Distributed Information Retrieval For Disruption-Tolerant Mobile Networks

Yun Zhou Brian Neil Levine W. Bruce Croft
Department of Computer Science, Univ. of Massachusetts, Amherst, MA 01003
{yzhao,brian,croft}@cs.umass.edu

ABSTRACT

We design and evaluate a distributed information retrieval system that operates over a mobile network where a wireless infrastructure unavailable. Such networks are common in developing nations, disaster-stricken areas, and even in the rural areas of the technologically progressive countries. This poses a new challenge for distributed IR, which normally relies on a wired Internet or always-available wireless coverage among mobile peers. In our mobile system, queries are propagated among peers only as they intermittently are in wireless range of one another. For each query received, peers retrieve top-ranked documents from their local collection and send them to the source of the query. Intermediate peers on the path to the source have to manage a finite buffer filled with documents from multiple collections and multiple queries. When too many documents are in the system, the intermediate peers must drop documents that are either unlikely to be relevant or for which a successful path to the destination is unlikely. To enable such a system, we propose a score normalization technique that works across queries and across multiple collections. We show that our method returns more relevant documents in the mobile network than existing normalization methods, which are not intended for multiple queries. Additionally, we compare our approach to existing networking algorithms for delivering data in such challenged networks. We show that although our method delivers less total documents, it delivers significantly more documents that are relevant to sources of queries.

1. INTRODUCTION

The ubiquity of inexpensive wireless computing devices has the potential to allow the Internet to reach beyond wired, stationary desktops. Users commonly carry Internet-capable PDAs and embedded devices in their pockets and vehicles, and with wireless connectivity, mobiles users could make use an information retrieval service.

However, the provision of a mobile IR service can be difficult for many reasons. First, an infrastructure may not be present to support wireless routing of queries and retrieved documents. This is the predominant case in developing nations, where cell towers are uncommon and Internet access appears rarely outside one cafe in a village, if anywhere at all. Even in the US, cell phone towers are primarily deployed along highways and populous cities. Satellites provide broad coverage but require enormous energy and expensive equipment and services to access. Second, a natural or man-made disaster can destroy infrastructure; e.g., a tsunami, electrical grid outage, or military action can destroy critical infrastructure across a county, state, or country. Radios on mobile devices, like 802.11 and Bluetooth, have ranges of less than 300m. Thus, it is problematic to assume a set of mobile users with only peer-to-peer radio links can provide wireless coverage across a large geographic area without partitions.

The goal of this paper is to design a distributed information retrieval service that operates in such limited environments, allowing the Internet to extend its reach. These environments can vary in severity. For example, at one extreme, workers responding to a disaster may find infrastructure to assist them has been destroyed, including power, phone and cellular networks, and Internet access. Our system would be an invaluable service for improving the quality of medical care in response to such disasters [4]. On the hand, in a more everyday scenario, a user wishing to query an IR service may simply find themselves out-of-range of cell towers or far from a wireless infrastructure allowing access to the Internet. A user carrying a 802.11-enabled PDA without phone service may be only a quarter-mile from the nearest Internet cafe — yet that is far enough to prevent access to web resources.

In this paper, we address these limitations by proposing a distributed IR service that operates over a different kind of mobile network. *Disruption tolerant networks* (DTNs) are a new form of mobile network that provide service even when no contemporaneous end-to-end path of peers exists. To forward packets end-to-end, data is *stored-and-carried* by intermediate mobile nodes until another wireless peer is discovered. Although message delivery latency can be lengthy, the other option is no service at all — when the information requested remains useful longer than the roundtrip time, DTNs are appropriate.

Each node in a DTN may be a PDA carried by a user enabled with an 802.11 radio and storage. Or, nodes may be buses that carry computers with wireless equipment and a hard drive. We operate a 39-node bus-based DTN that roams the county surrounding the Amherst campus[1]. (N.b., our buses travel outside the range of cell tower coverage in western Massachusetts.) In the system we propose here, the buses would pick up queries from the pedestrians they pass by, route the queries to the nearest Internet access point, and then drive the retrieved documents back to the pedestrian, using several different buses at each stage. Just like passengers, the queries and retrieved documents jump from bus-to-bus until the destination is reached.

More specifically, the protocols we propose support text-based distributed IR. We assume a homogeneous set of mo-

[1]See http://diesel.cs.umass.edu for more information.

bile peers that have limited-range radios, and have no wide-area communication infrastructure among them. One or more nodes may have access to Internet resources for resolving queries, but such access will not assist with the routing of information from or to peers.

We assume each peer is willing to act as an intermediary for the routing of both queries and retrieved documents that propagate through the network. Each peer donates a limited-size buffer for routing storage. Queries are propagated through the network by message exchange during limited duration meetings of two peers, called *transfer opportunities*. Each peer that sees a query for the first time returns sorted search results to the query source. Results may come from an Internet resource or from the peer's local storage.

These retrieved documents are returned to the sources by a DTN routing algorithm that we have extended to evaluate the relevance of the stored documents. Because each peer's buffer space is limited, the number of retrieved documents being routed through the network at one time may be larger than the peer can store; as a result, the peer must drop some documents. In our protocol, the peer determines what documents to drop based on both the likelihood of delivering the document as well as the relevance of the document. The documents in the peer's buffer are for different queries, and therefore we need to normalize the ranked document scores across all queries in the buffer, which is substantially different than the related problem of merging results in distributed retrieval [6]. We call our method *Score Normalization Across Queries* (SNAQ).

We offer a number of contributions after reviewing related work in the next section. In Section 3, we detail our model and assumptions. In Section 4, we propose our SNAQ method for normalizing document scores. SNAQ normalizes document scores by utilizing global corpus statistics estimated from a general English corpus. In Section 5, we detail the design of our routing algorithm for document delivery in DTNs. In Section 6, we present our evaluations that show the effectiveness of our approach. Moreover, we show that our routing algorithm provides better retrieval performance than the approaches proposed previously that route without relevance metrics [11, 14, 8]. In Section 7, we summarize the main conclusions of this paper.

2. RELATED WORK

2.1 Information retrieval

Language models [13] have been successfully introduced in the past few years as a new class of probability models supporting effective information retrieval. One basic language model is *query likelihood* [16], where each document is scored by the probability that the query was generated, given the language model of the current document. In this framework, if we assume a multinomial distribution and use linear smoothing, given document D and query Q, the document score is

$$Score(D) = P(Q|D) = \prod_{w \in Q} P(w|D)$$
$$= \prod_{w \in Q} \lambda P_{doc}(w|D) + (1-\lambda) P_{coll}(w|C) \quad (1)$$

where $w \in Q$ is a term in the query. We estimate $P_{doc}(w|D)$ by the number of times w occurs in document D divided by the total number of terms in D. We estimate $P_{coll}(w|C)$ by the number of times w occurs in collection C divided by the total number of terms in collection C. $P_{coll}(w|C)$ smooths the result; it approximates the distribution of a general English by maximum likelihood estimation of terms from collection C. In this paper, we use the query-likelihood model for retrieval.

Another popular language model framework is Lafferty and Zhai's risk minimization framework [10], where the similarity between a document and a query is measured by the negative Kullback-Liebler (KL) divergence between the document model and the query model. If the query model is estimated by the relative frequency of the terms of the query, the KL divergence is equivalent to the query-likelihood model, in terms of the document ranking for a given query.

When documents are retrieved from different collections, ranking is not straightforward. For example, distributed IR systems require a method of merging results from different sources for the same query, including different collections or different search engines. Luo and Callan [15] have a language modeling framework for normalizing document scores from different collections. Manmatha et al [12] propose a method of merging results by modeling the score distributions of a number of search engines.

The fundamental difference between existing work and this paper is that we are interested in scoring documents across different collections and different queries. Specifically, we are interested in the following question: Is document A for query Q_1 more likely to be relevant than document B for query Q_2? In Section 3, we use KL-divergence as a baseline for normalizing document score. We show that a score normalization that is ideally suited for result merging does not produce good results for our problem.

2.2 Networking

Unlike traditional wireless or ad hoc networks of mobile nodes, the common case in disruption tolerant networks (DTNs) is that the peers are disconnected from one another almost all the time. This allows the network to span enormous geographic areas, or alternatively, it removes the requirement that all nodes be powered continuously, which greatly extends the lifetime of a network of battery-powered peers.

The vast majority of work in DTNs has focused on the forwarding of messages from one peer to another. One strategy we've proposed in our previous work [7, 5] is for peers to keep track of the other peers they meet regularly over time. With this approach, peers initialize to 0 their likelihood of successful message delivery to all peers. For example, when a peer A meets another peer B, the former sets the likelihood of delivering messages to B to 1. Then A takes a portion of B's likelihood of delivering messages to the other nodes in the system. These values degrade over time, such that they are reinforced only if A and B meet periodically. This is an approach that is common to later work [11, 14, 8].

In our own previous work [9], we also proposed an IR scheme for networks where peers are sparsely located. In that work, peers each carry only a small randomized portion of a single IR database. The major difference with that work and this paper is that the earlier work assumes peers coordinate their collections ahead of time to ensure random collections; that kind of coordination is not always feasible and we do not assume coordination among peers in this paper. Additionally, in our earlier work, queries are not routed; peers are only allowed to ask neighbors with whom they happen to be in direct contact at the moment the query is gen-

erated. In this paper, peers not only ask neighbors, but also neighbors of neighbors, and so on; and thus by definition, more relevant documents will be available to the source of the query in the approach we propose here.

Finally, we note that it is possible for a single peer to carry a large collection on a DVD or large local hard drive. However, such a collection by definition prevents access to dynamic information, and it is difficult to determine ahead of time all information that may be relevant to a user. Secondly, sharing of heterogeneous collections is clearly more useful than having only resource. Finally, with our solution, if even one node has Internet access, then all reachable nodes have the ability to query any web resource, including google, yahoo, or other services.

3. MODEL AND ASSUMPTIONS

In our model, each peer carries a wireless radio and a finite buffer. Because of limited radio range and geographic area of movement, no network paths of length greater than one exist contemporaneously. Therefore, to enable a distributed IR system in a DTN, peers use their buffer to store and carry messages for one another. Peers exchange messages (including queries and retrieved documents) when they are temporarily in range of another, called *transfer opportunities*. Over time, each peer generates queries, which are propagated through the network so that relevant documents can be retrieved. Peers carry queries retrieved documents as intermediaries between the source of the query and other peers that have access to relevant documents.

We assume each peer has access to a unique collection of documents for resolving queries. In order to apply our results to a broad set of scenarios, we do not specify where the collection resides. We can view the collection as a resource carried by the user on a hard drive; or we can view the collection as a resource accessible through the Internet (via a cell phone data service, Internet cafe connection, or other data service).

We do not assume any coordination among peers of which documents are stored in the local collections. Peers do not have contact ahead of time. Term frequencies are estimated from a representative (English) text corpus and specifically not from the local collections. Note the finite buffer is separate from a peer's collection, and that documents from the peer's collection are never deleted.

At each transfer opportunity between two peers, A and B, the followings events occur:

1. **Delivery check.** If A's document buffer contains a document for a query that originated from B, then A delivers the document to B and deletes the document from its buffer immediately. The same occurs for B.
2. **Query exchange.** A takes from B any query it has not previously seen. Each query, Q, is actually a tuple containing a list of n terms and a source, s, as follows: $Q = ((w_1, \ldots, w_n), s)$. The same occurs for B.
3. **Retrieval.** Peer A retrieves a fixed number of top-ranked documents per query; call this set of new documents N_A. The same occurs for B, where the set is denoted N_B. A sends to B the scores for the documents in A's buffer and the scores for the documents in N_A.
4. **Scoring and Buffer Management.** Peer A creates a ranking based on document scores of the documents in its buffer, the set N_B, and the documents in B's buffer. Since the buffer at A is limited, it must drop some of the documents in this union. A makes decision on which documents to keep in its buffer according to some criteria. So does B.
5. **Document Exchange.** Peer B transfers to A the documents that A did not drop and that are not already in A's buffer. So does A.

All queries are given an expiration time when generated. After the timeout is reached, the peer deletes the query and all associated documents stored in the peer's buffer.

Our goal is to increase the total number of relevant results returned to each peer. Step 4 is the key mechanism that determines the quality of the result and is the subject of the remainder of the paper.

In existing DTN work, buffer management is determined strictly by the destinations of the messages. Messages with destinations for which that are estimated to have a low probability of delivery are dropped first.

In the next section we design a scoring method that is applicable to our scenario. In Section 5, we design hybrid schemes, and in Section 6, we evaluate what scheme performs best.

4. NORMALIZING DOCUMENT SCORES

The approach we propose for DTN-IR buffer management is to drop documents that are likely to be least relevant (and hence, bring the least utility to the network as a whole). Therefore, we require that each peer ranks the documents in its buffer by the likelihood of relevance. When the documents in a peer's buffer are results retrieved for the same query, our problem is the same as the problem of result merging in distributed IR [6]. However, in our system, each peer's buffer will hold retrieval results for different queries, and so we require a method of comparing document scores even when the documents are for different queries. We assume each peer uses the same ranking algorithm, specifically, the query-likelihood model we described in Section 2.

4.1 Score Normalization Across Queries

Two major problems arise when we directly apply Eq. 1 to our scenario. First, if the smoothing collection used in Eq. 1 is directly estimated from each peer's own collection, the document score computed from Eq. 1 is incomparable because each collection has a different word distribution. Second, even if each peer knows the exact global corpus statistics, the document scores are still not comparable because they are dependent on the query itself. For example, from Eq. 1 we see that documents for a long query unfairly tend to have a lower score than documents for a short query. In other words, we are interested in a measure that is both *collection independent* and *query independent*.

4.1.1 Collection Independence

To overcome the first problem, we propose using one large general-purpose English collection as the smoothing collection, C, in Eq. 1. We assume the statistics of this collection are available to all peers in the network. If each collection knows the word distribution of the single complete collection that consists of all available collections, the document scores would be collection-independent since any two collections would give the same score to the same document. In

practice, this actual single complete collection is unknown to each peer and is instead a large, general-purpose English collection. Additionally, since the purpose of $P_{coll}(w)$ in Eq. 1 is just to approximate the probability distribution of general English, we can use another large general-purpose English collection instead when the actual single complete collection is unknown.

The downside of our method is that each peer needs to keep word frequency information of a large background collection. However, in practice, we can reduce space requirement by only storing words that are likely to be used in a query instead of all the words in the collection.

4.1.2 Query Independence

To make document scores comparable across all queries, one natural way is to use the negative KL-divergence between query model and document model as the document score. That is,

$$Score(D) = -KL(\theta_Q||\theta_D) = \sum_w \theta_Q(w) \log \frac{\theta_D(w)}{\theta_Q(w)} \quad (2)$$

where θ_Q is the query model of Q, and θ_D is the document model of D. In such a framework, the document score can be seen as a distance between the distributions of query model and document model.

We propose a new method, called Score Normalization Across Queries (SNAQ), that uses global statistics. Given a query Q, a document D and a collection C, our approach computes the document score for D as follows:

$$\begin{aligned} Score(D) &= -KL(\theta_Q||\theta_D) - (-KL(\theta_Q||\theta_C)) \\ &= KL(\theta_Q||\theta_C) - KL(\theta_Q||\theta_D) \end{aligned} \quad (3)$$

where θ_C is the collection model of C. Here, the collection model is estimated from a large, general-purpose English corpus and can be seen as the document model averaged over all documents in the collection. Thus, $-KL(\theta_Q||\theta_C)$ denotes the distance between the query model and the average document model. The idea is that by using $-KL(\theta_Q||\theta_C)$ as a baseline, we further eliminate the influence of the query.

If the query model is estimated by the relative frequency of query term $w \in Q$ and linear smoothing is used for document model, the KL-divergence method (Eq. 2) can be rewritten as :

$$\begin{aligned} Score(D) &= -KL(\theta_Q||\theta_D) = \sum_{w \in Q} \theta_Q(w) \log \frac{\theta_D(w)}{\theta_Q(w)} \\ &= \sum_{w \in Q} \theta_Q(w) \log \theta_D(w) - \sum_{w \in Q} \theta_Q(w) \log \theta_Q(w) \\ &= \frac{1}{|Q|} \sum_{w \in Q} \log(\lambda P_{doc}(w|D) + (1-\lambda) P_{coll}(w|C)) \\ &\quad - \frac{1}{|Q|} \sum_{w \in Q} \#(w,Q) \log \frac{\#(w,Q)}{|Q|} \end{aligned} \quad (4)$$

where $\#(w,Q)$ is the number of occurrences of w in Q. Our SNAQ method (Eq. 3) can be rewritten as:

$$\begin{aligned} Score(D) &= KL(\theta_Q||\theta_C) - KL(\theta_Q||\theta_D) \\ &= \sum_{w \in Q} \theta_Q(w) \log \frac{\theta_D(w)}{\theta_C(w)} \\ &= \frac{1}{|Q|} \sum_{w \in Q} \log \frac{\lambda P_{doc}(w|D) + (1-\lambda) P_{coll}(w|C)}{P_{coll}(w|C)} \end{aligned} \quad (5)$$

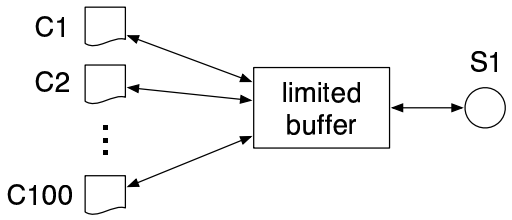

Figure 1: Setup of normalization comparison experiment.

Note that the re-written versions of the KL-divergence method (Eq. 4) and SNAQ (Eq. 5) are both exactly equivalent to the query-likelihood method (Eq. 1) in terms of document ranking for a given query.

4.2 Evaluation

To evaluate the performance of different methods for normalizing document scores, we ran simulations over a central client-server architecture assumed in traditional distributed IR, which represents a very simple version of our DTN scenario. This scenario is illustrated in Fig. 1. Instead of sending one query at a time, the central server sends multiple queries simultaneously to the clients. Each client sends back at the same time all top-ranked documents for all queries it received. We assume that there is a limited buffer space that the retrieved documents must pass through before reaching the central server; documents that do not fit in the buffer are dropped. To make the best use of its buffer, the server has to rank search results for different queries. Given a fixed buffer size, in this paper we are interested only in the total number of relevant documents in the server's buffer no matter which query generated the retrieved results.

We evaluated the three normalization techniques, plus one more:

- Query-likelihood model (Eq. 1);
- KL-divergence (Eq. 4);
- SNAQ (Eq. 5);
- Even allocation of the buffer

In the first three methods, documents from all queries share the whole buffer unevenly. In these methods, documents for query A may take more space than documents for query B. The *even allocation* method assigns a fixed buffer size to each query and merges results independently for each query.

Specifically, even allocation works as follows. Let N denote the total number of queries. We evenly divide the buffer space to each query, that is, each query is assigned $1/N$ of the total available buffer space. For each query, we merge the search results from all collections and keep top-ranked documents in the buffer. For example, if we have 100 queries and the size of available buffer is $1,000$, each query has the buffer size of ten containing the ten top-ranked documents from across all collections for the one query. We use the query-likelihood model (Eq. 1) to compute document scores in each collection. Note that in the case of even allocation, since we compare only document scores from the same query, the three scoring methods (query-likelihood model, KL-divergence, and SNAQ) are equivalent.

	TREC123	WT10G
Number of documents	1.07 Million	1.69M
Number of total terms	249.95M	676.83M
Number of unique terms	0.816M	4.8M
Average document length	231	399

Table 1: A comparison of the statistics of corpuses WT10G and TREC123.

4.2.1 Setup

In our experiments, we assume 100 clients, each assigned one of the 100 collections defined in TREC123-100-bysource-callan99.v2a [1]. We assume the central server sends out 100 simultaneous queries, and all 100 queries are received at all clients. The queries are from the title field of TREC topics 51–151. Each client uses one of the four methods to score the documents in its collection. This collection occupies only a small amount of storage and could be stored on one resourceful PDA; however, our use of TREC if for evaluations and is not intended to be representative of the size or content of collections that would be carried by each peer. In practice we expect each peer would carry (or have access to) a much larger collection that would take up available space, would be heterogeneous from each other peer, and created without coordination with each other peer.

We set the mixing weight to $\lambda = 0.6$ in all four methods. As discussed above, the first three methods are equivalent from the viewpoint of single-query multiple-collection retrieval. In the end, each client returns the ten top-ranked documents for each query (i.e., 100,000 documents are retrieved). In the first three methods, the central server stores only D top-scored documents in its buffer, according to its buffer size. For simplicity, we set the buffer size as a quantity of documents, D, instead of bytes.

The smoothing collection used for the query-likelihood, KL-divergence, and even allocation methods is from TREC CDs 1, 2, and 3, which is the union of all 100 individual collections from clients. We call this union the *global collection*. Given a query, the goal of result merging is to approximate the ranked list that would be produced if the query is retrieved from the global collection. By using the global collection as the smoothing collection, query-likelihood model method can lead to perfect result merging for any given query.

When applying query-likelihood model in our scenario, we act as if all documents are for the same query. Additionally, we assume the global collection is available to KL-divergence method to determine an upper bound on its best performance in the experiment. In practice, this global collection would be unknown, which is a common assumption in distributed IR. With our SNAQ method, we assume no knowledge of the global collection and instead use the WT10G [2] collection for smoothing. Table 1 shows comparison of corpus statistics between two collections. Hence, SNAQ is the most disadvantaged in our simulations.

4.2.2 Results

Figure 2 shows the performance of four methods. Figure 3 shows a zoom of the smallest buffer sizes for the same experiments. On the x-axis is the buffer size measured as the number of documents that can be held on the server's buffer. Shown on the y-axis is the number of relevant documents in the server's buffer for each method. From the figure, we can see that query-likelihood method performs significantly worse than the others because the scoring is highly query-dependent. This implies that scores from the query likelihood model are not quite comparable. Our method performs better than KL-divergence regardless of the buffer size. We can conclude that taking the average document model into account is helpful for normalizing document scores.

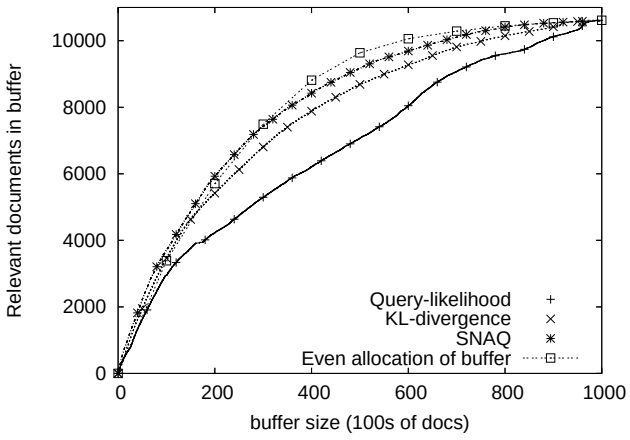

Figure 2: Performance of three normalization methods and even allocation.

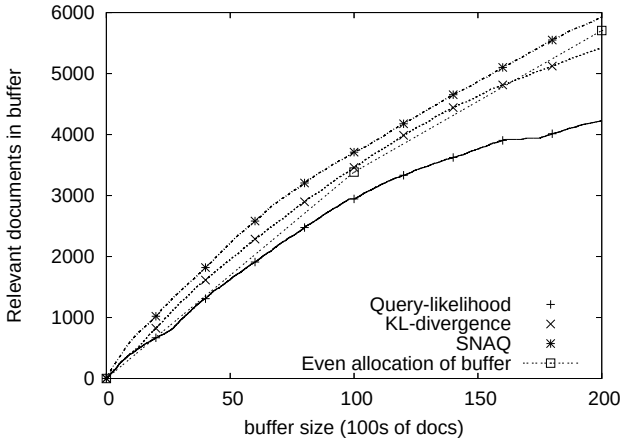

Figure 3: Zoom of Figure 2.

The even-allocation method performs poorly with small buffers, but performs better than SNAQ when the buffer size is large. The reason is that SNAQ is biased towards some queries when the buffer size is large and all queries are compared simultaneously. However, in Section 6.3, we show that the even-allocation performs the worst of all four methods in a simulation of a mobile network where peers cannot see all queries at once. The results in Section 6.3 also show that SNAQ performs best, delivering the most relevant documents in a mobile network.

5. DISTRIBUTED IR IN A DTN

The key challenge of a successful IR routing algorithm is the scoring and buffer management step listed in Section 3. We derived a method of scoring documents in Section 4; in this section, we propose a method for buffer management. In our algorithm, we rank documents and drop the lowest-

ranking documents first when the peer's buffer is overfull.

Since our goal is to maximize the number of relevant documents delivered to each peer, we have two criteria for ranking documents in a peer's buffer:

1. **Delivery:** if peer p can not deliver document D to its destination, then the document should be dropped from the buffer even if it is relevant.

2. **Relevance:** if document D is non-relevant to a query, q, then the document should be dropped from the buffer even if it is deliverable.

Accordingly, we propose the following routing metric, which combines information from both the state of the network and the relevance of documents. We rank a document D proportional to the score and deliverable estimate:

$$\frac{1}{Rank(D)} \propto \quad (1-\alpha)Deliverable(D) + \alpha Score(D) \qquad (6)$$

(Note that existing research in the networking literature sets $\alpha = 0$.)

The $Score(D)$ term in Eq. 6 is the estimation of relevance of document D that we compute in Section 4. To estimate the $Deliverable(D)$ term, we adopt the *meeting value* metric common to many network algorithms to DTNs [11, 14, 8]. This metric estimates the likelihood of a future meeting between pairs of peers given a history of meetings, and operates as follows. Each peer p maintains a meeting value for every other peer q in the network. Every time peer p meets peer q, p adds 1 to its current meeting value for q. Additionally, p adds a portion of q's meeting value for each other peer, $r \in R$, to p's current meeting value for r; where R is the set of all peers in the network. This captures the likelihood of multi-hop paths as well as one-hop paths. The meeting values degrade slowly over time, so meetings that occur more regularly result in higher values. At first, each peer initializes its meeting values to zero for all other peers.

In addition to Eq. 6, there are two more details that complete the description of our algorithm.

First, so far, we have assumed that all documents that score greater than zero from a peer's collection are offered to the next peer it meets. In fact, each peer can also *preselect* documents by offering only those documents that score higher than a specific threshold. Specifically, in Step 3 (retrieval, in Section 3), peer A sets a threshold on document score. Only retrieved documents with score higher than the threshold are added to the new document set N_A.

Second, we note that during document exchange (Step 4 in Section 3) we require that there is no overlap of documents between A's buffer and B's buffer (the node that ranks the document lower keeps the document). This saves buffer space and prevents duplicates from being delivered to the destination. Our experiments have shown this policy increases the number of relevant documents that are delivered in the network across all ranking and scoring methods.

In sum, our IR routing framework consists of two components:

1. *Document pre-selection for documents from the peer's collection:* During retrieval against a peer's collection, the peer attempts to increase precision by returning documents that score higher than some threshold. Note that the thresholding method requires that document scores are comparable between different collections and different queries. Otherwise, there is no way to set a threshold.

2. *Document ranking for the documents in the peer's buffer:* Peers rank documents based on Eq. 6 which is a combination of network information and document relevance.

6. EVALUATION

Recall that our goal is to design a distributed information retrieval service that delivers the largest number of relevant documents using a DTN. This differs greatly from existing Internet-based distributed IR retrieval algorithms, which assume all collections are always reachable in a short timescale and that the routing infrastructure can handle any number of retrieved documents. This also differs greatly from existing DTN routing algorithms, which aim only to deliver the largest number of documents.

In this section, we present the results of our simulation comparing several methods of Distrusted IR for DTNs. We perform two major comparisons. In both sets of experiments, we use the algorithm described in Section 3.

First, we compare difference scoring methods in the network simulator; in Section 4, our comparison was an upper bound on all methods as the document comparison did not occur in a distributed fashion. Our results show that our SNAQ method performs best.

Second, we evaluate the SNAQ method for various values of α from Eq. 6 and various threshold values, as discussed in the previous section. The value of α controls whether the peers use existing networking routing algorithms or take document scores into account. Our results show that while existing routing algorithms can deliver the most documents to the sources of queries, our approach of applying to SNAQ-based relevance to routing decisions in the network delivers the most relevant documents.

6.1 Evaluation Model

6.1.1 IR simulation

As we for experiments in Section 4, we used TREC123-100-bysource-callan99.v2a [1], which is a 100-collections testbed created from TREC CDs 1,2, and 3. The collections are organized by source and publication date. We use 100 queries from the title field of TREC topics 51–151. Relevance judgments for these queries are available from NIST [3].

6.1.2 Network simulation

The type of DTN we simulate is called a *small-diameter* network by the network community [11, 14, 8]. In these networks each peer, p, has a small set of preferred peers that it meets with often and fairly regularly over the course of simulation. Peer p meets with the remaining set of peers infrequently or not at all.

Since our algorithms assume no knowledge of geographic location, we simulate peer connectivity instead of peer movements. We create a link between each pair of peers in the simulation and links are activated (representing the start of a transfer opportunity) and deactivated (representing the end of a transfer opportunity) at random.

We create small-diameter networks as follows: each pair of peers in the network has a link connecting them, for each link, we draw a meeting count, at random from an exponential distribution with mean λ. For any links with $c < \lambda$, we

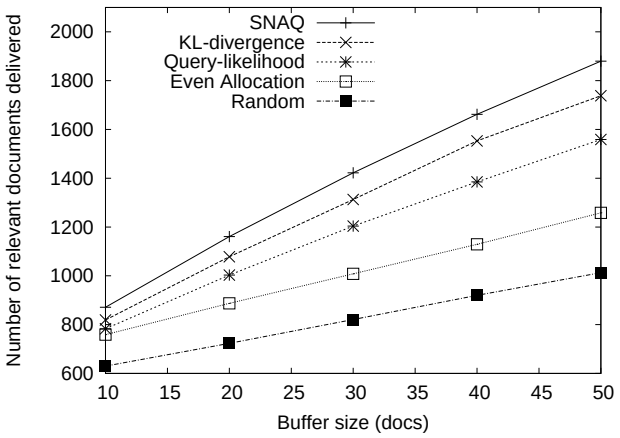

Figure 4: The number of relevant documents delivered by each normalization method as the buffer size varies (plots are slightly offset for clarity).

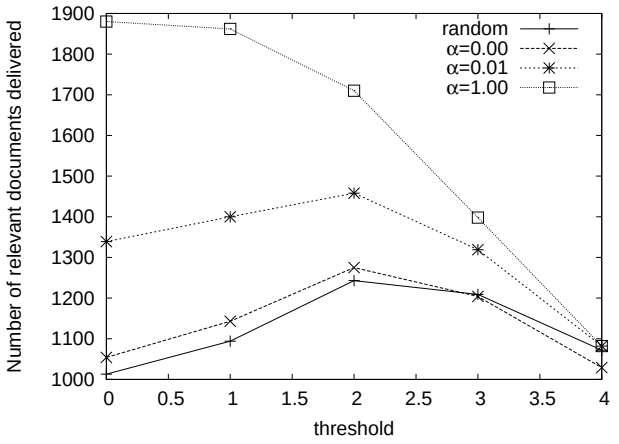

Figure 5: The number of *relevant* documents delivered by SNAQ with varied thresholds and values of α (see Eq. 6).

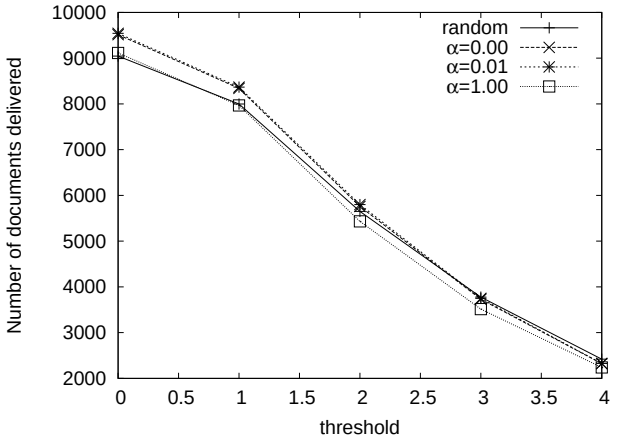

Figure 6: The *total* number of documents delivered by SNAQ with varied thresholds and values of α (see Eq. 6).

reset c=0. This creates fewer direct meetings peers and reduces the number of one-hop deliveries in the network. Let s represent the duration of the simulation. We then calculate an inter-meeting time, $i = s/c$. for each link independently. The actual times between meetings during the simulation are drawn randomly from another exponential distribution with mean of i.

We randomly pick a fixed number of peers as *query peers* that can generate queries from a query pool that consists of 100 TREC title queries. In a query peer the arrival time of a query is independently and randomly drawn from an exponential distribution with mean of t. We set $t = 10$ in the simulation.

In our experiments, there are 100 peers in the network and each peer is assigned one of the 100 collections described in 5.1. We use an average meeting count of $\lambda = 10$ along each link and let the total duration of the simulation be 200. For all experiments, we run 50 experimental trials and take the averages as the final results.

6.2 Normalization Simulation Results

Figure 4 shows that SNAQ consistently works better than the KL-divergence, query-likelihood, and even-allocation methods in the network environment. This is consistent with Figure 3, which suggested that SNAQ can normalize documents scores better than the other two methods. (In these experiments, for each method, we fixed the buffer size at 50.) Not surprisingly the random method works worst because it does not consider any relevance information at all.

The most interesting result is that the even-allocation method works poorly compared to SNAQ, KL-divergence and query-likelihood even though it performs very well in the client-server experiment shown in Figure 2. There are several reasons why this happens. The first is that we used small buffers. However, there is a more general explanation.

In the client-server architecture, the server has a global view — it can see at one time all retrieved documents from all collections for a given query. The top documents chosen for each query by the even-allocation method are exactly the top-ranked documents retrieved by a single complete collections (i.e., one that consists of all collections).

However, in the case of our small diameter network, each peer only has a local view. Locally top-ranked documents (the documents that are top ranked by local collection) are less often dropped from the buffer, taking up valuable buffer space. Since each peer is required to provide retrieval results for all available queries and each collection usually only has relevant documents for a small number of queries, some documents, even highly-ranked by a local collection, may have very low ranks in the single ranked list mentioned above. Since the even-allocation method compares only documents scores from the same query, it cannot easily judge if a locally top-ranked document would have a high rank in the single ranked list or not. SNAQ, on the other hand, is able to compare document scores among queries. Therefore, it works better because it can identify this case by comparing them with other documents for different queries.

6.3 Ranking Simulation Results

Given that SNAQ performed best in our simulations, in the remainder of this section we evaluate the affects of varying the threshold at each peer, and varying the value of α in Eq. 6. For example, for $\alpha = 0$, peer in the simulation use a buffer management technique that relies on only the like-

Score :	Percentage
0–1	: 24.2%
1–2	: 38.6%
2–3	: 21.7%
3–4	: 9.0%
> 4	: 6.4%

Table 2: Distribution of scores in retrieved documents.

lihood of delivery of documents (i.e., *deliverable(D)*). Similarly, a setting of $\alpha = 1.0$ means peers use only document scores to rank documents. A hybrid approach is $\alpha = 0.01$ where both terms of Eq. 6 are used to make buffer management decisions.

Thresholding controls the precision of documents that are sent to the network. Table 2 shows a distribution of document scores of all retrieved documents (that is, all top 10 ranked documents by each collection for each query). Setting a threshold to zero means each peer will provide all top-ranked documents to the buffers and let buffer management makes decision on which documents to keep in the buffers. Setting a positive threshold means each peer selects only these top-ranked documents with document scores above the threshold .

Our results are presented in Figure 5, which shows the total number of documents (relevant or not) returned to each peer with SNAQ for various values of α and threshold. Figure 6 shows the number of relevant documents returned to each peer for the same experiments. The curve labeled "random" denotes a method where peers randomly drop documents in overfull buffers instead of using Eq. 6 during buffer management. (Not shown, for clarity, are values of $\alpha \geq 0.1$, which performed close to but below $\alpha = 1.0$.)

The results suggest that ranking with only relevance (i.e., $\alpha = 1$) with a zero threshold provides the best result in terms of the number of relevant documents delivered. One explanation of this result is that SNAQ is a good normalization method. Another explanation is that it is easier to distinguish which of two documents is more likely to be relevant to a query than it is to distinguish which of two packets is more likely to be delivered to a destination. Figure 6 shows that the routing algorithm is not able to take strong advantage of the scenario we have placed it in. The greatest performance benefit from the routing algorithm over random is with larger buffer sizes and networks where peers have smaller cliques than we have used [11, 14, 8]. We did not evaluate many types of network connectivity patterns because our focus was on the performance of the IR. However, we believe for better routing algorithms, or different connectivity models, our hybrid method is a framework that can take advantage of the *deliverable()* estimate.

The results also suggest that setting a threshold is helpful for cases where routing is used in the ranking; e.g., both $\alpha = 0.0$ and $\alpha = 0.01$. In these two cases, $delivery(D)$ in Eq 6 dominates the buffer management, so pre-selection at the peer has a significant benefit. By retrieving and routing less documents, less packets at peers must be dropped. When a hybrid approach is required, the threshold is a significant improvement.

7. CONCLUSION

Our distributed information retrieval system is the first that operates over a disruption tolerant network, which is a mobile peer-to-peer network that covers a large geographic area. The different assumptions of a DTN pose a challenge to existing work in distributed IR. This is because retrieved results are returned hop-by-hop through the network of peers and stored in limited-size buffers. Intermediate peers on the path to the source must manage a finite buffer filled with documents from multiple queries and multiple collections.

As a solution, we proposed a method that evaluates both relevance scores and an estimate of whether the document can be delivered successfully. We showed that our scoring method, SNAQ, returns more relevant documents in our mobile network simulation than existing normalization methods, which are not intended to work across multiple queries. Additionally, we compared our approach to existing networking algorithms for delivering data in a DTN. We showed that although our method delivers less documents total, it delivers significantly more relevant documents to sources of queries. Our work shows that it is easier to distinguish likely relevance across documents than it is to distinguish a likelihood of successful routing and delivery in a DTN. However, as network routing methods improve, our ranking will be able to take advantage of the improvement and adjust its balance of scoring and delivery estimation.

8. REFERENCES

[1] http://hartford.lti.cs.cmu.edu/callan/Data.

[2] http://es.csiro.au/TRECWeb/wt10g.html.

[3] http://trec.nist.gov/data.html.

[4] J. Arnold, B.N. Levine, et al. Information Sharing in Out-of-Hospital Disaster Response: Types of information needs and information structure. *Journal of Prehospital and Disaster Medicine*, 19(3), 2004.

[5] B. Burns, O. Brock, and B.N. Levine. *MV* routing and capacity building in disruption tolerant networks. In *Proc. IEEE INFOCOM*, March 2005.

[6] J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval*, pages 127–150. Kluwer Academic Publishers, 2000.

[7] J. Davis, A. Fagg, and B. Levine. Wearable Computers and Packet Transport Mechanisms in Highly Partitioned Ad hoc Networks. In *Proc. IEEE Intl. Symposium on Wearable Computers*, pages 141–148, October 2001.

[8] M. Grossglauser and M. Vetterli. Locating nodes with ease: Mobility diffusion of last encounters in ad hoc networks. In *IEEE Infocom*, 2003.

[9] K. M. Hanna, B. N. Levine, and R. Manmatha. Mobile Distributed Information Retrieval For Highly-Partitioned Networks. In *Proc. IEEE ICNP*, pages 38–47, Nov 2003.

[10] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proc. of ACM SIGIR 2001*, pages 111–119, 2001.

[11] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, 2003.

[12] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *the Proc. of ACM SIGIR 2001*, pages 267–275, Sept 2001.

[13] J. Ponte and W.B. Croft. A Language Modeling Approach to Information Retrieval. In *SIGIR 98*, pages 275–281, 1998.

[14] N. Sarafijanovic-Djukic and M. Grossglauser. Last encounter routing under random waypoint mobility. In *Proc. NETWORKING*, May 2004.

[15] L. Si and J. Callan. Using sampled data and regression to merge search engine results. In *ACM SIGIR 2002*, pages 19–26. ACM Press, 2002.

[16] F. Song and W. Bruce Croft. A general language model for information retrieval. In *Proc. SIGIR 1999*, pages 279–280, 1999.

9. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.