Clustering-based Source Selection for Efficient Image Retrieval in Peer-to-Peer Networks

Martin Eisenhardt Wolfgang Müller Andreas Henrich Daniel Blank Soufyane El Allali
{martin.eisenhardt,wolfgang.mueller,andreas.henrich,daniel.blank,soufyane.el-allali}@wiai.uni-bamberg.de
Media Informatics, Applied Computer Science, University of Bamberg, Germany

Abstract

In peer-to-peer (P2P) networks, computers with equal rights form a logical (overlay) network in order to provide a common service that lies beyond the capacity of every single participant. Efficient similarity search is generally recognized as a frontier in research about P2P systems. One way to address it is using data source selection based approaches where peers summarize the data they contribute to the network, generating typically one summary per peer. When processing queries, these summaries are used to choose the peers (data sources) that are most likely to contribute to the query result. Only those data sources are contacted. There are two main contributions of this paper. We extend earlier work, adding a data source selection method for high-dimensional vector data, comparing different peer ranking schemes. More importantly, we present a method that uses progressive stepwise data exchange between peers to better each peer's summary and therefore improve the system's performance.[1]

1 Introduction

Peer-to-peer (P2P) systems have shown their viability in a number of large-scale applications. Their success as means of large-scale data distribution in so-called file-sharing networks has motivated research in P2P data management. One focus of research is therefore the search on stored data in a P2P network. Current research is spanning diverse areas such as efficient exact search (via distributed hash tables [11, 10]), similarity search on text and multimedia data [12, 7], and semantic networks expressed via RDF [9].

The topic of this paper is similarity search for Information Retrieval (IR) and in particular Content Based Image Retrieval (CBIR), a technique that allows searching images by their visual content. The basic approach followed by our algorithms is that of *source selection*: As each peer holds data, each peer is a potential source of interesting data. At query time, peers that are most probably a source of interesting data are selected. This selection is done based on *peer data summaries* that are maintained by the P2P network. The selected peers are queried, then their results are merged. This general approach that stems from classical Distributed IR [3] is an alternative to many mainstream P2P approaches that use distributed indexing structures or the ones that try to improve the link structure in order to gain performance. We feel that the Distributed IR approach is not only promising in terms of performance, but also in terms of costs that have to be paid for maintaining such a network. This new work is based on our previously published research about peer summaries for CBIR [8, 6]. The paper adds more performant peer selection methods, and *most importantly a strategy to progressively change the data distribution within the network such that the peers focus more concisely on areas of interest*, thus enabling dramatically improved source selection.

The remainder of this paper is organized as follows: In section 2 we present the source selection approach together with a description of the data source (peer) summaries. Section 3 explains our experimental setup and data acquisition, the performance measures, the source selection strategies, as well as our mechanism to redistribute data for improving performance. Our experimental results are shown in section 4. Section 5 finalizes our paper with a conclusion and an outlook on future work.

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

2 A Summary-Based Network for Single-Hop Semantic Routing

A broad state of the art of P2P-IR is given in [8] and [7]. In this paper we are looking for an approach that permits efficient indexing of documents without expensive replication of full indexing data. As indexing of high-dimensional feature vectors (representing di erent features of an image, such as color distribution, texture, or shape) is hard, we choose a *single-hop semantic routing approach*, also known as *data source selection*. In this case, the bulk of the indexing data remains where the indexed data resides. Only short summaries are distributed within the network. Queries are routed to peers potentially containing interesting data. Each peer contacted 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 summary* and the *peer selection* method. Successful uses of summaries have been described for text data [4, 2] and image data [7]. Within this paper we present further improvements. We will strive to use summaries that are i) simple to generate, ii) cheap to distribute and iii) successfully permit efficient source selection.

While our focus is rather on summary and selection quality, we give a short overview over the P2P architecture that is utilized for putting the summaries to use.

2.1 The P2P environment

Our considerations are based on PlanetP [4]. In PlanetP each peer knows the summary of every other single peer participating in the network. This makes routing simple and the network extremely robust. As a downside, this approach is not scalable. Depending on the churn rate (*i.e.* the number of peers arriving 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 any more. However, Rumorama [7], 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 ranking of peers and the effect of clustering in PlanetP like middle sized networks, we can easily extend this to large scale Rumorama like networks. The original PlanetP implementation is designed for text documents. Within this paper, we use cluster-based summaries for image data as described below.

For the following it is important that the summaries are disseminated by randomized rumor spreading. A popular model for describing rumor spreading is the *random phone call model* [5]. The idea is that at fixed intervals or *rounds* each participant randomly chooses another participant in the network and synchronizes with this other participant in a human like rumor spreading by phone. Eventually, after a logarithmic number of rounds, all peers in the network are synchronized. While the exact mechanism is well described in [4, 5, 7], it remains important for the rest of the paper, that peers are periodically in contact with each other in order to keep the network alive.

2.2 Summaries for efficient source selection

Each peer's data is summarized via a *cluster histogram*. The cluster histogram of a peer is a vector. Each component of the vector corresponds to one cluster c_i. The value of each component of the histogram denotes how many documents of the peer belong to a certain cluster. The global clustering of the data needed for this approach can be efficiently performed in a distributed fashion. An overview of the method along with a Bayesian justification can be found in [8]. Let us quickly review the properties of the summaries obtained.

Simple to generate: The base of a cluster histogram is a global clustering of documents. In [6] we have described how to efficiently obtain a suitable clustering via a distributed k–means implementation. Furthermore, in this paper we show that a random selection of the cluster centroids has little influence on the overall retrieval performance, we describe when clustering is more useful than a random selection of centroids.

Cheap to distribute: On entering the P2P network, each peer needs to know the centroids of the global clusters. After that, only cluster histograms need to be exchanged.

Selective: As our results given in the following sections show, the summaries can be a good basis for selecting the most promising peers. How to rank peers according to the information given by the summaries is a main focus of this paper, described in section 3.3.

2.3 Retrieval using compact peer descriptions

Retrieval in a P2P network with semantic routing is comparatively easy. First of all, a user states a query (in our case a query document) locally. The corresponding peer can then easily determine which cluster the query belongs to, *i.e.* which cluster centroid is closest to the query, because the peer knows the cluster centroids. Since the peer also contains the compact representations of all peers in the network, it can therefore determine which peers are most likely to contain documents relevant to the query and rank the peers by their likelihood to hold relevant documents. The rank of a peer in this ordered list is called *peer rank*. The querying peer will afterwards contact these peers in ranked order up to a certain point and obtain lists of relevant documents he can merge.

3 Experimental Setup

In the following we describe the data that our experiments are based on as well as the measure that is used for evaluating retrieval performance. Afterwards, we propose several rankers for performing the ranking task. To improve the retrieval performance we also attempt to transfer some indexing data amongst peers, resulting in a more concise data distribution.

3.1 Data acquisition

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 might store an arbitrary number of photographs and other pictures in his/her account. We take a randomly chosen subset of 2,623 user accounts so that the total number of images in these accounts is 50,000. We assign each user account to one peer to simulate Flickr.com in a P2P setting. The feature set used to index the images is a 166-bin color histogram as used in [8]. The HSV color space is quantized into 18 intervals in the *hue* dimension, into 3 intervals in the *saturation* dimension, and into 3 intervals in the *value* dimension. Four more bins are added to represent grey values. Every image is therefore represented as a single 166-dimensional real-valued vector.

3.2 Defining a performance measure

The main performance measure used in our experiments –unless stated otherwise– is the *Average Peer Rank* $APR_N(n)$. It describes how many peers have to be contacted on average to find at least n of the top-N matches for a query. The top-N 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. 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 cost factors in our P2P information retrieval scheme. In order to give an example for the Average Peer Rank, $APR_{20}(5) = 13.37$ indicates the number of peers to be contacted on average to retrieve at least 5 of the top-20 documents for a given query. In our experiments the performance measure is averaged over 100 queries in order to minimize the influence of outliers on the results. We also calculate the APR as a percentage of the total number of peers in the network (2,623 in our case) to make results more comparable. In our experiments we always search for the top-20 documents matching a certain query.

3.3 Ranking the peers

Peers are ranked based on their summaries and the global cluster centroids. This represents the available knowledge that every peer has access to. Based on this, three different ranking mechanisms are implemented.

The *SimpleRanker* ranks peers according to the absolute number of documents they have in the query cluster, *i.e.* the cluster with the centroid closest to the query. If peer contributes for example three documents to the query cluster, whereas peer only has two documents in this same cluster, peer obtains a higher peer rank than peer . Otherwise, if both peers and contribute the same number of documents to the query cluster, they are assigned the same rank. During query processing the peers are contacted in decreasing order based on their ranks.

The *StableSortRanker* makes its decision based on $L_{clusters}$, a list which contains all global cluster centroids sorted in ascending order by their distance to the query. The first element in this list therefore always corresponds to the query cluster. If peer and peer have different amounts of documents in the query cluster, this ranking strategy behaves exactly in the same way as the *SimpleRanker*. When this is not the case and peer and peer both have the same number of documents in the query cluster, *StableSortRanker* chooses the next closest cluster out of $L_{clusters}$ and compares peer and peer according to the number of documents in this second cluster. The *StableSortRanker* is based on the assumption that whenever a

i	1	2	3	4	5	6	7
n_i	11.71	3.66	1.68	0.93	0.56	0.36	0.25

Table 1. Top-20 documents n_i contained in the seven closest clusters c_i to the query.

relevant document is not in the query cluster, then it might be in the next closest cluster according to the order of $L_{clusters}$.

The *DistanceWeightingRanker* calculates weights $w_i = 1 - \frac{d_i}{d_{max}}$ for every cluster, where d_i is the distance from cluster c_i's centroid to the query and d_{max} denotes the maximum distance of all the cluster centroids to this query. Then, the ranking is obtained by comparing the sum: $s(\ ) = \sum_{i=1}^{k} w_i \times m_i(\ )$ where k is the number of clusters and $m_i(\ )$ is the number of documents that a peer contributes to the cluster c_i. Based on the value $s(\ )$ obtained, the decision if peer is ranked higher than peer is made. If for example $s(\ ) > s(\ )$, peer is therefore ranked higher.

Two of our rankers use distance information for ranking the peers. As a distance metric we are currently using the Euclidean distance in all our experiments, as is often done in the indexing literature ([8] contains references). Some authors (*e.g.* [1]) propose more meaningful distance measures for high dimensional space. This will be one direction for future work. Table 1 shows, on average, the total number of the top-20 documents n_i contained in a cluster c_i. For example the second cluster c_2 for which the centroid is the second closest (according to Euclidean distance) to the query contains on average 3.66 of the top-20 documents. With increasing i the number of documents in c_i decreases monotonically. The first seven closest clusters are shown in Table 1. Experiments on our data show that on average 11.7 of the top-20 documents lie within the closest cluster, and 19.15 of the top-20 documents lie within the seven closest clusters to the query.

3.4 Swapping indexing data between peers

So far indexing data (*i.e.* the feature vector of an image) and indexed data (*i.e.* the image itself) reside on the same peer. In order to identify similar feature vectors matching a given query, a large fraction of peers is therefore likely to be contacted, since the images stored by a peer might be very heterogeneous. By swapping indexing data between peers, so that a peer is only responsible for indexing a more or less homogeneous part of the documents, we can reduce the number of peers contacted for a given query. The goal is therefore to reduce the heterogeneity within the indexing data of a particular peer by making another peer responsible for administering this indexing data. In the following we describe a mechanism that performs the index swapping task.

To perform the swapping of indexing data, peer first needs to identify the tuples (*i.e.* feature vectors) that fit worst to its document collection. A simple way of doing this based on the global clustering is by identifying the least populated clusters. Let us assume that t is an index tuple belonging to a cluster c_i for which swapping is beneficial. Therefore peer needs to identify a ranking of the other peers, for example based on the number of tuples that lie in cluster c_i (peers with many documents in c_i are promising new hosts for t). This can be achieved by using *SimpleRanker* and t as a query. Then, peer iterates over this ranking of peers until one of the two following conditions is met. Either peer is able to successfully transfer the candidate tuple t to the other peer as peer has more documents in cluster c_i, or that peer owns a smaller number of documents in c_i than peer . Hence, an exchange of tuples would not be beneficial. With respect to the performance of the overall network, peers are allowed to limit the number of tuples they are willing to handle to keep the network free of hot-spots.

Index swapping combines the advantages of approaches such as [12] (where specialization is immediate but all indexing cost is paid up-front) and summary-based approaches. When peers stay in the network for a long period of time, peers specialize on regions of keyspace. On the other hand, if a peer only stays for a short period of time, the cost it incurs to the network is small.

4 Experimental Results

In the following we evaluate the different rankers and their performance. Additionally we look at the benefit of swapping indexing data between peers.

4.1 The performance of the peer rankers

Obviously one important parameter for our approach is the number k of clusters used in the distributed k–means clustering and therefore also in the peer data summaries. Higher values for k increase the storage requirements for the summaries but they seem to promise more fine-grained source selection decisions.

Figure 1 shows the performance of the different rankers described in section 3.3 with respect to the number k of clusters used when retrieving at least 10

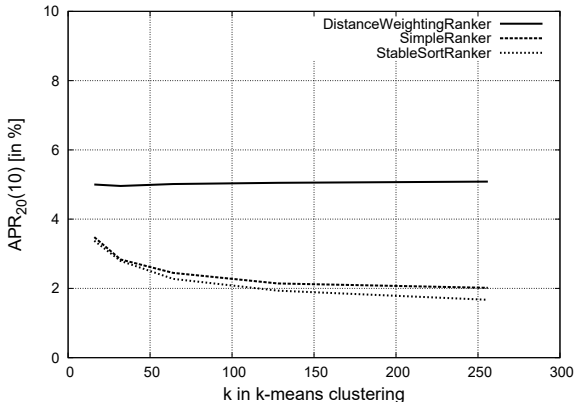

Figure 1. Performance of rankers with respect to the number of clusters k using k–means.

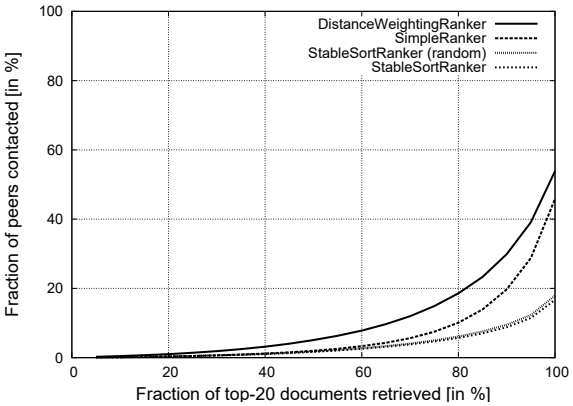

Figure 2. Performance of the rankers w.r.t. the fraction of peers contacted.

out of the top-20 documents. As can be seen in the figure, *StableSortRanker* and *SimpleRanker* outperform *DistanceWeightingRanker* independent of k. The number of clusters k is not impacting the retrieval performance dramatically for a large enough chosen k, since the $APR_{20}(10)$ is almost constant for $k > 100$.

We also compare the rankers with respect to the fraction of top-20 documents retrieved. Figure 2 (please ignore "StableSortRanker (random)" for the moment) shows that *StableSortRanker* contacts a far smaller fraction of peers to retrieve the same fraction of top-20 documents than the other rankers, if the fraction of the top-20 documents retrieved is relatively high. Notice that the performance of *StableSortRanker* and *SimpleRanker* only diverges if on average more than 55% out of the top-20 documents are retrieved. This corresponds to the average number of documents inside the query cluster ($n_1 = 11.71$) in Table 1.

It is also interesting to see the performance of the ranking when we use a random clustering, which is done simply by selecting a random number of cluster centroids without further refinement. We compare k–means clustering and the random selection of cluster centroids with respect to the number of clusters chosen. As shown in Figure 3, k–means clustering brings some little improvement. The "StableSortRanker (random)" curve in Figure 2 confirms these findings.

Back-of-the-envelope calculations based on 1 query per peer per day and query cost reduced by 8% when using clustering suggests that clustering pays off if the size of the network is big enough. For instance, when the network size is less than 1000 peers a break-even, at which clustering starts to pay off, is after a prohibitively long ≈500 days. On the other hand for bigger

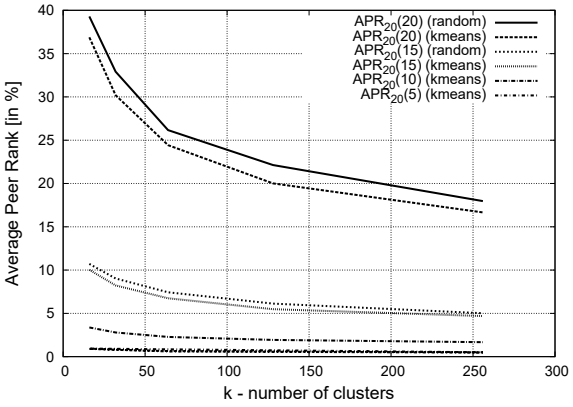

Figure 3. Performance of *StableSortRanker* w.r.t. the number of clusters k in k–means and random clusters.

network sizes of more than 1 million peers the break-even is after 0.5 days.

4.2 The effect of swapping index data

In section 3.4 we have presented an algorithm to swap parts of the indexing data from one peer to another. This is done to increase the homogeneity of indexing data within one peer so that the costs for a querying peer are reduced. When applied to our experiments, index swapping yields major improvements in retrieval performance. Since index data swapping is an iterative process, we track the progress in discrete *swapsteps*, a simplification made to enable efficient measurements. In a single swapstep, each peer in the network may exchange at most one tuple with another peer. It may happen that a peer does not swap

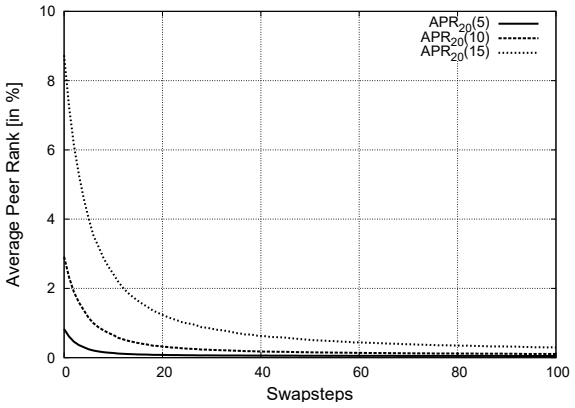

Figure 4. The effect of index swapping.

any tuple and the number of peers doing so increases over time. Figure 4 depicts the increase in retrieval performance when index swapping is applied to our P2P network. The background of these results is that the data of a peer in our network is not homogeneous but in some way "pre-clustered" around some themes (for example a hike through a mountain range) with some outliers. These outliers are expunged early so that the remaining tuples on a peer become more homogeneous.

We can conclude that index data swapping results in major improvements in retrieval performance, with moderate communication costs that decrease over time. It is possible to piggy-back the necessary data on messages that would have been sent anyway between two peers as part of the network administration. Index swapping makes our approach even more adequate for summary-based image retrieval in P2P networks.

5 Conclusion

We have presented an approach to clustering-based semantic routing in P2P networks, allowing for efficient operation and retrieval. Furthermore, we have experimentally shown that k–means clustering is better than randomly selecting cluster centroids. We have presented the scenarios under which k–means clustering is more useful than randomly selecting the cluster centroids. Also we considered di erent rankers and illustrated that *StableSortRanker* gives very good results for source selection.

For future work, we consider investigating the performance of our system using other clustering algorithms and other distance measures, as well as studying the effect of dimensionality reduction on query efficiency and result quality. It is worth noting that, since in PlanetP like networks communication costs for rumor spreading rise linearly with network size, the network size becomes an efficiency limit for larger scale PlanetP networks. Therefore, to scale beyond this point we use a hierarchical structuring technique called Rumorama [7] to organize small sub-networks in a tree-like structure. We see the approach presented here suitable for "micro-routing" within small networks while inter-network routing ("macro-routing") is provided by Rumorama.

References

[1] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973:420–434, 2001.

[2] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: collaborative p2p search. In *VLDB '05*, 2005.

[3] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *SIGIR'95*.

[4] 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 Comp. Science, Rutgers University, 2002.

[5] R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *FOCS '00*, Washington, DC, USA, 2000.

[6] W. Müller, M. Eisenhardt, and A. Henrich. Efficient content-based P2P image retrieval using peer content descriptions. In *Internet Imaging V. Proceedings of the SPIE, Vol. 5304.*, 2003.

[7] W. Müller, M. Eisenhardt, and A. Henrich. Scalable summary based retrieval in p2p networks. In *CIKM '05*, New York, NY, USA, 2005.

[8] W. Müller and A. Henrich. Fast retrieval of high-dimensional feature vectors in p2p networks using compact peer data summaries. In *MIR '03: Proc. of the 5th ACM SIGMM intl. workshop on Multimedia information retrieval*, pages 79–86, New York, NY, USA, 2003. ACM Press.

[9] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M. Schlosser, I. Brunkhorst, and A. Löser. Super-peer-based routing and clustering strategies for rdf-based peer-to-peer networks. In *WWW'03*, 2003.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *SIGCOMM '01*, San Diego, CA, USA, 2001.

[11] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *SIGCOMM'01*, San Diego, CA, USA.

[12] 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.