Intelligent Peer Networks for Collaborative Web Search

Filippo Menczer, Le-Shin Wu, and Ruj Akavipat

■ *Collaborative query routing is a new paradigm for web search that treats both established search engines and other publicly available indexes as intelligent peer agents in a search network. The approach makes it transparent for anyone to build his or her own (micro) search engine by integrating established web search services, desktop search, and topical crawling techniques. The challenge in this model is that each of these agents must learn about its environment—the existence, knowledge, diversity, reliability, and trustworthiness of other agents—by analyzing the queries received from and results exchanged with these other agents. We present the 6S peer network, which uses machine-learning techniques to learn about the changing query environment. We show that simple reinforcement learning algorithms are sufficient to detect and exploit semantic locality in the network, resulting in efficient routing and high-quality search results. A prototype of 6S is available for public use and is intended to assist in the evaluation of different AI techniques employed by the networked agents.*

Centralized search engines cannot cover the entire web (Lawrence and Giles 1999) because it is too large, fast-growing and fast-changing (Brewington and Cybenko 2000; Fetterly et al. 2003; Ntoulas, Cho, and Olston 2004). As a result, current centralized search engines focus on "important" portions of the web. However, the notion of importance is highly subjective: the biases that are introduced to address the needs of the "average" user can result in diminished effectiveness in satisfying many atypical search needs. Therefore, the "one engine fits all" model cannot handle the increasing size, rate of change, and heterogeneity of the web and its users. In addition, as search becomes more prevalent at the desktop level, users will increasingly want to make subsets of the files indexed in their computers available to others through the Internet. Peer networks provide us with an architecture for extending web search technology to capture the contextual needs of a diverse population of users, while leveraging their resources.

There are several models of peer network topologies and query protocols, including structured, unstructured, flooding, distributed hash tables, and hierarchical (Androutsellis-Theotokis and Spinellis 2004). Our design of a collaborative web search network is guided by the principle of *semantic locality:* peers with shared interests are likely to communicate with each other more frequently than unrelated agents, so they should be able to reach each other in a few virtual hops. However, a dense network would generate too much traffic. A good topology favors both effectiveness and efficiency, by making it possible for a query to reach a relevant target peer in few steps, without imposing a large traffic load on the entire network. Small-world networks (Watts and Strogatz 1998) provide both clustered communities and enough randomness to keep the network distance small between any two peers. Effective search requires that the clusters be associated with a high semantic similarity between neighbors (Watts, Dodds, and Newman 2002). Because there is no global knowledge of the network (what peers are currently present, what information they hold, and what information they seek), and the network is very dynamic (peers may join and leave the network at any time), we cannot impose semantic locality into the network by design; instead, we explore AI techniques through which semantic locality will emerge as the result of local interactions and learning by individual peer agents.

Our research group is currently developing 6S, an intelligent multiagent application for peer-based web search (Wu, Akavipat, and Menczer 2005; Akavipat et al. 2006). The name is a contraction of "six degrees of separation" and "search," to reflect the social network of peer agents at the base of the collaborative search process. Each 6S peer agent is both a (limited) directory hub and a content provider; it has its own topical crawler (based on local context), which supports a

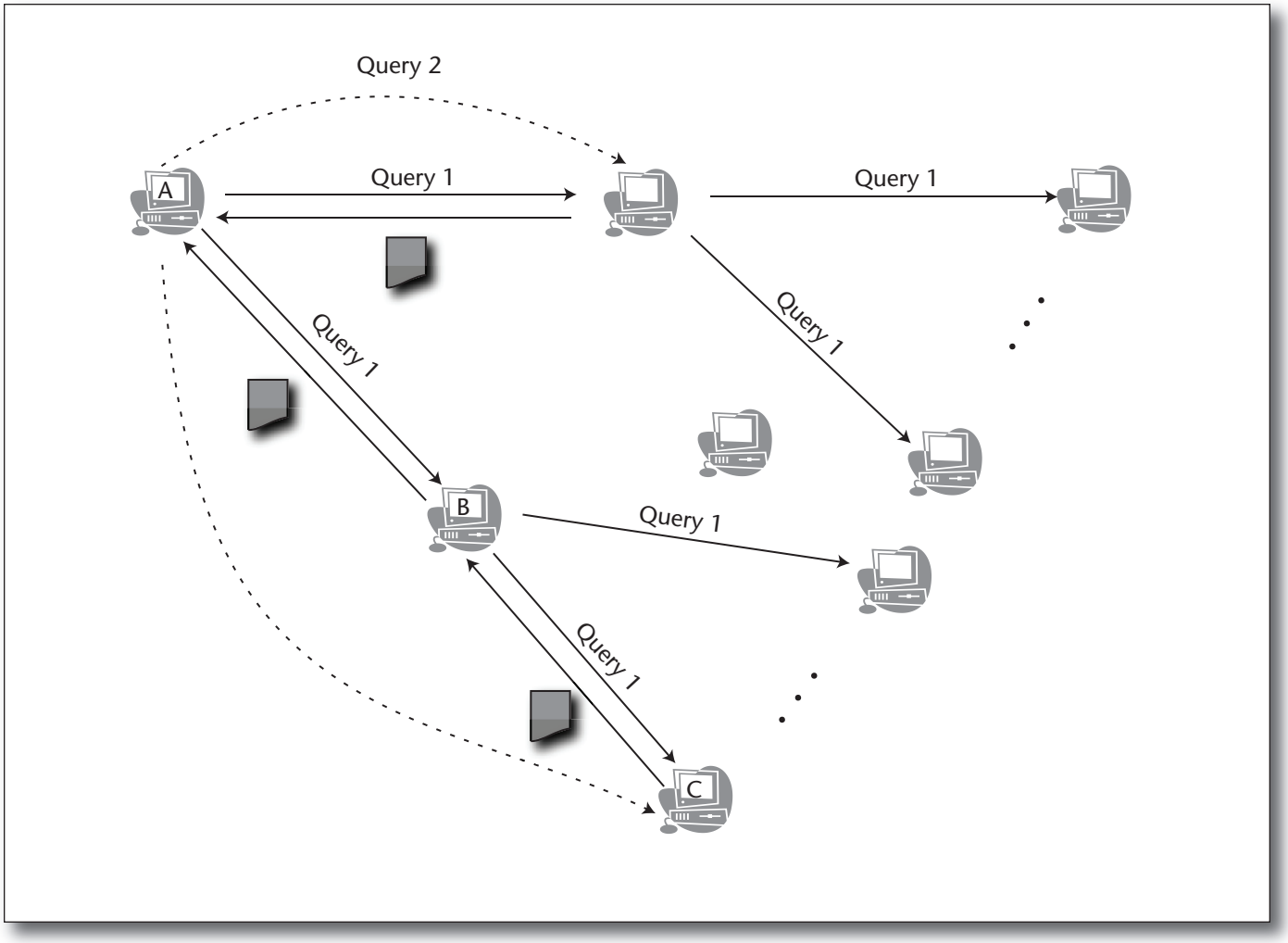


Figure 1. The 6S Search Mechanism and Peer Discovery.

The 6S application is designed not to have peers aggressively flooding the network for searching or discovering new peers. Therefore, the 6S peer only forwards its query to a small number of selected neighbors. A time-to-live mechanism ensures that a forwarded query will not survive in the network too long. Here Alice's agent A receives good results from agent C for Query 1. These results are forwarded through B. Later, A can send Query 2 directly to the newly discovered neighbor C.

local search engine—typically, but not necessarily, a small one. As shown in figure 1, queries are first matched against the local engine and then routed to neighbors to obtain more results. While receiving responses, an agent may discover new peers through its current neighbors. The new neighbor peers can later contact each other directly.

Figure 2 compares the collaborative search network framework with existing search models. Two major features we want to merge are *contextual learning* (as in intelligent web agents) and *social collaboration* (as in file-sharing peer networks). Intelligent web agents leverage local context from both the user and the information environment while learning to perform their tasks. Similarly, 6S agents use the local context captured from the user and from interactions with other peers as they learn to route queries to the most appropriate neighbors. The local user context of a 6S agent is a document collection created by the user.

With respect to social collaboration, 6S agents use a network to share information through queries and responses, as do nodes in a peer-to-peer (P2P) network. Without relying on a centralized resource collection, our search model emulates the information finding and spreading mechanisms in social networks. However, powerful central search engines such as Google and Yahoo can very well contribute to and profit from the social collaborative framework; indeed we expect that they would quickly turn into popular hubs thanks to their large collections and popularity-driven ranking algorithms. Therefore, our mod-

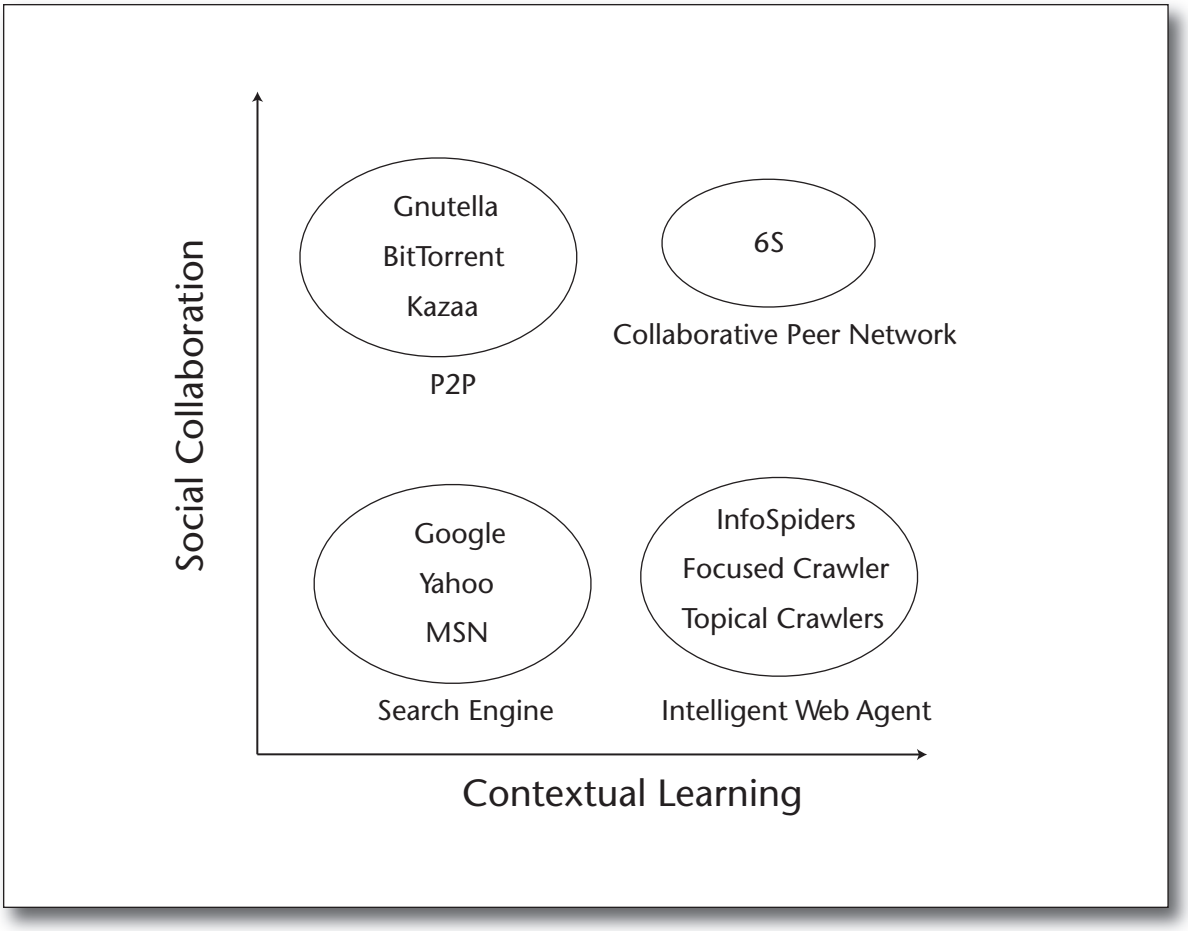


Figure 2. Two Dimensions of Search Systems.

The two dimensions of search systems are (1) the degree of social collaboration—as, for example, among networked agents—and (2) the degree of user contextual learning as in intelligent web agents such as InfoSpiders (Menczer and Belew 2000), focused crawlers (Chakrabarti, van den Berg, and Dom 1999), and other topical crawlers.

el of collaborative web search allows users to integrate both centralized and social search engines transparently. As in file-sharing networks, the incentive for people to collaborate is selfish—they can profit by participating in the network as they gain access to additional sources tailored to their needs.

Implementation and Deployment of 6S

The 6S application is designed to make it easy and transparent for users to index and share a collection of web pages, that is, to build a "micro search engine." A 6S *servent* (server + client) application integrates a topical crawler, a document-indexing system, a retrieval engine, a P2P network communication system, and a contextual learning system. In the current implementation, 6S relies on two open-source platforms: Nutch (nutch.org) for its search engine and JXTA (Waterhouse 2001) for the P2P network communication framework.

From the user's perspective, the main features of 6S are peer search, a personal web index management system, and a browser extension. The peer search functionality is an extension of local search. Local search is performed using the built-in search engine to provide users with relevant results from their local collections. Next, the application automatically selects neighbors that are best suited to answer the user's query based on the peer's prior query-response experience and sends the query to those peers. (Using the same mechanism, those neighbors forward the query to other peers, and so on; see figure 1.) Finally, the results obtained locally and from other peers' responses for the same query are combined to remove duplicates, reranked based on a simple voting algorithm, and then presented to the user. Results are updated dynamically as they arrive.

Behind the scenes, the application analyzes the

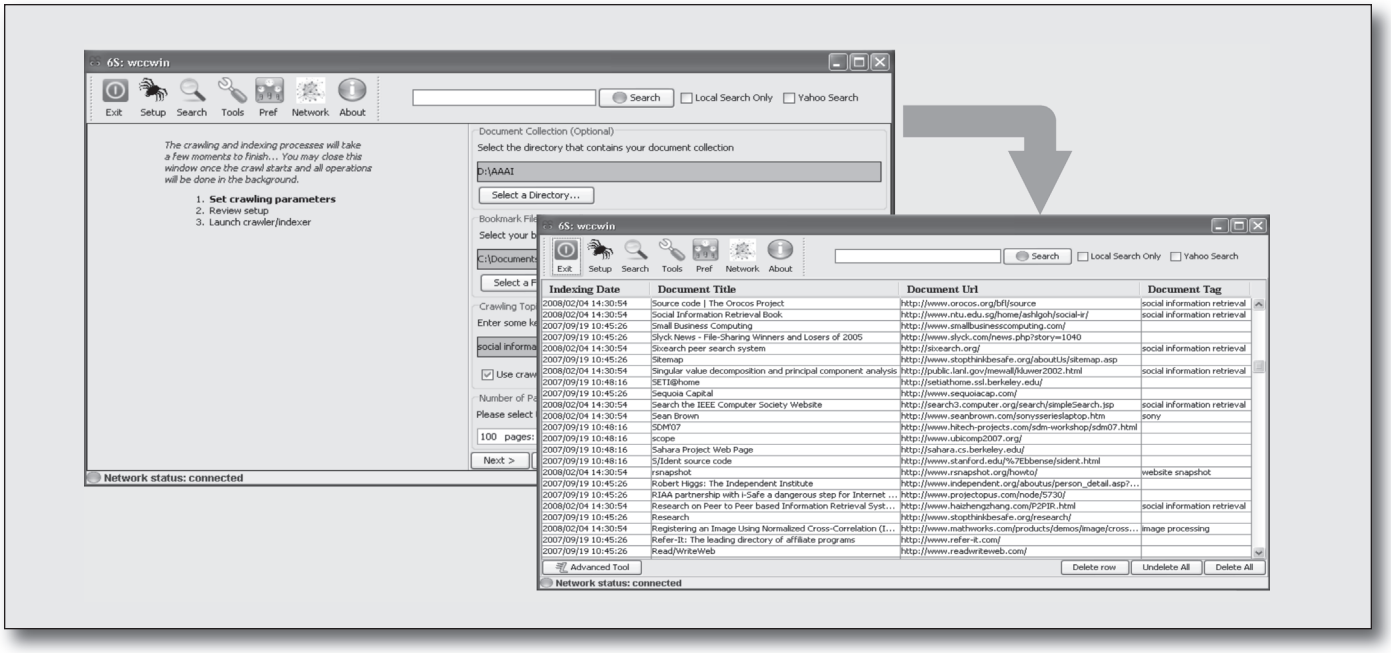


Figure 3. Setup of a 6S Peer.

To create a personal web index, the user may provide a crawling topic, a number of seed pages extracted from the user's bookmarks, or a local document collection. These cues are used to guide a topical crawler. The crawling results are then indexed for keyword searching. For each indexed document the user can assign or modify tags, which are searchable by the local engine. Users can also delete/undelete any document entries or remove/update the entire index.

results received from other peers, comparing them to the local search results, to learn a representation of the other peers. This representation is then used to improve the peer-selection algorithm, which is at the heart of the query-routing process. The details of the machine-learning algorithm used for this purpose are discussed below. The more a user employs the peer search network, the more she trains the system to better locate relevant information in the future.

The personal web index management system helps a user automatically create a web index. In fully automated (one-click) mode, the application selects pages from a local bookmark file and supplements them with results from a topical web crawler. Consider for example a user Alice. The application analyzes the queries in her web search history to construct a topic description, then launches the crawler. This process takes place the first time Alice sets up her peer, if she so chooses. Subsequently, 6S periodically updates the index with new additions from Alice's bookmarks or with a new topical web crawl based on her recent search history and current web index.

As shown in figure 3, the index management feature also allows users to manually create or add to the personal index or to launch crawlers with starting seeds and topics of choice. The current implementation employs a *best-N-first* topical crawler, which has been proven both efficient and effective for supporting a dynamic search engine among a number of crawling algorithms (Pant, Bradshaw, and Menczer 2003; Menczer, Pant, and Srinivasan 2004). Briefly, the crawler is given a set of topic keywords that are either entered by the user or extracted from the user's web search history and a number of seed pages that are obtained from the user's personal bookmarks and/or the local document collection. The URLs to be visited are prioritized by the similarity between the topic and the page in which a URL is encountered. Some additional mechanisms guarantee that the crawler is sufficiently exploratory. This crawler is publicly available (informatics.indiana.edu/fil/IS/JavaCrawlers). Once the index is built, the user can manage (tag, modify, delete, or recover) any indexed documents. For example Alice may index the documents in a review folder and provide a topic, "data mining," to guide the crawler. She can modify or tag the indexed documents as well.

An extension for the Firefox web browser enables convenient access to 6S while the application is active as a background process. As shown in figure 4, users can submit search queries to the local peer and the 6S network, see the results returned though the network, and instruct the application to index new pages—all from the browser. Search results are shown along with the usual information, as in any traditional search engine (title, snippet, and so on), as well as information about the peers that provided each result.

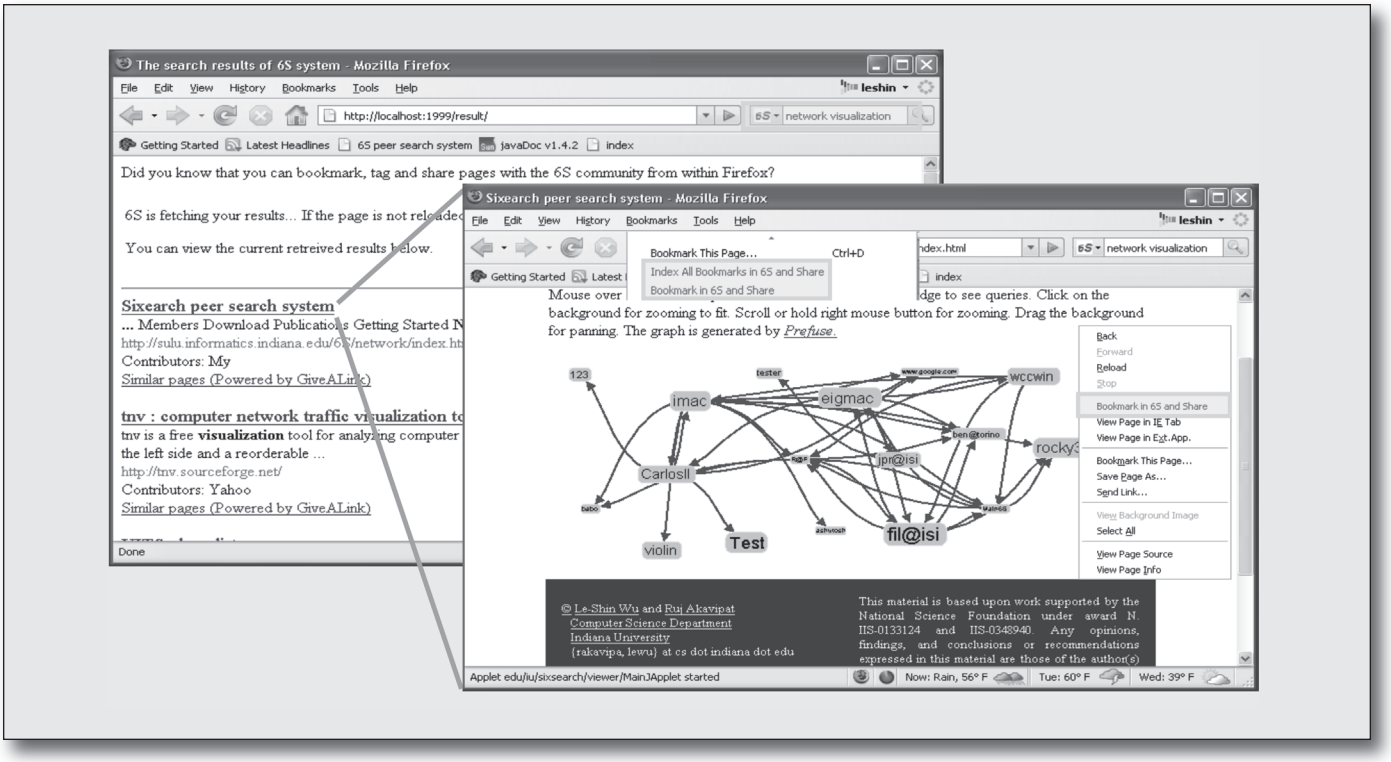


Figure 4. Accessing 6S Using a Firefox Extension.

With 6S running as a background process, a user can access 6S without leaving the web browser through the 6S extension for Firefox. It allows the user to search through the 6S community, export bookmarks to 6S, or index a single web page. All these operations can be done with only a few clicks.

To export pages to the local peer and share them with the 6S community, users can use the *Bookmarks* drop-down menu and select options to index all the bookmarks or just the current page. The latter option is also available in a contextual (right-click) menu. For example, upon receiving a relevant page from another 6S peer in response to a query about "social networks," Alice may choose to bookmark this page in her local 6S index, thus making it possible to share this page with other peers with related queries in the future.

Inside a 6S Agent

Each 6S agent uses a reinforcement learning algorithm to track the profiles of other peers based on their past interactions. A *neighbor profile* is the information that a particular agent maintains to estimate the neighbor's likelihood to provide relevant results for various keywords. For example, if a neighbor has previously provided good results for Alice's query "open source software," her agent should internalize this information so as to predict that this might be a good peer to forward a future query on "free software." By learning profile information, agents try to increase the probability of choosing appropriate neighbors for their queries.

Interactions with peers reveal information of varying reliability. We want to capture all available information in profiles but must discriminate cues on the basis of their reliability. To achieve this goal we let each peer maintain two profiles for focused and expanded information, respectively. The focused profile concerns only query terms, while the expanded profile includes keywords that co-occur frequently with query terms within hit pages. Each profile has the same structure and is represented as a matrix W, where each element $w_{p,k}$ is an estimate of how knowledgeable and reliable is peer p with respect to keyword k. When p returns results for a query containing k, $w_{p,k}$ is updated to reflect the quality of these results. The results from p are compared to local ones to obtain a reinforcement signal: good results induce a reward, by which $w_{p,k}$ is increased, while poor results induce a penalty and $w_{p,k}$ is decreased. The update occurs through a running average to slowly forget past performance while tracking new information. One of the main motivations behind this approach is that the learning context is likely to be extremely nonstationary, with highly dynamic peers' interests and collections. Details are illustrated in figure 5. Suppose for example that Alice submits the query "Lama," and that Peer 10 returns a set of hits with an average score of $S_{10} = 0.8$. Further suppose that the results from Alice's local index yield an

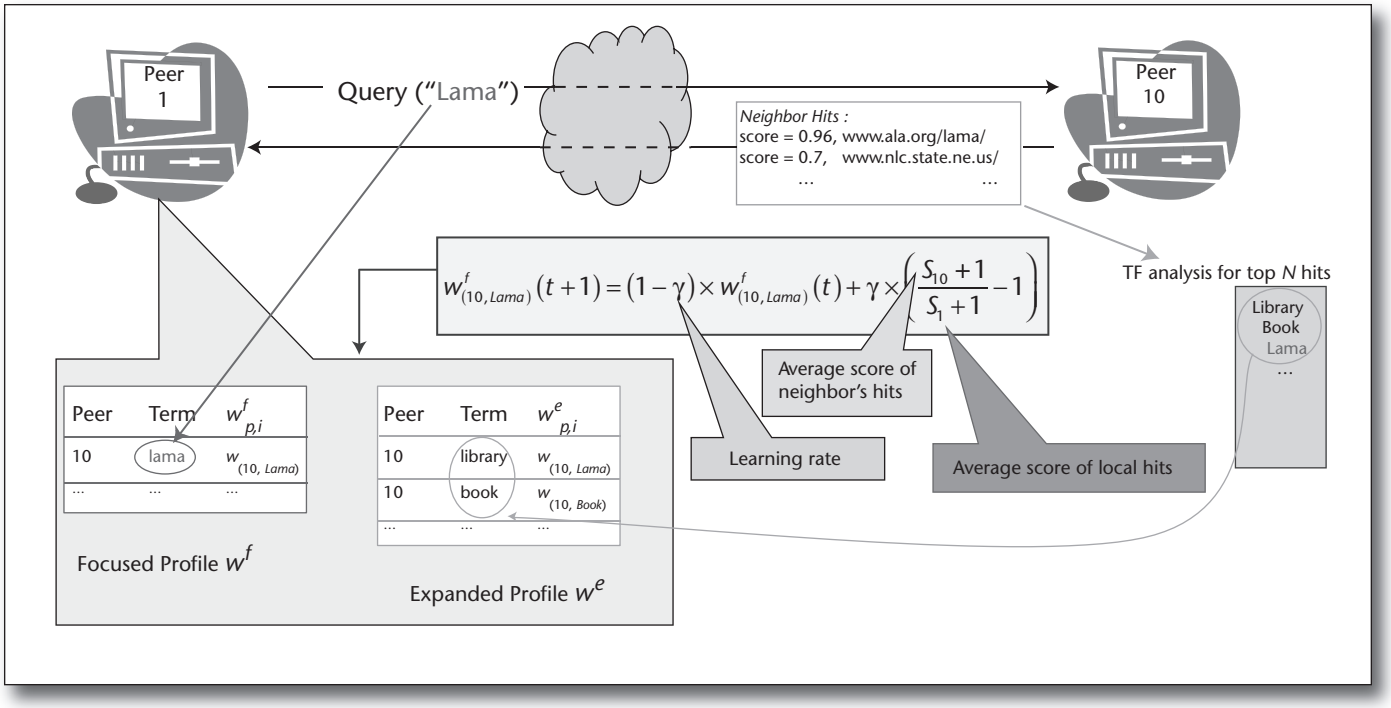


Figure 5. Peer Profile Update.

A peer's response to a query can indicate a peer's knowledge with respect to that query. This knowledge is captured by the focused profile, W^f. In addition, keywords that co-occur with query terms within hit pages may reflect (less reliable) information about the peer's knowledge. This is captured by the expanded profile, W^e.

average score of $S_1 = 0.2$. If the previous value of the weight associated with the term "Lama" in Alice's profile of Peer 10 was zero, the new value would be $w^f_{10,Lama} = 0.5\gamma$, where γ is a learning rate ($0 < \gamma < 1$). For multiword queries, the same update rule is applied to each term in the query.

In principle, a peer could track an arbitrarily large number of other agents. Every time that a new agent is discovered, its profile can be added to W. In practice, the size of W may be limited by storage availability. An agent can drop profiles for the least promising peers when space shortage requires it. Queries can only be routed to known agents, that is, those whose profiles are in W. To route a new query, known peers are ranked by the similarity between their profiles and the query, as shown in figure 6.

Each 6S agent uses the above peer learning and query-routing algorithms to refine a model of the other peers. The collaborative network in 6S is formed by the dynamic communication among the peers: queries and responses being sent and forwarded. The instantaneous topology of such a collaborative network reflects several dynamic processes: the changing web collections indexed by the peers, the evolving information needs of the users, and the knowledge that agents learn about others. Initially, when peers know nothing of each other, queries are routed randomly, and we observe a random network topology. As 6S agents refine their internal models of others based on observed queries and responses, query routing becomes more content driven. Semantic locality means that queries should be routed efficiently toward knowledgeable peers, and peers with similar interests should end up closer in the collaboration network. We postulate that such a locality should lead to the emergence of semantic clusters, as illustrated in figure 7, and thus prevent congestion.

The 6S Collaboration Network

Before the 6S prototype was developed, we experimented with a number of peer representations and machine-learning algorithms for query routing by running simulations with realistic synthetic users and queries. The details of these simulations and our findings have been reported elsewhere (Wu, Akavipat, and Menczer 2005; Akavipat et al. 2006). We found five promising properties about the 6S network, highlighted by these experiments: (1) The agents rapidly form clusters (spontaneous groups that communicate more within the group than outside), displaying a query topology that converges to a small-world network after each peer has routed as few as five or six queries, and this change in topology leads to an increase in the quality of the results. (2) The clusters, which are

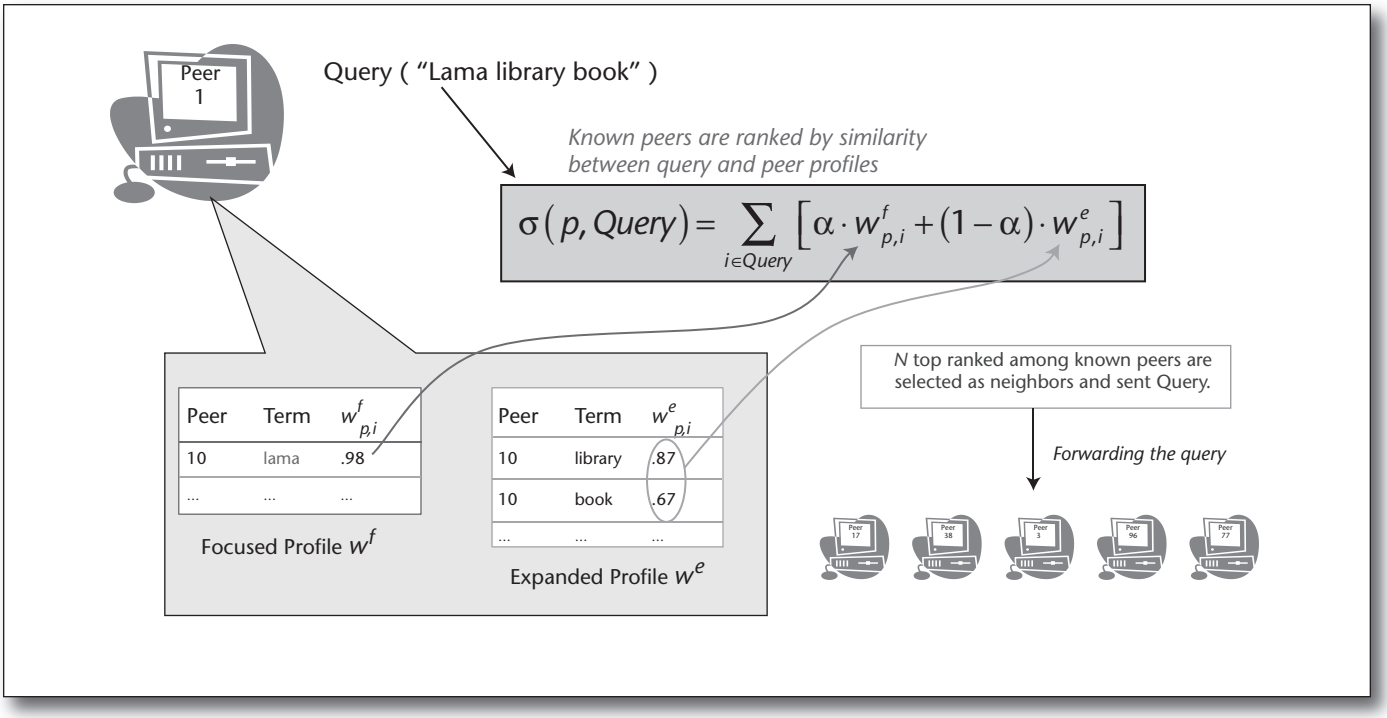


Figure 6. Peer Selection for Query Routing.

For forwarding a query, known peers are ranked by similarity between the query and the peer profiles. The reliability parameter regulates the contributions of focused and expanded profiles. Typically 0.5 < < 1 to reflect higher confidence in focused profile weights as they come from direct responses to queries.

formed by agents' query traffic, identify communities of peers with similar interests, indicating that the network exhibits semantic locality. (3) The collective search performance of the network improves when more sophisticated learning algorithms are employed by the agents to route queries, and as more network resources become available. Performance degrades gracefully as bandwidth and CPU cycles become scarcer. (4) The 6S peers achieve a search quality (in terms of precision and recall) that is comparable to that of Google and significantly outperform a centralized search engine with the same resources (crawl size) as the combined 6S peer collective. (5) The 6S algorithms scale well up to 500 peers, the maximum number of users we were able to simulate in a closely controlled testing environment.

Since the release of the 6S prototype, we have been tracking a small community of early adopters to see if these results hold "in the wild." This user study is designed to observe how people use 6S and how the collaborative search network evolves with users' activities. To this end, data is recorded and transmitted from participants' computers to a collection server through a secure channel once a day. The data collected includes query routing information, queries, results, size of personal web index, and most common indexed terms. Figure 8 plots the activity of the network in its first 12 weeks of life. The data and feedback we are collecting are helping to improve the software by making it more transparent, persistent, robust, and interactive. For example, in the prototype used to collect this data, the application does not run in the background, so that users quitting the application automatically leave the network. This behavior has been changed in recent releases, so that a peer can remain active and useful even when the user is not interacting with it.

Figure 9 visualizes the collaborative search network. We can distinguish the query network, which shows the propagation of queries among peers, from the response network, which shows who provides results to whom. There is evident heterogeneity in the number of queries received and results sent. One of the nodes in the network is a special peer that submits queries to the Yahoo search engine through its application programming interface (API) and returns the results obtained from Yahoo. This node is effectively "Yahoo in disguise"—but the other peers know nothing of its identity. We wanted to determine whether the network would learn to rely on this peer, which is clearly very good, given its universal expertise. Indeed, the Yahoo peer does become very central, with the highest number of incoming queries and also the highest number of incoming

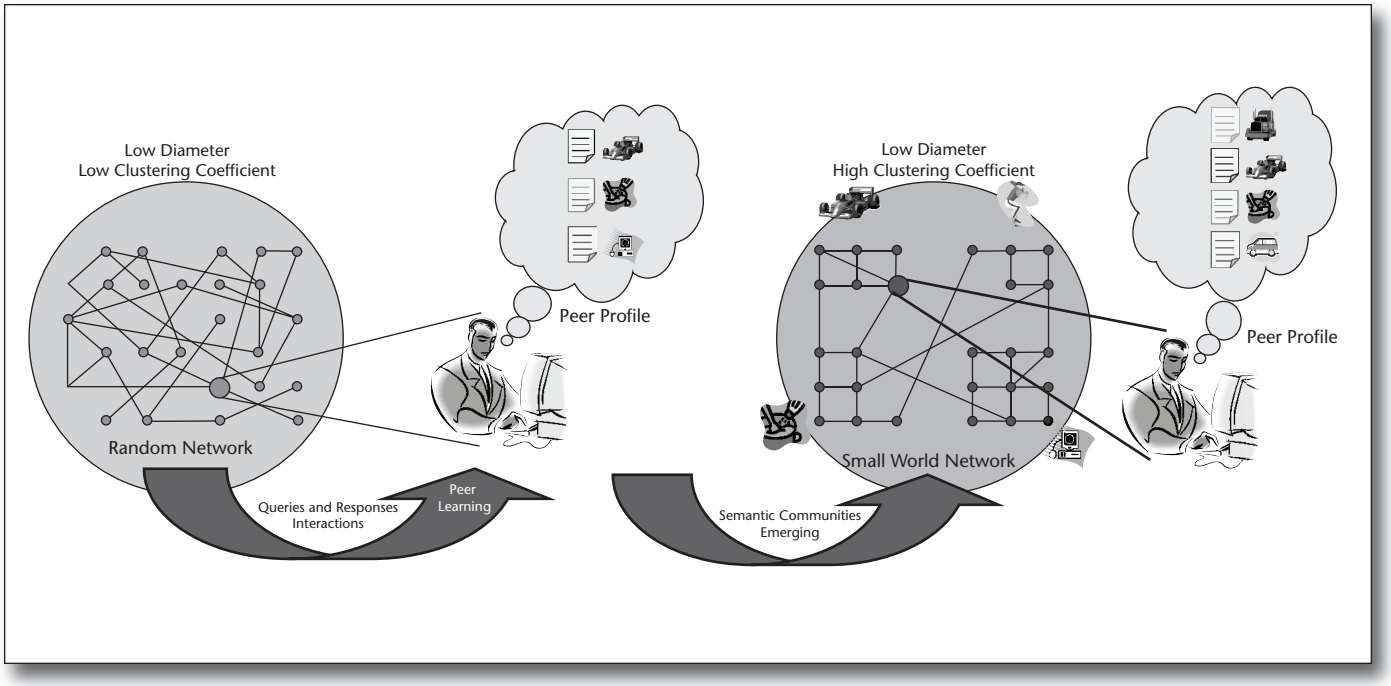


Figure 7. Semantic Locality in Emergent 6S Communities.

(The networks shown are a conceptual mock-up.) Agents initialize and maintain peer profiles by first asking a neighbor for its description, defined as a list of most frequent keywords in the neighbor's index, and then updating these profiles through query/response interactions. Such interactions cause the peers to route queries in such a way that peers with similar interests cluster together to find quality results quickly (high clustering coefficient), while it is still possible to reach any peer in a small number of steps (small diameter).

edges (peers that forward queries to it) for most of the experiment duration. It also provides many results to other peers.

To visualize the query network (shown on the left side of figure 9), we aggregated the queries routed during week 12 of our user study. This was the week with the largest number of active peers. Edge width is proportional to the number of queries exchanged between two peers. The area of each node is proportional to the number of queries received by the peer, which is an indirect measure of centrality, authority, or reliability of the peer as learned by the other agents. To visualize the response network (shown on the right side of figure 9), we aggregated results sent during week 6, which was the one with the largest number of queries and responses. This network is visualized from end to end, that is, an edge directly connects the provider and the receiver of a result, irrespective of the chain of peers through which the results were actually routed. Edge width is proportional to number of results exchanged, and node size is proportional to number of results provided. Thus, larger nodes are more helpful. In both networks, inactive nodes (those with no incoming queries or outgoing results) are not shown. The node marked with a white rectangle is the Yahoo search engine in disguise (see the previous paragraph); by design, this peer does not generate or forward queries, yet it is the most popular target of queries and the second most productive provider of results.

Figure 10 plots the small-world statistics of the 6S collaborative query network within our user study period. The *diameter* is defined as the average shortest path across all pairs of nodes (with adjustments to deal with disconnected networks). The network's *clustering coefficient* is the average of nodes' clustering coefficients, across all nodes. An individual node *i*'s clustering coefficient c_i is the fraction of triangles in which *i* participates, out of the possible ones. That is, c_i is the number of pairs of neighbors of *i* that are also neighbors of each other, divided by the total number of pairs of neighbors of *i*. It is interesting to compare these measures with what one would observe in a random network that is known to have a very short diameter and a very small clustering coefficient. Therefore, for each week, we construct an ensemble of random networks with the same numbers of nodes and edges as the 6S networks. Then we measure by how much the diameter and clustering coefficient in 6S exceed the average ones from the random networks. As figure 10 shows, the diameter remains small but the clustering coefficient

grows considerably. These conditions indicate the emergence of a small-world topology in our peer network (Watts and Strogatz 1998).

Related Work

A P2P computer network relies on the computing power and bandwidth of the participants in the network rather than concentrating it in a relatively few servers. The most popular use of a P2P network is for file sharing. Applications such as Gnutella, BitTorrent and KaZaa (Androutsellis-Theotokis and Spinellis 2004) allow peers to share content files without having to set up dedicated servers and acquiring large bandwidth to support the whole community. P2P file-sharing applications are by no means replacing dedicated servers in content distribution. They simply provide an alternative for content distribution by trading the speed and reliability of dedicated servers for the ease of sharing, lower cost, fault tolerance, and lower bandwidth requirement of a file sharer.

Just as P2P file-sharing applications are used to facilitate content distribution, P2P applications can be developed to facilitate web search. There is a wide variety of peer-based search applications. For example, a model proposed by the YouSearch project is based on maintaining a centralized search registry for query routing (such as Napster), while providing the peers with the capability to crawl and index local portions of the web (Bawa et al. 2003). NeuroGrid employs a learning mechanism to adjust metadata describing the contents of nodes (Joseph 2002). A similar idea has been proposed to distribute and personalize web search using a query-based model and collaborative filtering (Pujol, Sangüesa, and Bermúdez 2003).

An intermediate approach between the completely decentralized flood network (as in Gnutella) and the centralized registry is to store index lists in distributed, shared hash tables (Suel et al. 2003). In pSearch (Tang, Xu, and Dwarkadas 2003), latent semantic analysis (Deerwester et al. 1990) is performed over such distributed hash tables to provide peers with keyword search capability. Another alternative is that of hybrid peer networks, in which multiple special directory nodes (hubs) construct and use content models of neighboring nodes to determine how to route query messages through the network (Lu and Callan 2003).

Similar ideas are receiving increasing attention in the multiagent literature. For example, a model proposed by Bulka, Gaston, and desJardins (2006) includes a learning algorithm by which each agent uses local information and previous experience to refine a classifier. The agent then uses the classifier to decide which agent groups to join or whether to form a new group to complete a task. Pearce and Tambe (2007) study optimal collaborative strategies based on local interactions for teams of agents to solve distributed constraint optimization problems.

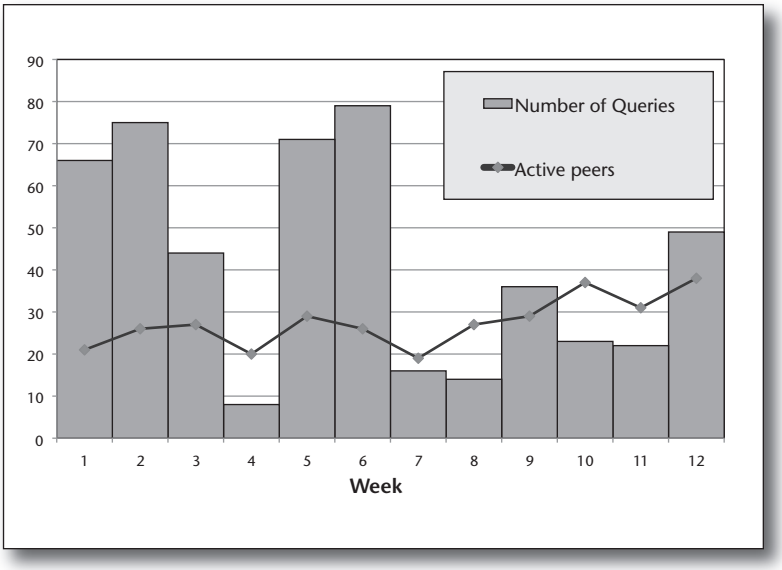


Figure 8. Activity on the 6S Network in the 12 Weeks after the Prototype Release (January–April 2007).

The number of active users (those who submit and forward queries) has increased slowly from about 20 to almost 40. Note that participants can join or leave the network arbitrarily. The query traffic through the network is rather variable, with bursts following releases of software updates.

Status and Future Work

The 6S application is freely available at Sixearch.org. We hope to attract a community of users, which will allow us to test its scalability and robustness, while improving its usability and effectiveness. Because collaborative peer search represents a new paradigm for web search, the interface between the 6S network and its users is critical. It is important that we understand how users interact with 6S and how to best keep their experience positive. The user study, still under way, should provide us with information that will help improve 6S. If users continue to find 6S useful, they will maintain their presence in the peer network.

We plan to explore additional learning algorithms to improve the performance of 6S's adaptive query routing. For example, we want to mine the streams of queries and responses that are forwarded though a peer. In the Gnutella v0.6 file-sharing network, peers tend to issue queries that are very similar to the content of files they have available for sharing (Asvanund et al. 2003). This suggests that a profile of a peer's knowledge should be updated based on the queries the peer issues in addition to the query responses that it produces.

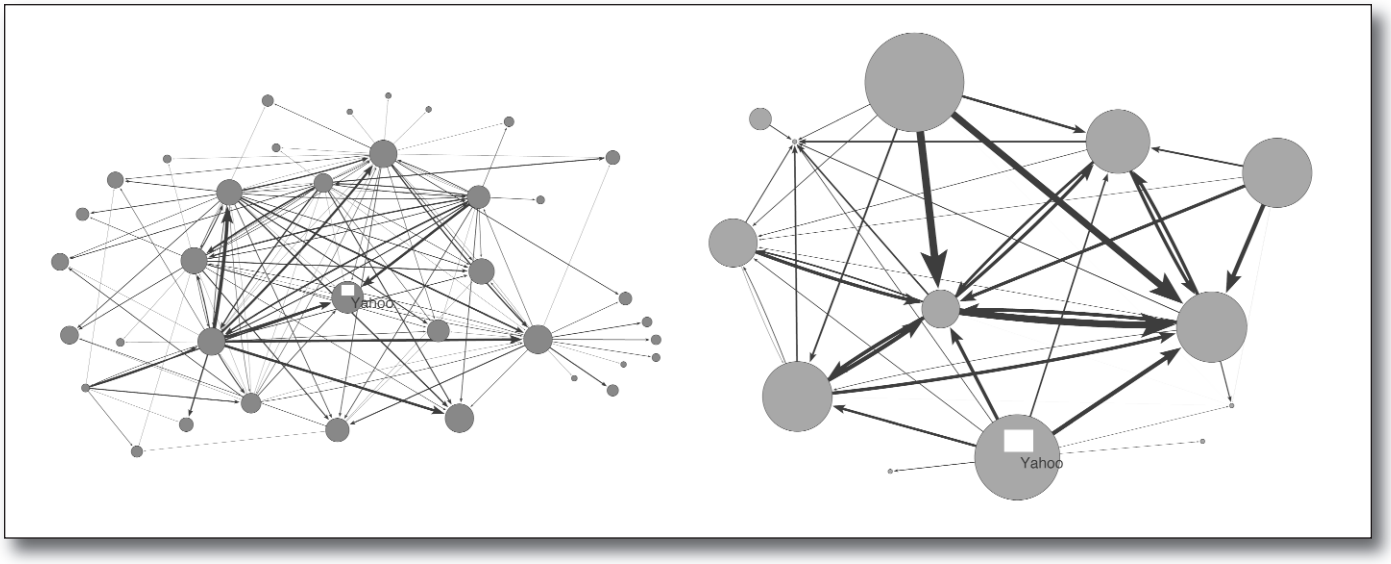


Figure 9. Weekly Snapshots of Two 6S Collaborative Search Networks.

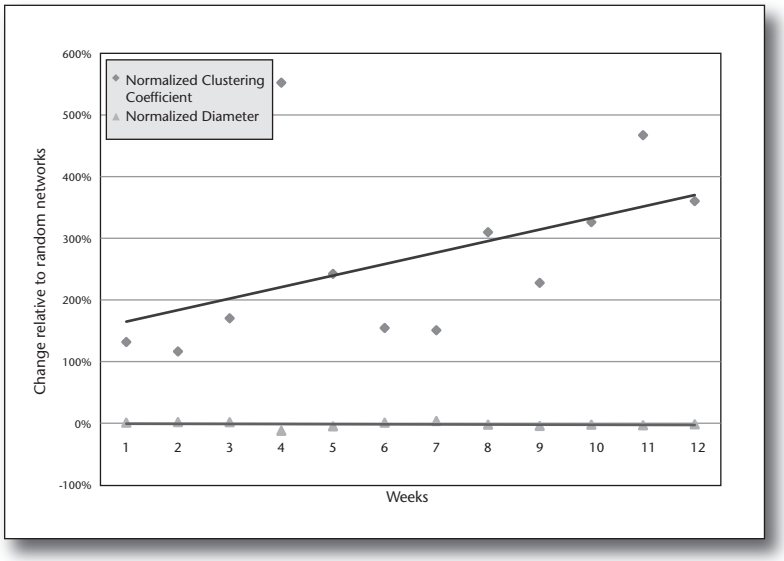


Figure 10. Relative Difference between the Diameter and Clustering Coefficient of the Collaborative Query Network and Those in Random Networks.

To measure both diameter and clustering coefficient, we disregard edge directionality. Trend lines show that the diameter remains equal to the random graph diameter, while the clustering coefficient increases considerably, compared to the random graphs.

Another technique we would like to examine, query relaxation, was proposed in a semantic web setting (Tempich, Staab, and Wranik 2004). A peer that queries for Research Description Framework (RDF) data assumes that a neighbor may have knowledge about a topic or query if it has knowledge about a more specific version of the topic or query. While our application is arguably more difficult due to the unstructured nature of generic web pages, we hope that the promising scalability results obtained for semantic web data will generalize to web information retrieval.

A number of other information-retrieval techniques are also under consideration. For example, profiles in the current prototype are based on simple vector space representations. Similarity between queries and documents is based on simple vector cosine measures. While these techniques are well established, they have limitations when one considers keyword sparsity, ambiguity, synonymity, and so on. Richer representation, for example based on co-occurrence statistics (for example, LSI [Deerwester et al. 1990]) or semantic ontologies (for example, WordNet) could address some of these issues.

A peer selection algorithm should be able not only to determine which peers are best suited for a given query but also to predict which combinations of peers provide the least-redundant results. Existing peer selection algorithms take into account only the predicted query-specific precision quality of known peers for peer ranking. In a purely unstructured network such as 6S, however, each peer crawls the web independently based on its own interests, without any central control mechanism. As a result, it is likely that peers with similar interests will have a high degree of overlap between their document collections. Consider the extreme case of two peers with identical collections. In a naive peer selection approach, if one peer is selected as a good neighbor, the other peer will definitely be selected as well. However, forwarding a query to both peers will generate no more relevant results than submitting to one peer alone, due to their collection overlap. We are investigating extensions to the peer selection algorithm in which a peer would pay attention to the overlap

between two neighbors in order to maximize recall as well as precision.

Finally, in developing a collaborative peer-based search network, one has to think about protecting the system from abuse. For example, by exploiting knowledge of how peers learn from query interactions, attackers can craft their responses to make targeted peers favor the attackers for future querying while directing users to spam content. Colluders can also set up peers that provide some high-quality responses but mixed with pointers to spamming peers. In addition, the victims may inadvertently help the attackers by forwarding other peers' queries to them, thus exposing those peers to the same response attacks. To prevent such exploitation, a collaborative search network such as 6S needs a security component. We are working on a reputation system that can help distinguish spammers from honest peers.

Acknowledgements

This paper owes considerable improvements to the comments of two anonymous reviewers. We are grateful to the Apache Lucene project for the Nutch open source search engine code and to Sun Microsystems for the JXTA open source P2P code. This work was supported by Dr. Menczer's NSF Career Grant IIS-0348940.

References

Akavipat, R.; Wu, L.-S.; Menczer, F.; and Maguitman, A. 2006. Emerging Semantic Communities in Peer Web Search. In *Proceedings of the International Workshop on Information Retrieval in Peer-To-Peer Networks (P2PIR '06)*, 1–8. New York: ACM Press.

Androutsellis-Theotokis, S., and Spinellis, D. 2004. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Survey* 36(4): 335–371.

Asvanund, A.; Bagala, S.; Kapadia, M.; Krishnan, R.; Smith, M.; and Telang, R. 2003. Intelligent Club Management in P2P Networks. Paper presented at the Second International Workshop on P2P Systems, 20–21 February, Berkeley, CA.

Bawa, M.; Bayardo Jr, R.; Rajagoplan, S.; and Shekita, E. 2003. Make It Fresh, Make It Quick—Searching a Network of Personal Webservers. In *Proceedings of the 12th International World Wide Web Conference*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Brewington, B. E., and Cybenko, G. 2000. How Dynamic Is the Web? In *Proceedings of the 9th International World Wide Web Conference*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Bulka, B.; Gaston, M.; and desJardins, M. 2006. Local Strategy Learning in Networked Multiagent Team Formation. *Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS)* 15(1).

Chakrabarti, S.; van den Berg, M.; and Dom, B. 1999. Focused Crawling: A New Approach to Topic Specific Web Resource Discovery. *Computer Networks* (Amsterdam, Netherlands) 31(11–16): 1623–1640.

Deerwester, S.; Dumais, S.; Furnas, G. W., F.; Landauer, T.; and Harshman, R. 1990. Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* 41(6): 391–407.

Fetterly, D.; Manasse, M.; Najork, M.; and Wiener, J. 2003. A Large-Scale Study of the Evolution of Web Pages. In *Proceedings of the 12th International World Wide Web Conference*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Joseph, S. 2002. Neurogrid: Semantically Routing Queries in Peer-to-Peer Networks. In *Proceedings of the First International Workshop on Peer-to-Peer Computing*. *Lecture Notes in Computer Science*, 2429. Berlin: Springer.

Lawrence, S., and Giles, C. 1999. Accessibility of Information on the Web. *Nature* 400: 107–109.

Lu, J., and Callan, J. 2003. Content-Based Retrieval in Hybrid Peer-to-Peer Networks. In *Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM'03)*. New York: Association for Computing Machinery.

Menczer, F., and Belew, R. K. 2000. Adaptive Retrieval Agents: Internalizing Local Context and Scaling Up to the Web. *Machine Learning* 39(2/3): 203–242.

Menczer, F.; Pant, G.; and Srinivasan, P. 2004. Topical Web Crawlers: Evaluating Adaptive Algorithms. *ACM Transactions on Internet Technology* 4(4): 378–419.

Ntoulas, A.; Cho, J.; and Olston, C. 2004. What's New on the Web?: The Evolution of the Web from a Search Engine Perspective. In *Proceedings of the 13th International Conference on the World Wide Web*, 1–12. New York: ACM Press.

Pant, G.; Bradshaw, S.; and Menczer, F. 2003. Search Engine-Crawler Symbiosis. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Lecture Notes in Computer Science,* Vol. 2769, T. Koch and I. Solvberg, eds. Berlin: Springer Verlag.

Pearce, J. P., and Tambe, M. 2007. Quality Guarantees on k-Optimal Solutions for Distributed Constraint Optimization Problems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*. Menlo Park, CA: AAAI Press.

Pujol, J.; Sangüesa, R.; and Bermúdez, J. 2003. Porqpine: A Distributed and Collaborative Search Engine. In *Proceedings of the 12th International World Wide Web Conference*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee.

Suel, T.; Mathur, C.; Wu, J.-W.; Zhang, J.; Delis, A.; Kharrazi, M.; Long, X.; and Shanmugasundaram, K. 2003. ODISSEA: A Peer-to-Peer Architecture for Scalable Web Search and Information Retrieval. Paper presented at the International Workshop on the Web and Databases (webDB), San Diego, CA, 12–13 June.

Tang, C.; Xu, Z.; and Dwarkadas, S. 2003. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM'03)*. New York: Association for Computing Machinery.

Tempich, C.; Staab, S.; and Wranik, A. 2004. REMINDIN': Semantic Query Routing in Peer-to-Peer Networks Based on Social Metaphors. In *Proceedings of the 13th Conference on the World Wide Web*, 640–649. New York: Association for Computing Machinery.

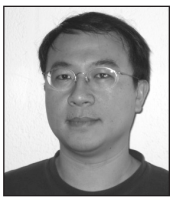
Waterhouse, S. 2001. JXTA Search: Distributed Search for Distributed Networks. Technical Report, Sun Microsystems Inc., Santa Clara, CA.

Watts, D., and Strogatz, S. 1998. Collective Dynamics of "Small-World" Networks. *Nature* 393: 440–442.

Watts, D.; Dodds, P.; and Newman, M. 2002. Identity and Search in Social Networks. *Science* 296(5571): 1302–1305.

Wu, L.-S.; Akavipat, R.; and Menczer, F. 2005. 6S: Distributing Crawling and Searching across Web Peers. In *Proceedings of the IASTED International Conference on Web Technologies, Applications, and Services*. Calgary, Alberta: International Association of Science and Technology for Development.

Filippo Menczer is an associate professor of informatics and computer science, adjunct associate professor of physics, and a member of the cognitive science program at Indiana University, Bloomington. He holds a *laurea* in physics from the University of Rome and a Ph.D. in computer science and cognitive science from the University of California, San Diego. Menczer has been the recipient of Fulbright, Rotary Foundation, and NATO fellowships and a Career Award from the National Science Foundation. His research interests span web, text, and data mining, web intelligence, social web search, and adaptive agents in complex information networks.



Le-Shin Wu is a Ph.D. candidate in computer science at Indiana University, Bloomington. He is a lead developer of the NSF-sponsored Sixearch.org peer search network. Since 2007 he has worked as a principle system analyst for the University Information Technology Services unit. He is pursuing research in the areas of databases, distributed information retrieval, web mining, machine learning, and web services.

Ruj Akavipat is a Ph.D. candidate in computer science at Indiana University, Bloomington. He is a lead developer of the NSF-sponsored Sixearch.org peer search network. He also works as a research associate in Indiana University's Department of Psychological and Brain Sciences. His current research focuses on the development of trust and security measures for distributed, collaborative peer-to-peer systems.