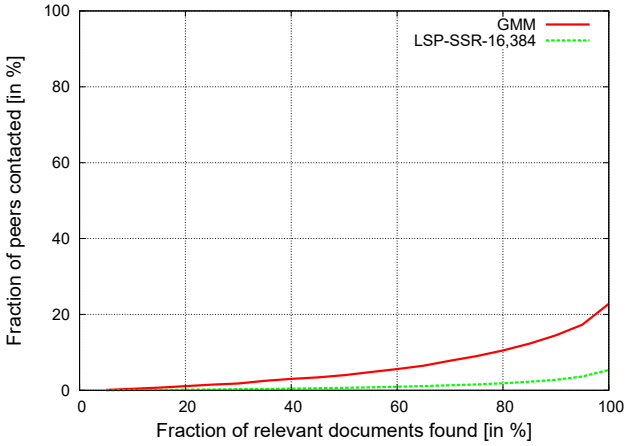

Figure 5: Retrieval performance using GMM approach vs. random points selection approach

3.6 Comparisons

In the next two sections we compare the previously described results with a Gaussian Mixture Model approach [9] for summary-based P2P image retrieval and a CAN [21] based approach, an alternative for multidimensional indexing in P2P networks.

3.6.1 Comparison with Gaussian Mixture Models approach

In our previous work [9] a probabilistic view was used as an approach for retrieval and similarity search in summary-based P2P systems. This approach treats the image retrieval problem as a vector classification problem. We can define a mapping from images to image classes. It has been shown that this view is flexible and successful [26]. In our case we tried to find the best mapping between query images and peers in the network. This mapping maps a query image feature vector $\vec{x}$ to the peer p_α that is most likely to contain similar feature vectors to $\vec{x}$. In other words we choose the peer that maximizes the probability $p(p_\alpha|\vec{x})$[3]. Using Bayes rule this can be defined as follows:

$$\begin{aligned} g^*(\vec{x}) &= \underset{p_\alpha}{\text{argmax}}\Big(p(p_\alpha|\vec{x})\Big) \\ &= \underset{p_\alpha}{\text{argmax}}\Big(\frac{p(\vec{x}|p_\alpha)p(p_\alpha)}{p(\vec{x})}\Big) \\ &= \underset{p_\alpha}{\text{argmax}}\Big(p(\vec{x}|p_\alpha)p(p_\alpha)\Big) \text{ since } p(\vec{x}) \text{ is constant} \end{aligned}$$

where g is a mapping from indexing data $\vec{x} \in X$ to the peers. $p(\vec{x}|p_\alpha)$ is the probability that $\vec{x}$ is found in peer p_α, and $p(p_\alpha)$ is the prior probability of drawing the result for $\vec{x}$ from p_α without any prior knowledge other than p_α's size. $p(p_\alpha)$ is proportional to the number of documents contained in peer p_α.

The goal in this case is to find a representation for every peer in the network that enables the estimation of the probability to find a given data vector $\vec{x}$ within a given peer p_α. In order to achieve this, every peer generates its own model that we refer to as *peer model*, this model is then shared with other peers. Once a query is issued, the querying peer makes a ranking of peers to contact based on their models.

The *peer model* is a Gaussian Mixture Model (GMM) that represents the indexing data that a peer holds. The GMM parameters set $\mathbf{\Theta}$ is computed using the Expectation-Maximization algorithm. Once the GMM parameter sets $\mathbf{\Theta}$ are generated by the peers they are distributed in the PlanetP network in order for every peer p_α to be able to reconstruct peer p_β's GMM given $\mathbf{\Theta}_{p_\beta}$. This distribution process is taken care of by the PlanetP setup.

When a peer requests a query for $\vec{q}$, all it needs to do is figure out which other peer's distribution in the network produces the maximum likelihood for $\vec{q}$. Hence, a ranking R of the peers is produced based on this likelihood such that:

$$p(p_\alpha|\vec{q}) \geq p(p_\beta|\vec{q}) \Rightarrow R(p_\alpha) \geq R(p_\beta)\ \forall \alpha \neq \beta$$

where $p(p_\alpha|\vec{q})$ is the probability of a peer p_α given a query $\vec{q}$ such that:
- $p(p_\alpha|\vec{q}) = p(\vec{q}|p_\alpha)p(p_\alpha) = p\left(\vec{q}|\mathbf{\Theta}_{p_\alpha}\right)p(p_\alpha)$.
- $\mathbf{\Theta}_{p_\alpha}$ is the GMM parameters set corresponding to p_α.
- $p\left(\vec{q}|\mathbf{\Theta}_{p_\alpha}\right)$ is the probability of $\vec{q}$ given a model's parameter set $\mathbf{\Theta}_{p_\alpha}$.

Therewith, we rank p_α higher than p_β given a query $\vec{q}$ if the query has higher likelihood to come from p_α's model than from p_β's one.

We compare our new method results with the results of the GMM approach in Figure 5 where we use a second collection (50k) that consists of 50,000 images distributed over $|P| = 2,623$ peers due to simulation computational limitations resulting from the GMM computational complexity. From the figure we see that the *LSP-SSR* approach, where th sample points are chosen out of the 250k collection, outperforms the GMM approach. It is approximately 5 times better on average to retrieve all of the top-20 documents for a given query.

We compare in the next section our similarity search approach based on random sample points to a CAN extension approach allowing similarity queries in P2P networks.

3.6.2 Comparison with CANs

CANs [21] have been the first multidimensional indexing

[3] $p(p_\alpha|\vec{x})$ in itself means only maximizing the probability of finding $\vec{x}$ in p_α. However, if we assume that the probability distribution over peers is smooth, $p(p_\alpha|\vec{x})$ also means finding the peers that most probably contain similar documents to the query $\vec{x}$.

structures for P2P networks. They allow for the storage of key/value-pairs. Every key is represented as a d-dimensional vector $\vec{v} \in [0;1]^d$ out of a d-dimensional unit cube.

In a CAN every peer is responsible for an axis-aligned box within the unit cube. All key/value-pairs $(\vec{v}, x)$ that are indexed in the CAN are administered by the node/peer that is responsible for the region containing $\vec{v}$. Every peer p_i holds connections to the peers that are responsible for regions in the neighborhood of the responsibility region of p_i. In their original form CANs allow efficient (exact) membership queries. In a d-dimensional CAN with $|P|$ nodes $\mathcal{O}(d \cdot \sqrt[d]{|P|})$ routing steps need to be undertaken in order to answer membership queries[4].

Similar to [25, 10] we use the extensions to CANs that allow for similarity queries. Therefore we need to make design decisions, that basically affect the so called *split strategy* and the *query execution*.

In standard versions of CANs new peers incorporate into the network immediately. In our experiments we assume (for comparison issues) that new peers queue before entering the network. During insertion of a new key/value-pair the responsible peer p_{v_n} of a new vector $\vec{v_n}$ is chosen. If it contains more than n_{split} key/value-pairs, the region for which p_{v_n} is responsible is split. Therefore a new peer p' is drawn from the queue and is incorporated into the network. The performance of the CAN is mainly depending on the *split strategy*, i.e. the selection of the dimension based on which the responsibility region of p_{v_n} is split into two new responsibility regions.

When performing an exact query it is sure that the desired key $\vec{q}$ is located in a single peer, i.e. the peer that is responsible for $\vec{q}$. In contrast to that, when applying similarity queries, it is not even sure that the most similar vector w.r.t. the query vector $\vec{q}$ can be found at peer p_q, that is responsible for $\vec{q}$. One therefore has to find peers, starting with p_q, that contain the k-nearest neighbors (k-NNs) of $\vec{q}$. We choose both the *split strategy* and the mechanism used for *query execution* in a way, so that the approach based on CANs performs as good as possible w.r.t. the properties that are measured in our experiments:

Split strategy: In contrast to [10] we use a split strategy that depends on the data that is used. For the node p_s that is to be split, mean and variance are computed along each dimension of the data that is contained in p_s [13]. As split dimension i_s the dimension with the highest variance is chosen and the data collection is split along this dimension, so that the points $\vec{v}$, whose i_s-th component v_{i_s} is smaller than the mean, stay inside p_s. The remaining key/value-pairs are migrated into the new peer.

Query execution: Query execution consists of two steps. Firstly, the peer p_q that is responsible for the query vector needs to be found and contacted; p_q will then own the query process. When needed, it will contact neighbors (i.e. candidate peers p_c) that could possibly contain one or more of the k-NNs of $\vec{q}$.

Therefore p_q administers a priority queue, which contains points and regions ordered w.r.t. their distance to $\vec{q}$ [14]. If a point is chosen from the priority queue, it is a k-NN of $\vec{q}$. If a region is chosen, it could possibly contain a k-NN.

[4]Small-world variants of CANs that offer logarithmic complexity exist (e.g. [10]). Nevertheless, for the high dimensional spaces that we are looking at, the saving that is expected hereby is not relevant.

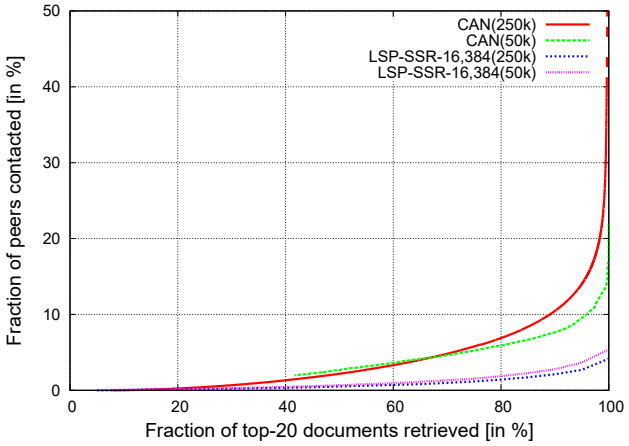

Figure 6: Growing number of peers

Therefore the responsible peer is contacted. It sends a list of points and regions to p_q. Afterwards peer p_q inserts them into the priority queue. This is continued until all of the k-NNs are found or the queue is empty.

Figures 6 and 7 – explained in detail below – show that *LSP-SSR* performs better than our CAN implementation. Besides the fact that network traffic under churn can be reduced with the use of our system (compared to the CAN implementation), there are more benefits. Firstly, the relative performance of the CAN implementation significantly drops when the total number of peers/documents that are simulated is increased. On the other hand, the performance of *LSP-SSR* is not influenced by the increase in the number of peers. This issue is shown in Figure 6, where we use a second collection (50k) that consists of 50,000 images distributed to $|P| = 2,623$ peers in addition to distributing 250,000 images (250k) to $|P| = 10,961$ peers in the same manner as described earlier (see Section 3.1).

The second factor that negatively influences the relative performance of the CAN implementation more than it influences the approach based on cluster summaries is the dimensionality of the feature space. The effects are shown in Figure 7 using the 50k collection with $|P| = 2,623$ peers, where the sample points are chosen out of the 250k collection. We additionally applied a 166-dimensional feature set uniformly quantizing the HSV color space [7]. Whilst *LSP-SSR* only needs to contact less than 10% of the peers, the CAN implementation finds the top-20 after contacting approximately 40% of the peers for this feature set.

4. RELATED WORK

The work on P2P Information Retrieval is diverse. A broader state-of-the-art is given in [20] and [19]. We only list some selected approaches that try to enable similarity search in distributed scenarios.

It is possible to implement similarity search in structured P2P networks like PRISM [22] based on distributed hash tables (DHTs). Several extensions of CANs (i.e. [25, 10]) also allow for similarity search. Minerva [2] administers summarizations of local indexes in a DHT applying peer selection and query routing strategies for answering mainly textual queries.

Purely routing-based P2P networks have been proposed as

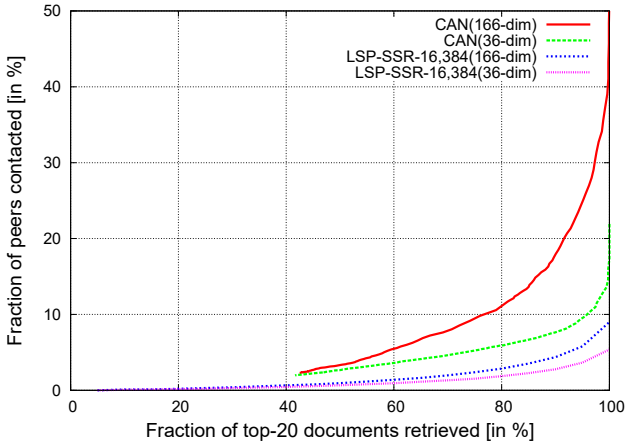

Figure 7: Analyzing the dimensionality

alternatives to the approaches relying on distributed indexing structures. DISCOVIR [15] supports similarity queries based on images by routing queries to clusters of peers with similar summaries. The use of summaries for distributed similarity search is common. Successful uses of summaries have been described for text data [6, 2] and image data [19]. Chang et. al [4] were the first to present a histogram-based summarization technique based on image templates and a resource selection algorithm for distributed image databases.

Other approaches allowing for similarity search in P2P networks are based on replication [23] or super-peer architectures (e.g. [24, 29, 17]).

In [1], SWAM, a family of distributed access methods is presented. One of these makes use of the small-world model and partitions the key space into a Voronoi diagram. The authors claim that the query cost is independent of the dimensionality of the feature whereas their maximum dimensionality used is limited to 20. Without proof but by qualitative arguments that are out of scope here, we would expect that SWAM networks perform similarly, but by a linear factor better than our summary-based approach. However, there are some advantages to our method. Joining the network is much cheaper in our scenario, and (for future applications more important) peers in SWAM know only a very small part of the network and do not have a general view on the data distribution in feature space.

Considering usage scenarios (at a level of detail similar to [18]) one realizes that given the conceptual differences, the diverse approaches perform surprisingly similar. However, there are two important advantages of summary-based methods. *Firstly,* peers participating in the network do not have to divulge their complete indexing data. This potentially lowers network joining cost and improves the publishers' privacy. *Secondly,* peers participating in the network have the occasion to know the characteristics of many other peers. This opens the way towards systems that go beyond query by visual example to interactive browsing, i.e. explorative search using overviews, such as [5].

5. CONCLUSION & OUTLOOK

We have presented a pragmatic approach to query routing in P2P networks that uses sample points for creating peer data summaries allowing for efficient similarity search. With high numbers of sample points and applying a compression algorithm we can produce compact summarizations of a peer's data in order to make good routing decisions. The approach we propose is robust w.r.t. the images we use for selecting the sample points.

In addition to *lower costs under churn*, our system shows *an increase in performance* w.r.t. to an implementation of a distributed indexing structure when it comes to *growing networks* (with more documents and more peers) and when the *dimensionality* of the feature is *increased*. We also showed *an increase in performance* when comparing our system to a probabilistic approach for P2P-CBIR using GMM representations of peers. The overall performance of the approach presented in this paper is extremely promising (top-20 can be found on average when contacting approximately 4% of the peers) with potential for further improvements by increasing the value of k, i.e. the number of sample points.

One interesting result of our work is that large subsets of flickr are very similar, to the extent that sample points derived from one subset describe well another disjoint collection. It would be interesting to undertake measurement studies if flickr samples also describe well other community consumer photo collections, eventually characterizing what makes such collections special.

Obviously a direction of future research is investigating the use of our summaries for more/other features and media types such as audio or video.

Another line of research is investigating the impact of our methods on the *usability* of P2P-CBIR systems, an often neglected part of P2P-CBIR and P2P-IR. We already have developed a prototype using our summaries. We are extending this prototype into the direction of a summary-based super-peer network in order to apply our scenario under real-world conditions in large networks. Details of this prototype are the topic of a future publication. The next step is to put to use the *collection overview* provided by our summary-based approach in order to bring distributed image browsing with relevance feedback to P2P networks.

6. REFERENCES

[1] F. Banaei-Kashani and C. Shahabi. SWAM: a family of access methods for similarity-search in peer-to-peer data networks. *CIKM*, pages 304–313, 2004.

[2] M. Bender et al. The Minerva Project: Database Selection in the Context of P2P Search. In *BTW, Karlsruhe*, pages 125–144, 2005.

[3] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *CACM*, 13(7):422–426, 1970.

[4] W. Chang, G. Sheikholeslami, A. Zhang, and T. F. Syeda-Mahmood. Efficient resource selection in distributed visual information systems. *ACM Multimedia, Seattle, Washington, USA*, pages 203–213, 1997.

[5] I. Cox, M. Miller, T. Minka, T. Papathornas, and P. Yianilos. The Bayesian Image Retrieval System, PicHunter: Theory, Implementation, and Psychophysical Experiments. *IEEE TIP*, 9(1):20–37, 2000.

[6] F. M. Cuenca-Acuna and T. Nguyen. Text-Based Content Search and Retrieval in ad hoc P2P Communities. Technical Report DCS-TR-483, Dept. for CS, Rutgers University, 2002.

[7] M. Eisenhardt, W. Müller, A. Henrich, D. Blank, and S. El Allali. Clustering-based Source Selection for Efficient Multimedia Retrieval in Peer-to-Peer Networks. *ISM, San Diego, CA*, pages 823–830, 2006.

[8] S. El Allali, D. Blank, M. Eisenhardt, A. Henrich, and W. Müller. Untersuchung des Einflusses verschiedener Bild-Features und Distanzmaße im inhaltsbasierten P2P Information Retrieval. *BTW 2007, 12th GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web*, pages 382–396, 2007.

[9] S. El Allali, D. Blank, W. Müller, and A. Henrich. Image data source selection using Gaussian Mixture Models. *5th International Workshop on Adaptive Multimedia Retrieval AMR'07*, 2007.

[10] P. Ganesan, B. Yang, and H. Garcia-Molina. One torus to rule them all: multi-dimensional queries in P2P systems. In *WebDB*, pages 19–24, 2004.

[11] P. Garbacki, A. Iosup, D. Epema, and M. van Steen. 2Fast: Collaborative Downloads in P2P Networks. *P2P2006*, pages 23–30, 2006.

[12] Gnutella. URL: `http://www.gnutella.com`.

[13] A. Henrich and H.-W. Six. How to split buckets in spatial data structures . *Int. Conf. on Geographic Database Management Systems*, pages 212–244, 1991.

[14] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM TODS*, 24(2):265–318, 1999.

[15] I. King, C. H. Ng, and K. C. Sia. Distributed content-based visual information retrieval system on peer-to-peer networks. *ACM TOIS*, 22(3):477–501, 2004.

[16] J. Li and J. Z. Wang. Automatic linguistic indexing of pictures by a statistical modeling approach. *IEEE TPAMI*, 25(9):1075–1088, 2003.

[17] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM*, pages 199–206, New York, NY, 2003.

[18] W. Müller, P. O. Boykin, N. Sarshar, and V. P. Roychowdhury. Comparison of image similarity queries in P2P systems. In *P2P2006*, pages 98–105, 2006. http://arxiv.org/abs/cs.DC/0606122.

[19] W. Müller, M. Eisenhardt, and A. Henrich. Scalable summary based retrieval in P2P networks. In *CIKM*, pages 586–593, 2005.

[20] W. Müller and A. Henrich. Fast retrieval of high-dimensional feature vectors in P2P networks using compact peer data summaries. In *MIR*, pages 79–86, New York, NY, USA, 2003.

[21] S. Ratnasamy et al. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[22] O. D. Sahin et al. PRISM: indexing multi-dimensional data in P2P networks using reference vectors. In *ACM Multimedia*, pages 946–955, 2005.

[23] N. Sarshar, P. O. Boykin, and V. P. Roychowdhury. Percolation search in power law networks: making unstructured peer-to-peer networks scalable. In *P2P2004*, pages 2–9, 2004.

[24] H. Shen, Y. Shu, and B. Yu. Efficient Semantic-Based Content Search in P2P Network. *IEEE TKDE*, 16(7):813–826, 2004.

[25] C. Tang, Z. Xu, and M. Mahalingam. pSearch: Information Retrieval in Structured Overlays. In *1st Workshop on Hot Topics in Networks*, pages 89–94, Princeton, NJ, 2002.

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

[27] R. C. Veltkamp and M. Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Department of Computing Science, Utrecht University, 2002.

[28] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-Sensitive Integrated Matching for Picture LIbraries. *IEEE TPAMI*, 23(9):947–963, 2001.

[29] B. Yang and H. Garcia-Molina. Designing a Super-peer Network. In *17th International Conference on Data Engineering, ICDE*, pages 49–60, 2003.