Improving Information Retrieval Effectiveness in Peer-to-Peer Networks through Query Piggybacking

Emanuele Di Buccio, Ivano Masiero, and Massimo Melucci

Department of Information Engineering, University of Padua, Italy
{dibuccio,masieroi,melo}@dei.unipd.it

Abstract. This work describes an algorithm which aims at increasing the quantity of relevant documents retrieved from a Peer-To-Peer (P2P) network. The algorithm is based on a statistical model used for ranking documents, peers and ultra-peers, and on a "piggybacking" technique performed when the query is routed across the network. The algorithm "amplifies" the statistical information about the neighborhood stored in each ultra-peer. The preliminary experiments provided encouraging results as the quantity of relevant documents retrieved through the network almost doubles once query piggybacking is exploited.

1 Introduction

One of the potentials of P2P networks when adopted as "a federated search layer for digital libraries" [1] is to allow each Digital Library (DL) node, namely each peer, to contribute to the document collections by pushing their own documents, yet without delivering the full content. Although P2P is a promising paradigm, it poses new daunting challenges: the main one is that the content to be searched is anarchically distributed across large networks of not necessarily cooperative peers. Such a little cooperation implies that each peer has got a little knowledge about the content of the other peers and, more specifically, it has got no knowledge about most peers. A suitable solution is unstructured networks built on top of different indexes which summarize groups of peers and not only the collections of the peers [1,3]. In particular, the P2P networks considered in this work are unstructured (i.e. no DHT-like data structures), hybrid (i.e. the simultaneous presence of peers and ultra-peers) and hierarchical (i.e. each peer refers to one and only one ultra-peer which serves a group of peers acting as a hub/router for queries sent by a user in its group). Figure 1 depicts an instance of this kind of P2P networks, where rounds represent ultra-peers and rhombi represent peers.

At searching time, ranking ultra-peers is necessary for selecting those groups to which peers storing most relevant documents belong. The specific algorithm adopted in this paper is the one proposed in [3] and implemented in the SPINA software architecture [2]; such algorithm is based on a statistical model used for ranking documents, peers and ultra-peers. In SPINA each ultra-peer maintains the information to rank its neighbors in its *ultra-peer index* – in this index, the

M. Agosti et al. (Eds.): ECDL 2009, LNCS 5714, pp. 420–424, 2009.

elements in the posting list associated to each feature are the identifier of the neighbor and the weight of the feature in that neighbor. Because of its content, the ultra-peer index has "ultra-peer granularity". The set of *neighboring ultra-peers* or *neighbors* of an ultra-peer X are the ultra-peers whose aggregate statistics are maintained in the ultra-peer index of X. In Figure 1, an arc connecting X and Y denotes that Y is a neighbor of X. Let suppose a query is sent to ultra-peer E. At each hop, the query is forwarded to the m top-ranked neighbors. After two hops, ultra-peers H, I, L are possibly contacted (gray colored area) during the query routing. In this scenario, starting from E, if the ultra-peer A stores relevant documents, these will not be retrieved and recall decreases.

This paper describes an algorithm which aims at improving recall in similar scenarios. The algorithm is based on the framework proposed in [3] together with a "piggybacking" technique performed at query routing time. Although the use of piggybacking is a well known technique in computer networks for managing the messages carried at the lowest levels of the network protocols, to our knowledge, no research work reported the use of piggybacking for Information Retrieval (IR) across P2P networks at *retrieval level* in the way reported in this paper.

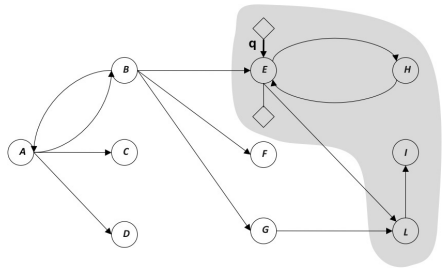

Fig. 1. Instance of unstructured hierarchical hybrid P2P network

2 Improving Recall through Query Piggybacking

An ultra-peer Y is a *known ultra-peer* with respect to the ultra-peer X if X has just a partial knowledge of Y, that is, it stores a subset of the aggregate statistics of Y. This notion differs from the one of *neighboring ultra-peer* where a complete knowledge of the aggregate statistics about the ultra-peer is required. The aggregate statistics are stored in the Dynamic History Index (DHI) and refer to ultra-peers which are *not* in the neighborhood of X. Let suppose the user expresses his information need by submitting a query as a bag of features f_i's, e.g. keywords. At the beginning, the DHI of X is empty. When X receives the query, it selects some top-ranked neighbors. Then, the weight of each feature in the neighbors of X are *piggybacked* onto the message used to forward the query. The message is received by the neighbors which (i) forward the query to the top-ranked peers of the group governed by the ultra-peer and (ii) extract the piggybacked ranking information for each feature of the query. The features and the weights which do not refer to the neighbors are used to increment the DHI. Following this, each ultra-peer ranks the set of its *neighboring* and *known* ultra-peers searching across the ultra-peer index and the DHI as they were a single index. The process is then iterated for each contacted ultra-peer.

Table 1. Indexes and ranking steps in the ultra-peer A (Tab. 1a) and B (Tab. 1b); k_i is a feature in the indexes, and f_i is a feature of the submitted query

(a)

k_i	A	B	C	D
a	2	1	0	2
b	4	1	1	1
c	2	2	1	2
d	0	5	0	0
e	5	2	1	5

f_i	A	B	C	D
b	4	1	1	1
c	2	2	1	2
e	5	2	1	5
$\sum_{f_i}$	**11**	**5**	3	**8**

(b)

k_i	A	B	E	F	G
a	2	1	0	1	5
b	4	1	2	3	4
c	2	2	7	2	1
d	0	5	0	1	1
e	5	2	1	0	2

k_i	C	D
b	1	1
c	1	2
e	1	5

f_i	A	B	C	D	E	F	G
b	4	1	1	1	2	3	4
c	2	2	1	2	7	2	1
e	5	2	1	5	1	0	2
$\sum_{f_i}$	11	5	3	8	**10**	**5**	**7**

Consider, for example, the network depicted in Fig. 1 where $q' = \{b, c, e\}$ is the query submitted by a user (peer) in the group led by ultra-peer A. Let $m = 3$ be the number of top-ranked ultra-peers to which the query is routed by another ultra-peer. Let assume that the DHIs of A and B are empty at the beginning and A, B, C, D are the neighbors of A and A, B, E, F, G are the neighbors of B.

The left part of Table 1a represents the ultra-peer index of A listing the weights for each feature k_i in the index. When the query arrives at A, the ultra-peers in its neighborhood are ranked and the m top-ranked ones are selected to forward the query q'. The right part of Table 1a shows the selected top-scores (shaded in grey). Furthermore, A sends the query to the top-ranked ultra-peers, adding the ranking information about each feature, including its own scores. Ultra-peer B receives the query, forwards q' to the top-ranked peers of its group, extracts the weights for each feature of q' and increments its DHI using the statistics about known ultra-peers (not in the set of neighbors of B). Afterward, it iterates the ranking process by accessing its ultra-peer index and DHI, which act as a single index. As for B, Table 1b represents (from left): the index terms k_i's and their weights in the ultra-peer granularity index, the updated contents of the DHI and, finally, the results of the ranking step. B sends the query to the top-ranked ultra-peers (in this case E, F, G), adding to query message all the ranking information about each feature of q'. Note that ultra-peers A, B, D are not selected because they have already processed the query. A mechanism to avoid a re-forwarding of the query to the same receivers can be easily implemented.

Fig. 2 depicts the changes in the overlay network after two hops w.r.t. ultra-peer E. The dashed lines point out the increment of the DHI's size due to the propagation of aggregate statistics for each feature of the query. For example, during the last query's hop to ultra-peer E, high granularity information on five new ultra-peers – namely A, B, C, D, F, G – are added to the DHI of E, with regard to the features b, c and e. This example shows how the "horizon" of

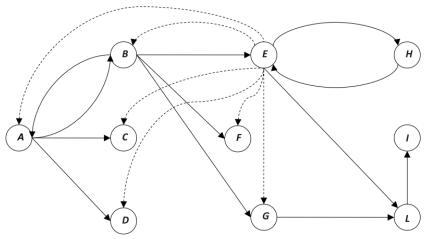

Fig. 2. Changes in the topology after two hops w.r.t. ultra-peer E

the ultra-peer E can be widened by the query piggybacking technique. If a new query with the features b and e starts from the ultra-peer E, the most promising ultra-peer A can be reached in one hop by exploiting the information stored in the DHI.

3 Experiments

The experiments were based on a realistic P2P network: the 1,421,088 documents distributed across 2,500 collections of the DLLC (Digital Libraries Lu and Callan) collection were distributed on an actual, real peer network. The functionalities for indexing, retrieval and communication among peers were provided by SPINA [2]. The connections between the ultra-peers were generated randomly. Each ultra-peer had no less than 3 and no more than 5 neighbors.

Some preliminary experiments run on the TREC topics numbered from 451 to 500 are reported in this paper. The starting ultra-peer was drawn at random and the draw was repeated ten times. Time-To-Live (TTL) was the number of times the query was routed to an ultra-peer, m was the number of top-ranked neighbors to which a query was routed by an ultra-peer, whereas k was the number of top-ranked peers to which a query was routed by the ultra-peer of the group, and n was the number of top-ranked documents retrieved from each selected peer. Table 2 reports the average number of relevant retrieved documents per starting ultra-peer for two runs when TTL = 2, $m = 5$, $k = 1$, $n = 50$. Run 1 did not make use of the query piggybacking technique starting from an empty DHI — a DHI for each ultra-peer was created —, Run 2 exploited the data stored in the DHI during Run 1. The obtained results suggest that the query piggybacking technique has a positive effect over the resource selection step.

Table 2. Number of relevant retrieved documents per starting ultra-peer for two runs

Starting ultra-peer	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*L*
Run (1)	60	57	55	66	74	62	54	72	59	64
Run (2)	135	111	84	94	154	133	103	125	128	112

4 Concluding Remarks

This paper has illustrated an algorithm based on a query piggybacking technique and some preliminary experimental results using a reference test collection. The obtained results are encouraging and suggest an extensive evaluation of the proposed technique. The impact of the query distribution and small connected network topologies on the effectiveness of the proposed algorithm will be matter of future investigation. To our knowledge, this infrastructure will be the first realistic peer network for large scale P2P-IR experiments.

References

1. Lu, J., Callan, J.: Full-text federated search of text-based digital libraries in peer-to-peer networks. Information Retrieval 9(4), 477–498 (2006)
2. Di Buccio, E., Ferro, N., Melucci, M.: Content-based information retrieval in SPINA. In: Proceedings of IRCDL 2008, Padova, Italy, January 2008, pp. 89–92 (2008)
3. Melucci, M., Poggiani, A.: A study of a weighting scheme for information retrieval in hierarchical peer-to-peer networks. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 136–147. Springer, Heidelberg (2007)