A Distributed Ranking Algorithm for the iTrust Information Search and Retrieval System

Boyang Peng, L. E. Moser, P. M. Melliar-Smith, Y. T. Chuang, I. Michel Lombera
Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA
bpeng@umail.ucsb.edu, {moser, pmms, ytchuang, imichel}@ece.ucsb.edu

Keywords: Distributed Ranking: Search and Retrieval: Peer-to-Peer Network: iTrust.

Abstract: The iTrust system is a decentralized and distributed system for publication, search and retrieval of information over the Internet and the Web, that is designed to make it difficult to censor or filter information. In the distributed ranking algorithm for iTrust presented in this paper, a source node that publishes a document indexes the words in the document and produces a term-frequency table for the document. A requesting node that issues a query and receives a response uses the URL in the response to retrieve the term-frequency table from the source node. The requesting node then uses the term-frequency tables from multiple source nodes and a ranking formula to score the documents with respect to its query. Our evaluations of the distributed ranking algorithm for iTrust demonstrate that the algorithm exhibits stability in ranking documents and that it counters scamming by malicious nodes.

1 INTRODUCTION

Our trust in information accessed over the Internet and the Web depends on benign and unbiased administration of centralized search engines, indexes and repositories. Search and retrieval of information over the Internet and the Web are centralized for efficiency and economy of scale. However, the administrators of those centralized facilities, as well as government agencies and officials, can cause information accessed over the Internet and the Web to be selectively filtered or censored completely.

An alternative distributed search and retrieval system, without centralized mechanisms and centralized control, can reduce people's concerns about filtering and censoring of information on the Internet and the Web. Such a system can provide assurance to its users that a small number of administrators or officials cannot prevent them from distributing their ideas and information and from retrieving the ideas and information of others.

The iTrust information search and retrieval system (Badger et al., 2012; Chuang et al., 2011; Melliar-Smith et al., 2012; Michel Lombera et al., 2013) is a distributed system for search and retrieval over the Internet and the Web, and over social and mobile networks, with no centralized mechanisms and no centralized control. The iTrust system allows people to distribute information, and to search for and retrieve information, within a peer-to-peer network without fear of censorship of information.

When searching for information, people generally prefer to be presented with small sets of ranked documents, rather than unordered sets of all of the documents found. However, selecting and ranking documents in a distributed manner is difficult, because decisions must be made locally.

Most conventional search engines, and other information search and retrieval systems, rely on indexing keywords, because doing so is the most viable way to process large amounts of text. Typically, they rank documents based on a similarity measure between the documents and the user's query, and then return a list of documents that appear to be relevant to that query. After viewing the list of documents, the user can then retrieve the documents of interest.

Our objective for the iTrust search and retrieval system is to provide useful and compact ranking of information for textual documents, so that the user can quickly interpret the information and retrieve the documents of interest. We also aim to counter the actions of malicious nodes in the network by distributing the responsibilities of indexing, scoring and ranking among the nodes in the network.

In our distributed ranking algorithm for iTrust, a source node that publishes a document indexes the words in the document and produces a term-frequency table for the document. A requesting node that receives a response for its query uses the URL in the response to retrieve the term-frequency table from the source node. The requesting node then uses the term-frequency tables from multiple source nodes to score the documents with respect to its query using a ranking formula. Thus, in our distributed ranking algorithm for iTrust, the documents are ranked at the requester based on the user's query.

The rest of this paper is organized as follows. Section 2 provides an overview of the iTrust system. Section 3 presents the distributed ranking algorithm for iTrust. Section 4 discusses trustworthiness as it relates to the distributed ranking algorithm, and Section 5 presents an evaluation of the distributed ranking algorithm. Section 6 discusses related work, and Section 6 concludes the paper and presents future work.

2 THE ITRUST SYSTEM

The iTrust system (Badger et al., 2012; Chuang et al., 2011; Melliar-Smith et al., 2012; Michel Lombera et al., 2013) is a decentralized and distributed information publication, search and retrieval system that aims to combat censorship in the Internet and the Web. The iTrust system operates over a peer-to-peer network that comprises a *membership* of *participating nodes* in the network.

Some of the participating nodes, the *source nodes*, publish information and make it available to other participating nodes. The source nodes generate metadata that describes the information they publish and distribute the metadata along with the URL of the information to a subset of the participating nodes chosen at random. Other participating nodes, the *requesting nodes*, request and retrieve information. The requesting node nodes generate requests (queries) that contain keywords, and distribute their requests to a subset of the participating nodes chosen at random.

Nodes that receive a request compare the keywords in the request with the metadata they hold. If a node finds a match (also called an *encounter*), it returns the URL of the associated information to the requesting node. The requesting node can then use the URL to contact the source node. A *match* between the keywords in a request and the metadata held by a node might be an exact match or a partial match, or it might correspond to synonyms.

The steps for the operation of iTrust, with the distributed ranking algorithm, are described below and are illustrated in Figures 1-4.

1. Nodes with information (the source nodes) distribute their metadata at random to a subset of the participating nodes. With a given probability, those nodes might forward the metadata they receive to other nodes in the network.
2. Nodes that need information (the requesting nodes) distribute their requests at random to a subset of the participating nodes. Again, with a given probability, those nodes might forward the requests they receive to other nodes in the network.
3. When a node receives both the metadata and a request, the node determines whether the metadata and the keywords in the request match.
4. If a node finds that the keywords in the request match the metadata it holds, the matching node provides, to the requesting node, the URL where the requesting node can retrieve the information.
5. Subsequently, the requesting node can retrieve the information from the source node using the URL provided by the matching node.

This paper focuses on the distributed ranking algorithm for iTrust, and describes what happens when the requesting node receives a response to its request or query containing the URL, as shown in Figure 4.

3 THE DISTRIBUTED RANKING ALGORITHM

Like conventional centralized search engines, iTrust needs an algorithm to rank documents by their relevance to a query. Traditional search engines such as Google, Yahoo, and Bing use spider bots to crawl the Internet looking for Web pages and indexing them prior to a query. However, for a distributed search and retrieval system like iTrust, it is both unrealistic and inefficient for every node to index all of the documents of every other node especially in a dynamic network where nodes are frequently joining and leaving. Thus, iTrust needs a ranking algorithm different from that of traditional search engines.

To design an effective distributed ranking algorithm for iTrust, we must consider:

- What metrics the ranking algorithm will use and what the ranking formula is.
- What information the ranking algorithm needs and how to retrieve that information.

3.1 Ranking Formula

The formula that we use in iTrust to rank documents employs a term frequency factor and an inverse document frequency factor, both of which are commonly

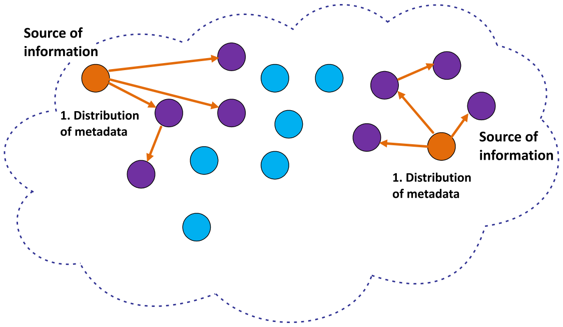


Figure 1: A source node distributes metadata for its documents to randomly selected nodes in the iTrust network.

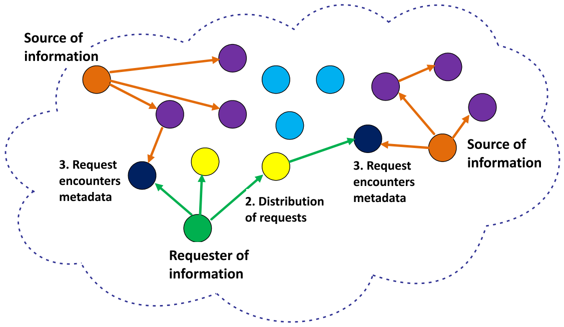

Source of information
3. Request encounters metadata
2. Distribution of requests
3. Request encounters metadata
Source of information
Requester of information

Figure 2: A requesting node distributes requests containing keywords to randomly selected nodes in the iTrust network. When a node in the network receives a request containing keywords that match metadata it holds, an encounter occurs.

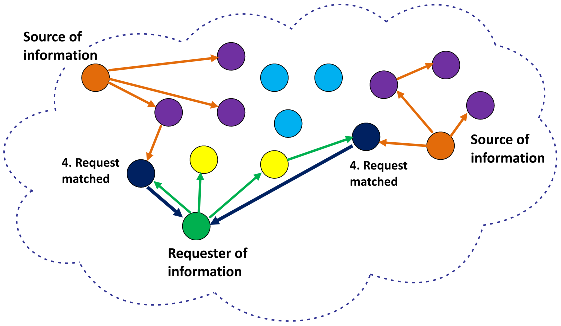
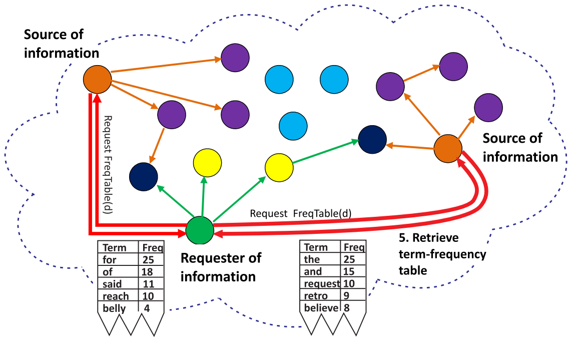


Figure 3: The node that matches the keywords in a request with metadata it holds reports to the requesting node that a match has occurred and supplies the URL of the document at the source node to the requesting node.



Term	Freq
for	25
of	18
said	11
reach	10
belly	4

Term	Freq
the	25
and	15
request	10
retro	9
believe	8

Figure 4: The requesting node retrieves the term-frequency tables from the source nodes.

found in ranking formulas based on the vector space model (Lee et al., 1997; Cohen et al., 2007; Perez-Iglesias et al., 2009). The information needed to rank a document is essentially a table of all of the distinct terms and their corresponding frequencies of occurrence in the document. We refer to this table as the *term-frequency table* or the $freqTable(d)$ for the document d. Each entry in the table consists of a term found in the document and the number of occurrences of the term. In iTrust, we adopt the following formula to score a document d with respect to a query q:

$$Score(d,q) = norm(d) \times \sum_{t \in q} tf(t,d) \times idf(t)$$

where:

- $norm(d)$ is the normalization factor for document d, computed as:

$$log(1 + \frac{number\_of\_uncommon\_terms(d)}{1 + number\_of\_common\_terms(d)})$$

- $tf(t,d)$ is the term-frequency factor for term t in document d, defined as:

$$\frac{log(1 + freq(t,d))}{log(1 + avg(freq(d)))}$$

- $idf(t)$ is the inverse document frequency factor for term t, computed as:

$$1 + log(\frac{numDocs}{1 + docFreq(t)})$$

The normalization factor $norm(d)$ for a document d uses the following definitions:

- *number_of_common_terms* for a document d is $|s \cap c|$, where s is the set of all terms in the document d and c is the set of common terms.
- *number_of_uncommon_terms* for a document d is $|freqTable(d)| - number\_of\_common\_terms$ or the number of entries in the term-frequency table for document d minus the $number\_of\_common\_terms$.

The common terms list is provided for the user, or by the user if preferred. The common terms list is like a stop word list. However, the terms in the common terms list are not simply removed like the words in a stop word list; instead, the common terms are used to marginalize the efforts of malicious nodes to undermine the accuracy of the ranking algorithm.

In particular, a malicious node might distribute a fabricated term-frequency table containing the most common terms used in documents to achieve a higher rank for a document by increasing the probability of matching a query term. However, with high probability, many of the terms in the fabricated term-frequency table are also part of the list of common terms. Thus, the score of a document d will be low, because the $number\_of\_uncommon\_terms(d)$ is small relative to the $number\_of\_common\_terms(d)$.

If the common terms were removed from the query, there might not be a match between the query terms and any of the terms in a document's term-frequency table, which would result in a zero document score. The document corresponding to the fabricated term-frequency table might have a low score but be ranked higher than legitimate documents with a zero document score. However, if the common terms remain in the query, the documents with a zero score might actually have a non-zero score and be ranked higher than the document corresponding to the fabricated term-frequency table.

The term-frequency factor $tf(t,d)$ for term t in document d measures how often term t occurs in document d. The rationale behind the formula is that the score of $tf(t,d)$ is proportional to $freq(t,d)$ and inversely proportional to $avg(freq(d))$.

The inverse document frequency factor $idf(t)$ for term t measures how often term t occurs in all of the documents. The more frequently a term t occurs in all of the documents, the less that term contributes to the score of the document. Consequently, the more uncommon terms are more significant than the more common terms.

3.2 Retrieval of Information

As discussed in the previous section, the ranking formula uses $freqTable(d)$ to score a document d based on a query q. However, the following questions arise: What node is indexing document d to generate $freqTable(d)$? What node is applying the ranking formula to score the document d?

Because iTrust is a distributed peer-to-peer search and retrieval system, we must consider the possibility that some of the participating nodes in the iTrust network are malicious. With network fidelity in mind, we require the requesting node to rank the documents using the ranking formula, because the requesting node represents a user who is searching for information and inputting queries.

The question then arises: How does a requesting node obtain the $freqTable(d)$ corresponding to a response to its query that contains a URL for a document d? As mentioned previously for a peer-to-peer network like iTrust, it is both unrealistic and inefficient for every node to index all available documents. The indexing of documents must be done in a distributed manner. We considered the following alternative approaches:

1. The requesting node downloads all of the documents from the source nodes corresponding to the URLs in the responses it received for its query. The requesting node indexes the downloaded documents. From the information it extracted in the indexing process, the requesting node ranks the documents.
2. The requesting node uses the keywords in its query and the metadata for the documents, contained in the responses it received for its query, to rank the documents.
3. The requesting node supplies the query to the source node and the source node determines the frequency of occurrence of the keywords in the document it holds and then informs the requesting node.
4. The source node indexes the documents. On a request for a particular document d, the source node supplies the term-frequency table $freqTable(d)$ of the document d to the requesting node.

The first approach is simple but inefficient and problematic. The documents downloaded can be large and, consequently, the time needed to present the results to the user can be unreasonably long. Furthermore, the network bandwidth consumed by each request can be large. More importantly, the user might select only a few of the returned results and, thus, downloading all of the documents before the user chooses a few of the results is inefficient.

The second approach has the advantage that the requesting node needs to download only a small amount of data. However, it has the disadvantage that the metadata consists of only a short list of keywords which might not be a good representation of the document described by the keywords, and determining the importance of the keywords to the document can be difficult.

The third approach appears to be appealing, because it is relatively simple and the amount of information that needs to be exchanged between the source node and the requesting node is minimal. This approach would work well if there were no malicious nodes in the network. However, we cannot make such an assumption and one of the goals of iTrust is to be resistant to malicious attacks. If the query is given to the source node, the source node can easily claim that a document it holds has a high number of occurrences of the keywords in the query. Thus, giving the source node the query can result in a malicious source node falsifying and exaggerating information to achieve a higher rank for its document at the requesting node.

The fourth approach utilizes more effectively the distributed peer-to-peer nature of the iTrust system and, thus, it is the approach we use. In our distributed ranking algorithm, the source node performs the indexing of the documents that it holds. For each such document d, the source node maintains a term-frequency table, $freqTable(d)$, for d in the form of an SQL table. The table is sorted by the frequency of occurrence of each term, from the most frequent to the least frequent. When the requesting node receives a response to its query, the requesting node retrieves the term-frequency table, $freqTable(d)$, from the source node using the URL in the response it received from the matching node, as shown in Figure 4.

4 TRUSTWORTHINESS

Now we consider the possibility of malicious nodes that attempt to subvert the distributed ranking algorithm of iTrust. A malicious node might falsify or exaggerate information to render the ranking algorithm ineffective, as described below.

4.1 Falsifying Information

A malicious node can choose to distribute a term-frequency table containing a list of every word known in the English language, thus scamming the ranking algorithm at the requesting node and causing the document that it is trying to distribute to be ranked high at every requesting node.

To mitigate this problem, we adopt the policy that the requesting node retrieves only a top number T of the entries in the term-frequency table from a source node. If the document does not have T distinct terms, then the requesting node simply retrieves the number of distinct terms less than T that the document does have. Thus, the number T serves as an upper bound on the number of entries in the term-frequency table retrieved from a source node.

The size T of the term-frequency table retrieved is, of course, less than the total number of words in the English language. It should be large enough so that it can represent the content of the document. For documents of reasonable size (100 - 10,000 words), T is set between 100 and 200.

A scammer might distribute a term-frequency table consisting of a list of most frequently occurring terms to obtain as many matches between the terms in the term-frequency table and the query. For our experiments, the scam document corresponding to such a term-frequency table contains a list of the most commonly used words in textual documents.

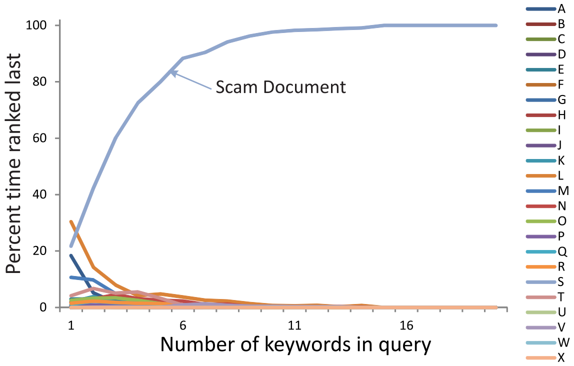


Figure 5: The percent time that a document is ranked last, as a function of the number of keywords in the query. Each line, referenced by a letter of the alphabet, represents a different document. The scam document is referenced by the letter S.

The graph in Figure 5 illustrates the ranking of 28 documents of various lengths, of which one document is a "scam" document. The size of the term-frequency table for all of the documents is 200. From the graph, we see that, by employing the policy that the requesting node retrieves only a top number T of terms from the term-frequency table, the percent time the scam document is ranked last increases dramatically as the number of keywords in the query increases.

4.2 Exaggerating Information

A malicious node can also exaggerate the information about a document it is distributing by replacing the actual frequencies in the term-frequency table with larger frequencies, in an attempt to achieve a higher ranking at the requesting nodes. Such an attempt to scam the ranking algorithm is futile, due to the calculation of the $tf(t,d)$ factor for term t in document d. How much the term-frequency factor adds to a document's score is relative to the average frequency of terms found in $freqTable(d)$. Thus, if a malicious node tries to scam the ranking algorithm by multiplying the frequencies in $freqTable(d)$ by a factor F, the overall average frequency of $freqTable(d)$ is also increased by a factor F, thus decreasing the $tf(t,d)$ factor by the factor F as well.

Even though a malicious node can exaggerate the statistics of a document by a factor of, say, $F = 5$ or $F = 10$, the resulting score and the score of the original document are similar, as shown in Figure 6.

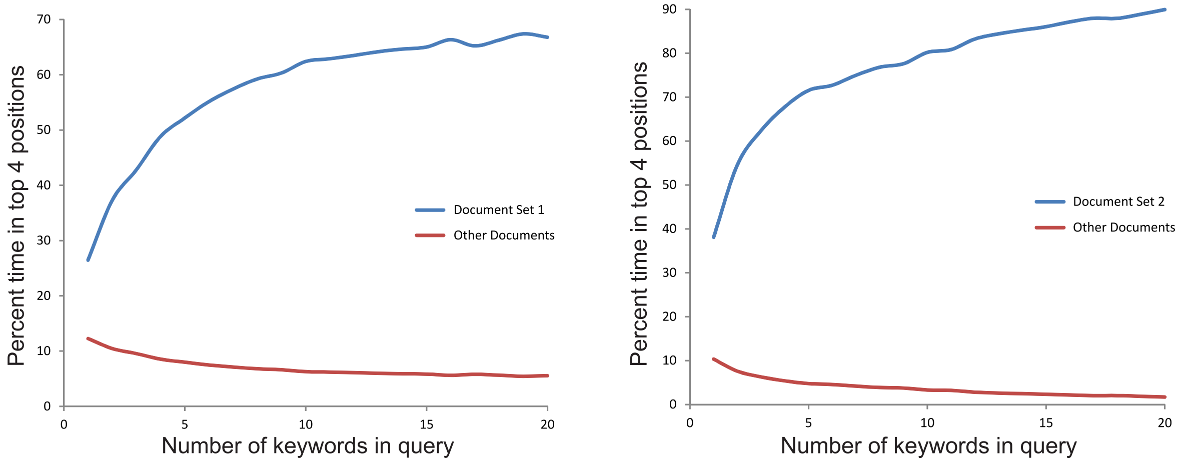


Figure 7: The mean percent time of 1000 rankings that a set of documents (Document Set 1 at the left and Document Set 2 at the right) are ranked the top four, as a function of the number of keywords in the query. At the left, Document Set 1 contains 4 documents, whereas Other Documents comprise the 4 documents in Document Set 2 and the 20 documents in Document Set 3. Similarly, at the right, Document Set 2 contains 4 documents, whereas Other Documents comprise the 4 documents in Document Set 1 and the 20 documents in Document Set 3.

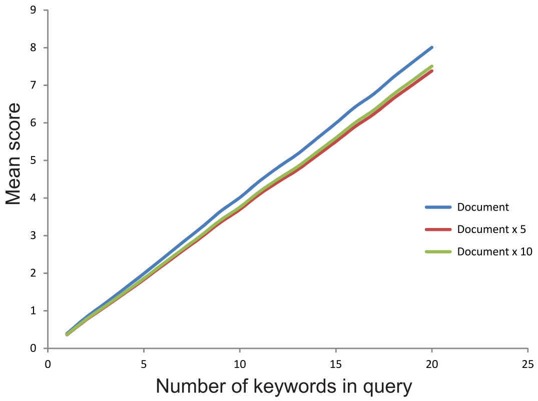


Figure 6: The mean score of 1000 rankings of a document, as a function of the number of keywords in the query. The lines, Document × 5 and Document × 10, correspond to the frequencies in the term-frequency table of a document, multiplied by a factor of 5 and 10, respectively.

5 EVALUATION

Because iTrust is a distributed and probabilistic system, for reproducibility of results, we evaluate the effectiveness of the ranking algorithm by simulation, separate from the iTrust system implementation. We will see that, as the number of keywords used in the query increases, the accuracy of the results increases.

5.1 Document Similarity

Three sets of documents were used to evaluate the effectiveness of the distributed ranking algorithm for iTrust. Document Set 1 consists of documents that are similar in content, as does Document Set 2. Document Set 3 consists of documents of various lengths that are not similar in content to the other documents in the third set and not similar to the documents in the first two sets.

The three document sets contain 28 documents, but Document Set 1 and Document Set 2 each contain only 4 documents. We want Document Set 1 and Document Set 2 to contain only a small proportion of the total number of documents, so that a document in Document Set 1 or Document Set 2 is not ranked high by chance, which might happen if they contained a higher proportion of the total number of documents.

To ensure that the queries are comparable, the queries include keywords randomly selected from one of the documents in either Document Set 1 or Document Set 2. If the ranking formula works correctly, the documents in the set of documents containing the document from which the keywords are randomly selected should be ranked higher than the rest. This holds true regardless of whether a document is selected from Document Set 1 or Document Set 2.

The ability of the ranking algorithm to rank similar documents together can be seen in both graphs in Figure 7. When a document in Document Set 1 is chosen, the documents in Document Set 1 are ranked higher than the rest of the documents and, similarly, when a document in Document Set 2 is chosen.

5.2 Document Ranking Stability

Given an appropriate number of keywords in a query, the rank of a document should not differ much when a similar query is supplied.

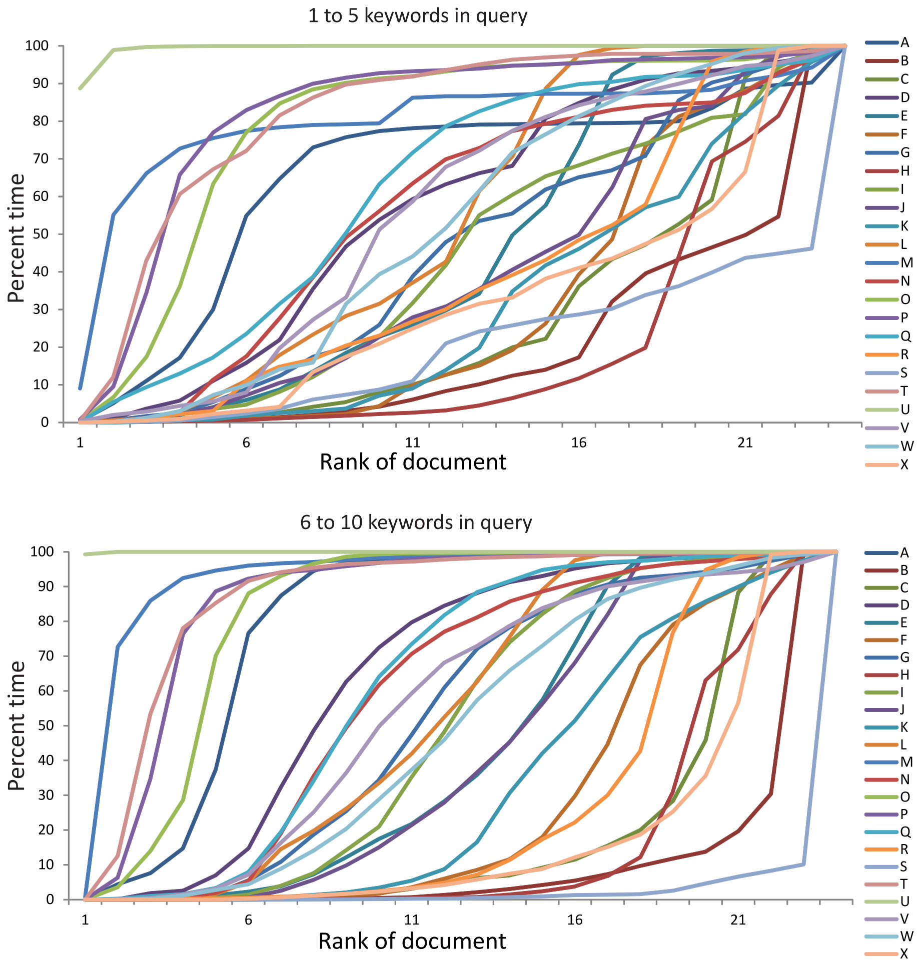


Figure 8: The cumulative distribution of the percent time a document is ranked at a certain position, with term-frequency tables containing 200 entries. The x-axis represents the rank of a document, where 0 represents the top rank. The y-axis represents the percent time that a document is at a particular rank or higher rank. Each line, referenced by a letter of the alphabet, represents the cumulative distribution of a document.

The graphs in Figure 8 show a more comprehensive view of the stability of ranking as the number of keywords in the queries increases, for term-frequency tables with 200 entries. These graphs display cumulative distributions for ranked documents. In particular, for a document d, the graphs display the cumulative distribution $P(X \leq x)$, where x is the rank of document d and $P(X \leq x)$ is the probability that x is greater than or equal to a specific rank X. In the top graph, a line corresponds to the average of the values of $P(X \leq x)$ of a document for queries containing 1 to 5 keywords. In the bottom graph, a line corresponds to the average of the values of $P(X \leq x)$ of a document for queries containing 6 to 10 keywords.

In the top graph, the percent time a document is ranked at some rank x or higher varies considerably. This behavior is undesirable, because the documents should be ranked at similar positions, given similar queries. The position at which a document is most likely to be ranked is unclear and unpredictable.

In the bottom graph, the rank of a document is more predictable. For each line representing a ranked document, there exists a distinct rank such that the percent time the document is ranked at that rank or higher, dramatically increases, which indicates that the document is most likely to be ranked at that rank. This behavior can be more methodically quantified, as the number of ranking position changes between the 20th percent time and the 80th percent time of the cumulative distribution graph of the ranked document. Better behavior of the ranking algorithm corresponds to fewer position changes between the 20th percent time and the 80th percent time. The fewer number of position changes between the 20th percent time and the 80th percent time corresponds to less ambiguous queries and more accurate rankings of the documents.

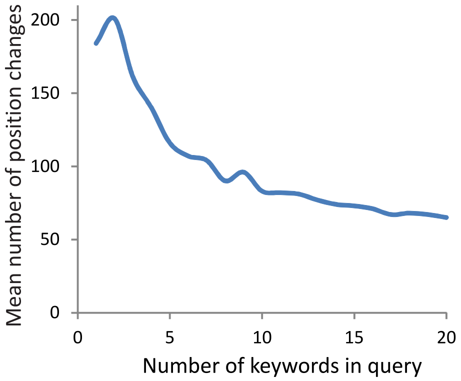


Figure 9: The mean number of position changes between the 20th and the 80th percent time for a set of documents, as a function of the number of keywords in the query.

The behaviors seen in the top graph and the bottom graph of Figure 8 are as expected. With a greater number of keywords in a query, the query is less ambiguous and the results are more accurate.

The graph in Figure 9 shows a condensed view of how the number of words in a query affect the rank stability of a document and, therefore, the accuracy of the ranking of that document. The graph shows the mean number of ranking position changes between the 20th percent time and the 80th percent time in the cumulative distribution $P(X \leq x)$ for a document d, as shown in Figure 8.

In Figure 9, note the steep decrease in the number of position changes from 1-5 keywords in a query. This decrease slows down at 5 or more keywords in a query. That is, a query containing 1-5 keywords creates ranks that are less stable and, thus, less accurate than queries containing 6-10 keywords. This behavior is expected because, as the number of keywords in a query increases, the query becomes less ambiguous.

We conclude that, as the number of keywords in the queries increases, the more stable the ranking of documents becomes which, in turn, makes the ranking algorithm more accurate. From Figure 8 and Figure 9, we conclude that the number of keywords in a query should be at least 5.

5.3 Appropriate Size of the Term-Frequency Table

With the policy that a requesting node retrieves only a number T of entries from a document's term-frequency table, the question remains: What is an appropriate size of the term-frequency table? The graph in Figure 10 depicts the mean number of position changes, as a function of the size of the term-frequency table, among a set of documents between the 20th percent time and the 80th percent time in the cumulative distribution $P(X \leq x)$ for a document d, shown in Figure 8. The set of documents consists of documents of various lengths.

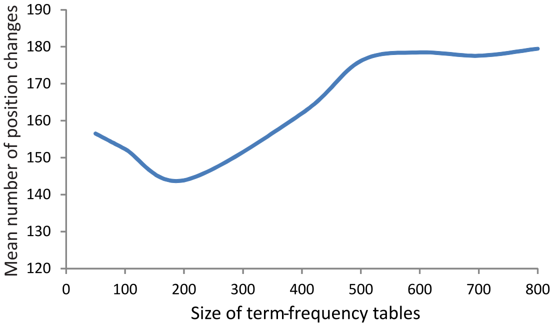


Figure 10: The mean number of position changes of the rank of the documents in a document set, as a function of the size of the term-frequency tables of the documents in the document set.

From the graph in Figure 10, we see that the mean number of position changes of the rank of a document in a document set is the least when the term-frequency table contains about 200 terms (entries). A term-frequency table of size 200 is a good choice, particularly for a set of documents with various lengths. A term-frequency table of size 200 diminishes the occurrence of erratic document length biases in the ranking process which are responsible for the instability of ranking. Thus, to achieve stability and effective behavior of the ranking algorithm, the size of the term-frequency tables should be around 200 for reasonsable size documents.

6 RELATED WORK

Although a lot of research has been conducted on ranking formulas, very little research has been conducted on distributed ranking algorithms. We discuss some of this research below.

The Lucene ranking formula (Lucene, 2009) is used in the Apache Web server, and several groups of researchers (Cohen et al., 2007; Perez-Iglesias et al., 2009) have proposed improvements to it. We adopt many of the improvements they suggest in our ranking formula to improve its effectiveness and accuracy. However, the scoring they consider is targeted toward traditional centralized search engines, and is not distributed. In their systems, all of the available information is located in one central location, rather than being distributed among many nodes as in iTrust.

The vector space model (Lee et al., 1997) has been widely used for ranking, by the information retrieval research community. The vector space model allows a document that contains a subset of the query terms to be ranked relatively high if the terms in the subset occur infrequently in the entire corpus but frequently in the document.

PlanetP (Cuenca-Acuna et al., 2003) is a content-based, peer-to-peer search system that uses the vector space model. Nodes store vectors locally, but gossip digests of their local content. Queries are evaluated by first ranking the nodes and then evaluating the queries at the top-ranked nodes.

Other researchers (Gopalakrishnan et al., 2007) have extended a class of well-known information retrieval ranking algorithms to decentralized systems, including both structured and unstructured networks. Their distributed ranking algorithm for keyword-based queries employs approximation techniques that use the vector space model and uniform random sampling to estimate term weights. Their sampling-based algorithm has the advantage that the cost of the algorithm remains constant as the size of the network increases. However, it does not address the possibility of malicious nodes, like our distributed ranking algorithm for iTrust does.

The TileBars system (Hearst, 1995) allows a user to view the length of the retrieved document, the number of query terms in the document, and the frequency of each term. Our distributed ranking algorithm for iTrust allows the requester to retrieve the term-frequency table from the source node and view it, in relation to the requester's query. Based on the term-frequency tables obtained from different source nodes, it ranks the documents at the requester. The user can then retrieve the documents of interest. The TileBars system allows the user to view the placement of terms in the text and the overlapping of terms in the document, which currently our distributed ranking algorithm for iTrust does not do.

Other researchers (Kalogeraki et al., 2002) have described an intelligent search mechanism for distributed information retrieval for the Gnutella file sharing network (Gnutella, 2000). Their mechanism requires a node in the network to maintain profiles of the peers that are directly connected to it. A profile contains the most recent past responses of the peer. A peer ranks its peers using the profiles and a specific query, and then sends the query to the peer that is most likely to provide an appropriate response. Unlike our distributed ranking algorithm for iTrust, their mechanism does not consider malicious nodes that attempt to undermine or scam the system by falsifying or exaggerating information.

The Distributed WWW Index Servers and Search Engine (D-WISE) (Yuwono and Lee, 1997) consists of a set of index servers, instead of a single index server. D-WISE addresses the scalability issue that arises in the case of a single index server due to the large number of documents on the Web. In iTrust, we carry this idea further in that every source node acts as an index server. However, for iTrust, the issue is not so much scalability as it is trust. A single index server might be malicious and might demote or censor documents and might promote its own documents to higher ranks. Therefore, in our distributed ranking algorithm for iTrust, the requester is given the responsibility of scoring and ranking the documents from different source nodes, based on its query.

Other researchers (Melnik et al., 2001) have constructed a distributed inverted index for a collection of Web pages. The inverted index consists of a set of inverted lists, one for each word (index term) in the collection. The inverted list for a term is a sorted list of locations where the term appears in the collection. A location consists of a page identifier (which might include the number of occurrences of the term in the page) and the position of the term in the page. To speed up index construction, the system uses pipelining. However, their focus is somewhat different than ours, and they do not consider malicious nodes.

Other researchers (Shi et al., 2003) have described Open System PageRank and two distributed page ranking algorithms, derived from the Page-Rank algorithm used by the centralized Google search engine (Page et al., 1998). Page ranking is based on the idea that, if a page P_1 has a hyperlink to another page P_2, then P_1 is implicitly conferring some kind of importance to P_2. The distributed page ranking algorithms employ a set of page rankers. To divide the Web pages among the page rankers, a hash code of the URLs is used. Because the system is iterative, synchronization messages must be sent periodically between the page rankers. The distributed ranking algorithms are intended for structured peer-to-peer networks, whereas iTrust is an unstructured peer-to-peer network, which lacks the hyperlink structure for page ranking and, thus, must employ other ranking mechanisms.

Like our distributed ranking algorithm for the iTrust system, many of the systems described above perform indexing prior to search. However. in iTrust, the indexing of a document is the responsibility of the source node and not another independent entity. Moreover, some of those systems (Yuwono and Lee, 1997; Melnik et al., 2001) use a set of indexing servers to index the documents and process the queries. In iTrust, all of the source nodes serve as the indexers for the documents they hold and the requesting nodes perform the ranking based on the queries that they have made.

7 CONCLUSION

We have presented a distributed ranking algorithm for the iTrust information search and retrieval system. A source node that publishes a document indexes the words in the document and produces a term-frequency table for the document. It also distributes metadata and the URL for the document to a set of randomly chosen nodes in the iTrust network. A requesting node issues a query containing keywords to a set of randomly chosen nodes in the iTrust network. For all responses that the requesting node receives for its query, the requesting node retrieves the term-frequency tables from the source nodes. It then uses the term-frequency tables to score the documents with respect to its query using a ranking formula. Finally, the requesting node retrieves the documents of interest from the source nodes. Our evaluations of the distributed ranking algorithm for iTrust demonstrate that it exhibits stability in ranking documents and that it counters scamming by malicious nodes.

The distributed ranking algorithm for iTrust presented in this paper ranks the documents in terms of their relevance to the query. In the future, the reputation of a document or the reputation of the source node holding the document might be taken into account when scoring the document. Future work also includes evaluating the distributed ranking algorithm on a larger and more varied set of documents. It also includes postulating additional schemes that malicious nodes might use to gain an unfair advantage in the ranking process and devising countermeasures to those malicious schemes.

ACKNOWLEDGMENTS

This research was supported in part by the U.S. National Science Foundation under grant number NSF CNS 10-16193 and by an REU supplement to support the first author.

REFERENCES

Badger, C. M., Moser, L. E., Melliar-Smith, P. M., Lombera, I. M., and Chuang, Y. T. (2012). Declustering the iTrust search and retrieval network to increase trustworthiness. In *Proceedings of the 8th International Conference on Web Information Systems and Technologies*, pages 312–322.

Chuang, Y. T., Michel Lombera, I., Moser, L. E., and Melliar-Smith, P. M. (2011). Trustworthy distributed search and retrieval over the Internet. In *Proceedings of the 2011 International Conference on Internet Computing*, pages 169–175.

Cohen, D., Amitay, E., and Carmel, D. (2007). Lucene and Juru at Trec 2007: 1 million queries track. In *Proceedings of the 16th Text REtrieval Conference, http://trec.nist.gov/pubs/trec16/papers/ibm-haifa.mq.final.pdf*.

Cuenca-Acuna, F. M., Peery, C., Martin, R. P., and Nguyen, T. D. (2003). PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In *Proceedings of the 12th Symposium on High Performance Distributed Computing*, pages 236–246.

Gnutella (2000). http://gnutella.wego.com/.

Gopalakrishnan, V., Morselli, R., Bhattacharjee, B., Keleher, P., and Srinivasan, A. (2007). Distributed ranked search. In *Proceedings of High Performance Computing, LNCS 4873*, pages 7–20. Springer.

Hearst, M. A. (1995). TileBars: Visualization of term distribution information in full text information access. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 59–66.

Kalogeraki, V., Gunopulos, D., and Zeinalipour-Yazti, D. (2002). A local search mechanism for peer-to-peer networks. In *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 300–307.

Lee, D. L., Chuang, H., and Seamons, K. (1997). Document ranking and the vector space model. *IEEE Software*, 14(2):67–75.

Lucene (2009). http://lucene-apache.org/java/docs/.

Melliar-Smith, P. M., Moser, L. E., Michel Lombera, I., and Chuang, Y. T. (2012). iTrust: Trustworthy information publication, search and retrieval. In *Proceedings of the 13th International Conference on Distributed Computing and Networking, LNCS 7219*, pages 351–366. Springer.

Melnik, S., Raghavan, S., Yang, B., and Garcia-Molina, H. (2001). Building a distributed full-text index for the Web. *ACM Transactions on Information Systems*, 19(3):217–241.

Michel Lombera, I., Moser, L. E., Melliar-Smith, P. M., and Chuang, Y. T. (2013). Mobile decentralized search and retrieval using SMS and HTTP. *ACM Mobile Networks and Applications Journal*, 18(1):22–41.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The PageRank citation ranking: Bringing order to the Web. In *Technical Report, Stanford University Database Group*.

Perez-Iglesias, J., Perez-Aguera, J. R., Fresno, V., and Feinstein, Y. Z. (2009). Integrating the probabilistic model BM25/BM25F into Lucene. In *arXiv preprint arXiv:0911.5046v2 [cs.IR]*.

Shi, S., Yu, J., Yang, G., and Wang, D. (2003). Distributed page ranking in structured P2P networks. In *Proceedings of the 2003 International Conference on Parallel Processing*, pages 179–186.

Yuwono, B. and Lee, D. L. (1997). Server ranking for distributed text retrieval systems on the Internet. In *Proceedings of the Fifth International Conference on Database Systems for Advanced Applications*, pages 41–50.