Find-Similar: Similarity Browsing as a Search Tool

Mark D. Smucker and James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst

ABSTRACT

Search systems have for some time provided users with the ability to request documents similar to a given document. Interfaces provide this feature via a link or button for each document in the search results. We call this feature *find-similar* or *similarity browsing*. We examined find-similar as a search tool, like relevance feedback, for improving retrieval performance. Our investigation focused on find-similar's document-to-document similarity, the reexamination of documents during a search, and the user's browsing pattern. Find-similar with a query-biased similarity, avoiding the reexamination of documents, and a breadth-like browsing pattern achieved a 23% increase in the arithmetic mean average precision and a 66% increase in the geometric mean average precision over our baseline retrieval. This performance matched that of a more traditionally styled iterative relevance feedback technique.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation

Keywords

Find-Similar, Similarity Browsing, Relevance Feedback

1. INTRODUCTION

Relevance feedback is not a widely adopted feature of popular search services even though it is known to be a powerful tool for improving retrieval performance. The feedback-like feature that has been adopted by some services is a feature we term *find-similar* or *similarity browsing*. Find-similar allows a user to request documents that are similar to a particular document. For example, the Excite web search engine once provided this feature by adding a link to each result that read "More Like This: Click here for a list of documents like this one"[31]. Today, Google's web search engine provides a find-similar link for each item in the search results. The U.S. National Library of Medicine's PubMed search system offers find-similar as a link to "Related Articles" for each search result [26].

In our experiments, the starting point for the use of find-similar is always a results list produced by a query. Find-similar can be applied to any document listed in the results list. Like the above examples, typical instantiations of find-similar show a button or link next to each document in the results list. Clicking on find-similar produces a new results list of documents similar to the selected document. To browse a collection of documents by similarity, a user can use find-similar to jump from list to list of similar documents.

This paper focuses on find-similar's use as a search tool rather than as a browsing interface. To use find-similar as a search tool, a user will apply find-similar to a relevant document to find more relevant documents, and so forth. While feedback-like, the version of find-similar that we study has no notion of relevance. Each use of find-similar on a document is separate from uses on other documents. While the potential exists for some form of implicit relevance feedback, we examine find-similar merely as a tool that returns similar documents.

In contrast to find-similar, the traditional research interface to support relevance feedback involves showing the user the top 5 or 10 results for the query and asking the user to judge the relevance of these results. The user's feedback is then used to update the query and a new ranked list is produced. Typically the already judged documents are not shown in the new results list. Repeated use of this relevance feedback iterates through the collection of documents, and we call this *iterative relevance feedback*. In both batch and user experiments, iterative relevance feedback has been shown to be an effective means for improving search results [27, 11, 8, 15].

It appears that some users attempt to use relevance feedback systems designed for judgments on multiple documents in a manner resembling find-similar. Croft reports that users will often use a single document, which may be unrelated to the query, for relevance feedback and effectively be "browsing using feedback" [7]. Hancock-Beaulieu et al. studied 58 user sessions that used interactive query expansion (IQE) via a relevance feedback interface [10]. Of the 58 sessions, 17 used only a single document for feedback. Algorithms designed for multiple-document feedback may not always work

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

well for single document feedback. Only 3 of the 17 sessions were successful at finding additional relevant material.

We ran all of our experiments in a batch style without user involvement. We agree with White et al. [40] that simulation studies can find better ways to implement the algorithms behind interface features before investing in user studies. Simulation studies don't replace user studies.

We used two browsing patterns to evaluate various aspects of find-similar: a greedy depth-first pattern and a breadth-like pattern. At best, these browsing patterns are crude models of user behavior, but our primary use of the patterns is to demonstrate find-similar's possible effect on retrieval.

Using these browsing patterns, we examined find-similar's potential as a search tool to improve document retrieval as compared to iterative relevance feedback. An important aspect of our investigation focused on find-similar's document-to-document similarity. We also looked at the cost of having to reexamine documents while using find-similar. The browsing patterns themselves give insight into how much a user's browsing pattern can affect performance.

1.1 Related work

Wilbur and Coffee studied several aspects of find-similar [41]. They found that on average, a single relevant document used as a query does not perform as well as the original query, but that relevant documents similar to the query will do better than the query. They also used a set of browsing patterns and found that a method they called *parallel neighborhood searching* performed better than the other patterns. This method attempts to search the find-similar lists of all discovered relevant documents to the same depth. This browsing pattern is likely too complex for a user to follow. They suggested that a system could hide the complexity by showing the user one document at a time to judge, but such a system no longer supports similarity browsing or traditional lists of results.

Spink et al. have analyzed samples of Excite's query logs and reported that between 5 and 9.7 percent of the queries came from the use of the "more like this" find-similar feature [31, 32]. There is little evidence that users repeatedly used the find-similar feature to browse by feedback. Most web users look at very few results [32]. Thus, it is not surprising that find-similar found limited use by web users who are likely precision oriented.

Other systems offering similarity browsing include those of Thompson and Croft [34] and Campbell [4]. Both systems draw a display that allows the user to browse to similar documents. One of the functions of such a display is as a map to prevent users from becoming lost in their browsing. Such a display may be of use to interfaces incorporating find-similar, but find-similar does not require such a display.

One possible reason for search systems' adoption of find-similar is its appearance as an easily understood and simple to use form of relevance feedback. Of the large body of relevance feedback research [28], Aalbersberg's *incremental feedback* is an illustrative example of simplifying relevance feedback [1]. With incremental feedback, the user is shown one result at a time. To see more results, the user must judge the relevance of the presented item. In batch experiments, Aalbersberg found that incremental feedback worked better than Rocchio, Ide Regular, and Ide Dec-Hi. For these other approaches, Aalbersberg used an iteration size of 15 documents. While incremental feedback builds a model of relevant documents one document at a time, each use of find-similar involves a single document without any accumulation of documents or model of relevance.

In many systems, users can browse documents via hyperlinks. If a collection lacks hyperlinks, they can be automatically generated [2]. Find-similar effectively adds a hyperlink from a document to those most similar to it. For hypertext systems like the web, researchers have created programs to assist the user with finding relevant pages via browsing [20, 24]. In contrast to these approaches, find-similar does not observe the user to determine what the user considers relevant, and find-similar does not offer any assistance in choosing where to browse.

Another set of research has focused on helping the user better process ranked retrieval results. This work is related to but different from relevance feedback and find-similar, both of which are applied to the entire collection of documents and not restricted to the set of top ranked results. For example, Leuski [19] created a software agent to guide users in their exploration of the top results. Other approaches involve presenting the results grouped by an online clustering of the results or by predetermined categories [12, 9, 13, 5]. These approaches are different from find-similar in that while the user gets to see documents grouped by similarity, the user does not get to request more documents similar to a document.

2. METHODS AND MATERIALS

We first describe in section 2.1 how we retrieved documents for find-similar, the baseline, and an implementation of iterative relevance feedback. We then explain in section 2.2 how we created a query model for a document to which a user has applied find-similar. In sections 2.3 and 2.4 we discuss our hypothetical user interfaces and two browsing patterns used for evaluation of find-similar. We finish by describing the test collection and the evaluation methodology in sections 2.5 and 2.6.

2.1 Retrieval methods

We used both the language modeling approach to information retrieval [25] and its combination with the inference network approach [21] as implemented in the Lemur [18] and Indri [33] retrieval systems.

Language modeling represents documents and queries as probabilistic models. We used multinomials as our probabilistic models of text. For a given piece of text T, we write the probability of the word w given the model M_T of the text as $P(w|M_T)$.

The maximum likelihood estimated (MLE) model of text estimates the probability of a word as the count of that word divided by the total number of words in the text. As such, the probability of a word w given a text T is: $P(w|M_T) = T(w)/|T|$, where $T(w)$ is the count of word w in the text T and $|T| = \sum_w T(w)$ is the text's length.

For find-similar, we ranked documents using the Kullback-Leibler divergence of the query model M_Q with the document model M_D:

$$D_{KL}(M_Q||M_D) = \sum_w P(w|M_Q) \log \frac{P(w|M_Q)}{P(w|M_D)} \qquad (1)$$

where $0 \log 0 = 0$, and the query model is a model of the document to which find-similar is being applied. We detail

```
#weight(
  0.8 #combine( international organized crime )
  0.1 #combine(
    #1( organized crime )
    #1( international organized )
    #1( international organized crime ) )
  0.1 #combine(
    #uw8( organized crime )
    #uw8( international crime )
    #uw8( international organized )
    #uw12( international organized crime ) ) )
```

Figure 1: TREC topic 301, "international organized crime," converted to an Indri query by Metzler and Croft's dependence models. This query gives a weight of 0.8 to the unigram model of the topic. The ordered phrases, `#1`, have a weight of 0.1 as well as the unordered windows, `#uwN`. Not shown here is the unigram relevance model that provides a pseudo-relevance feedback component when combined with the dependence model query for our baseline run.

the two ways we constructed query models for find-similar in section 2.2.

To avoid zero probabilities and better estimate the document models, we calculated the document models using Dirichlet prior smoothing [42]: $P(w|M_D) = \frac{D(w)+mP(w|C)}{|D|+m}$, where $P(w|C)$ is the MLE model of the collection, and m is the Dirichlet prior smoothing parameter.

The inference network approach by Metzler and Croft [21] takes the probability estimates from language modeling and uses them as part of the Bayesian inference network model of Turtle and Croft [37]. The inference network provides a formal method for combination of evidence, and is easily accessed by users via a structured query language.

For our baseline, we used Metzler et al.'s method [23] that combines Metzler and Croft's [22] dependence models with Lavrenko and Croft's [17] relevance models. This method can be seen as using a precision enhancing retrieval method, dependence models, with a pseudo-relevance feedback technique, relevance models. Unlike Metzler et al., we used only the existing collection for query expansion with relevance models and did not use any external collections for expansion.

The dependence model uses the Indri query language to combine three types of evidence. The first is the standard bag-of-words unigram model as used by language modeling. The second type captures the sequential ordering of the terms in the query. The third uses the close proximity of query terms as evidence. Figure 1 shows the Indri query produced by Metzler and Croft's dependence models for TREC topic 301, "international organized crime."

To perform the baseline retrieval, first the dependence model Q of the query is run. Then a relevance model is created from the top k ranked documents. The relevance model M_R is calculated as: $P(w|M_R) = \sum_{i=1}^{k} P(D_i|Q)P(w|D_i)$, where $P(D_i|Q) = P(Q|D_i)/\sum_{j=1}^{k} P(Q|D_j)$, and $P(Q|D_i)$ is the Indri belief that document model D_i is relevant to the query Q. Finally, the dependence model and the relevance model are combined to create the final baseline query using Indri's #weight operator.

Parameter	Value
Dirichlet smoothing for unigram terms, m	1500
Dirichlet smoothing for ordered and unordered windows, m	2000
Weight of unigram model in dependence model	0.8
Weight of ordered windows model in dependence model	0.1
Weight of unordered windows model in dependence model	0.1
Number of pseudo feedback documents for relevance model	10
Weight of dependence model when mixed with pseudo relevance model	0.3
Max. terms in pseudo feedback relevance model	25
Max. terms in find-similar document models	50
Max. terms in iterative feedback relevance model	50
Weight of initial query when mixed with iterative feedback relevance model	0.3

Table 1: Retrieval parameters.

The baseline is also used as the initial retrieval for both find-similar and iterative relevance feedback.

Our implementation of iterative relevance feedback is akin to that used by Rocchio [27]. We mix in a model of the relevant documents with the original baseline query model using Indri's #weight operator. We tried weights of 0.0, 0.3, 0.5, and 0.7 for the original query and found 0.3 to work best. The model of relevant documents is calculated as: $P(w|M_R) = \frac{1}{k}\sum_{i=1}^{k} P(w|D_i)$, where k is the number of documents the user has judged to be relevant. An alternative is for us to replace the pseudo feedback component of the baseline query model with the real relevance model as provided by the user's judgments, but we have not yet investigated this variant.

We used the same parameter settings that Metzler et al. derived from training on the TREC 2004 Robust track data and that they used for the 2005 Robust track [23]. The 2004 Robust track includes the same 150 topics we used for evaluation (topics 301-450) in its 250 topics. Table 1 shows the retrieval parameters' settings for all runs. We used the same smoothing parameters for all experiments.

2.2 Document-to-document similarity

An obvious way to implement find-similar for documents is to treat the document as a very long query. A problem with this approach is that each document will often be about several topics of which only one is the user's search topic. A document may well be about "organized crime" but it may also be about the prosecution of criminals. Not all stories about criminal prosecution are about organized crime. Rather than finding documents that are similar to all the topics mentioned in a story, we think a user will want to find documents that are similar with respect to the current search topic.

We examined two types of similarity for find-similar: *regular* and *query-biased*. Regular similarity treats the document as a query to find other similar documents. Query-biased similarity aims to find similar documents given the context of the user's search and avoid extraneous topics. For both regular and query-biased similarity, we construct a unigram model of the find-similar document that is then used as a

query to find similar documents (see equation 1). Regular similarity uses the maximum likelihood estimated (MLE) model of the document as the query. For query-biased similarity, we create a MLE model of the document text that consists of all words within a certain distance W of all query terms in the document. For our experiments, we set W to 5. Thus the 5 preceding words, the query term, and the 5 words following a query term are used. Should a document not contain any query terms, the whole document is used. For both types of similarity, we truncate the document model to the 50 most probable terms.

Our notion of query-biased similarity is more akin to query-biased summaries [36, 29] than to query-biased clustering [9, 13] or query sensitive similarity [35]. The nature of query-biased summaries is to extract the sentences or text surrounding query terms in a document and use this extracted text as a summary of the document. In contrast to query-biased summaries, both Eguchi [9] and Iwayama [13] increase the weight of query terms in the documents before clustering. Tombros' query sensitive similarity modifies the cosine similarity measure to place more weight on the query terms [35]. Preliminary experiments where we linearly combined the query model with the document model as a form of query-biasing showed poorer performance. We hypothesize that this poorer performance was the result of a lack of diversity in the find-similar lists.

2.3 Hypothetical user interfaces

We ran all of our experiments in a batch style without user involvement. Assumptions about the interface affect the batch evaluation of retrieval features. In particular, we only consider browsing patterns that could be reasonably executed by a user with our hypothetical user interface. We next describe our hypothetical user interfaces for find-similar and iterative relevance feedback.

The find-similar interface we envision is similar to the web-based PubMed search system [26]. Our hypothetical interface has "back button" support like a web browser. If a user has performed find-similar on a document, the user can decide to stop examining the documents presented as similar to that document and hit the back button. The back button returns the user to the previous list at the position in the list where the user had applied find-similar.

Results are presented in rank order with query-biased summaries for both the initial query and the find-similar lists. Sanderson [29] demonstrated that users are able to judge the relevance of documents from query-biased summaries with 75% of the accuracy of full documents. Thus, we assume users will examine most documents by reading the already visible summaries. Reading a summary requires no manipulation of the interface and therefore provides no feedback to the system that the document has been examined. When a user applies find-similar to a document, the user will be presented with a new page listing the similar documents. The find-similar lists will contain some documents that the user has already examined on previous pages. The user will have to reexamine documents unless there is a visual marker to designate already examined documents.

In our evaluation, we compared two conceptual variations of our imagined find-similar interface. In one variation non-relevant documents are reexamined and in the other they are not. Both variations prevent the reexamination of relevant documents.

The hypothetical iterative relevance feedback interface displays the top N documents of the ranked results. The user judges each of the displayed documents and then submits the feedback to retrieve the next N documents. In our experiments, we set the iteration size N to 10. The previously displayed documents are not shown again for the current topic. This process repeats until 1000 documents have been examined. This interface does not provide for use of a back button like find-similar. The system only allows forward iteration.

2.4 Find-similar browsing patterns

Klöckner et al. [14] used an eye tracker to observe how people processed search results. They used a Google results list containing 25 results for a query. The subjects' task was to find the relevant documents in the list. Subjects could click on a result to see the result's web page. Of the subjects, 65% followed a depth-first strategy. These users examined the documents in order, from highest to lowest rank, and did not look ahead. Another 15% used a breadth-first strategy by looking at the entire results list before opening a single web page. The remaining 20% used a partial breadth-first strategy by sometimes looking ahead at a few results before opening a web page.

Given the user behavior observed by Klöckner et al., we used two browsing patterns to evaluate find-similar. The *greedy* pattern represents the depth-first behavior, and the *breadth-like* pattern aims to capture the breadth-first search behaviors. Neither pattern is a true depth-first or breadth-first search pattern. A true depth-first pattern does not reflect that a user is likely to stop examining a results list if no relevant documents are found. A true breadth-first pattern is not feasible for a user to implement. While inspired by the user behavior observed by Klöckner et al., these patterns are at best crude models of user behavior. Users could execute these patterns, but we have little knowledge of how users actually search with find-similar. Instead, these patterns give us insight into the potential of find-similar and the degree to which find-similar's performance can be affected by different browsing patterns. Both patterns use the baseline as the initial retrieval.

The greedy browser examines documents in the order that they appear in a results list. As section 2.3 explained, the browser will only examine a relevant document once. When a relevant document is examined, the greedy browser performs a find-similar operation on this document. The greedy browser ceases to examine documents in a results list after examining 5 contiguous non-relevant documents. When the browser stops examining a list, the browser hits the "back button" and returns to the previous list and continues examining documents in that list. If the browser is examining the initially retrieved list of documents, the only stopping criterion is that the browser stops when 1000 documents have been examined.

The breadth-like browser also examines documents in the order that they appear in a results list. What differs from the greedy pattern is that the breadth-like browser only begins to browse via find-similar when the results list's quality becomes too poor. As the breadth-like browser examines relevant documents, it places these documents in a first-in first-out queue local to the current list. When the precision at N, where N is the rank of the current document, drops below 0.5 or when 5 contiguous non-relevant documents are

encountered, the browser applies find-similar to the first relevant document in the queue. When the browser returns to the current list, it applies find-similar to the next document in the queue until the queue is empty. The browser never uses find-similar on a relevant document more than once. Thus documents in the queue will be ignored if the browser has already performed find-similar on them. There is not any notion that the breadth-like browser knows which relevant documents are the best for find-similar. The breadth-like browser merely delays exploration until the current list seems to have gone "cold." The browser stops examining a results list in the same manner and with the same criteria, i.e. 5 contiguous non-relevant documents, as the greedy browser.

Early experiments with a greedy browsing pattern influenced our design of the breadth-like browser. We saw that the greedy browser could degrade the performance of an already good retrieval. Thus, the breadth-like browser uses list quality as its criteria for delaying use of find-similar. While the breadth-like browsing pattern could be seen as a "corrected" greedy pattern, we feel that it does capture the goal of a breadth-first user, that is, to look ahead before acting.

2.5 Queries, documents, and retrieval tools

The topics used for the experiments consisted of TREC topics 301-450, which are the ad-hoc topics for TREC 6, 7, and 8. TREC topics consist of a short title, a sentence length description, and a paragraph sized narrative. The titles best approximate a short keyword query, and we used them as our queries.

We used TREC volumes 4 and 5 minus the Congressional Record for our collection. This 1.85 GB, heterogeneous collection contains 528,155 documents from the Financial Times Limited, the Federal Register, the Foreign Broadcast Information Service, and the Los Angeles Times.

We used the Lemur toolkit [18] for all of our experiments including its Indri subsystem [33]. In particular, we generated the results for the find-similar runs using a Lemur index of the collection with stop words removed at index time. For the baseline and iterative relevance feedback runs we used an Indri index with stop word removal at query time. We stemmed all words with the Krovetz stemmer [16]. We used an in-house stopword list of 418 noise words.

2.6 Evaluation methodology

We constructed our runs' results lists for evaluation in the same manner as Aalbersberg [1]. The results lists that we evaluated represent the order in which the simulated user examines the documents. For the baseline retrieval, the documents are examined in rank order. For find-similar, the browsing patterns of section 2.4 determine the order in which documents are examined. For iterative relevance feedback, documents are examined in the same manner they are judged — one iteration of 10 documents at a time.

All relevance judgments are made using the "true" relevance judgments per NIST. We treat a reexamined non-relevant document the same as any other non-relevant document found at that position in the results. All of the retrieval techniques we studied do not reexamine relevant documents.

We report metrics using both the arithmetic mean and the geometric mean. The TREC Robust track has established the geometric mean as a useful tool for analyzing performance [38]. As opposed to the usual arithmetic mean, the geometric mean emphasizes the lower performing topics. The arithmetic mean can hide large changes in performance on poorly performing topics with small changes in the better performing topics. As with the 2005 TREC Robust track [39], for computing the geometric mean, we set values less than 0.00001 to 0.00001 to avoid zeros. We used version 8 of trec_eval [3] to compute per topic metrics. We measured statistical significance with a two-sided, paired, randomization test with $100,000$ samples (see page 168 [6]). Unless otherwise stated, significance is at the $p < 0.05$ level.

3. RESULTS

Table 2 shows the arithmetic mean, non-interpolated, average precision (AMAP) and the geometric mean (GMAP) across the 150 topics of TREC 6, 7, and 8, for the baseline, find-similar, and iterative relevance feedback runs. The find-similar runs vary based on whether or not non-relevant documents were reexamined (section 2.3), whether a greedy or breadth-like browsing pattern was used (section 2.4), and whether the similarity was regular or query-biased (section 2.2).

In general, find-similar and iterative relevance feedback are better able to improve on a poor initial retrieval than on a good initial retrieval. To highlight this behavior, Table 2 also reports results for the 150 topics divided into three sets of 50 topics. The topics are ordered by their performance on the baseline and then divided into three sets (like quartiles except into thirds instead of quarters). These sets are roughly equivalent to poor, fair, and good retrieval performance with baseline AMAPs of 0.036, 0.202, and 0.548 respectively. With the topics divided up in this manner, the geometric mean adds little insight and we report only the arithmetic mean of each topic set.

The average precision results are based on the TREC standard of 1000 results. To understand the performance when a user examines fewer documents, Table 3 shows the precision at 20 and 100 documents. Feedback techniques can increase recall as well as precision. Table 4 shows the recall at 1000 documents.

4. DISCUSSION

The best find-similar run avoids reexamining non-relevant documents, follows a breadth-like browsing pattern, and uses query-biased similarity. Table 2 shows that this run matches the performance of our implementation of iterative relevance feedback and achieves a 23% improvement in the arithmetic mean average precision (AMAP) and a 66% improvement in the geometric mean average precision (GMAP) over the baseline. Iterative relevance feedback achieves a 69% improvement in GMAP, but this is not a statistically significant difference compared to the best find-similar run.

The use of a high quality baseline retrieval is required to avoid overstating the performance gains possible with a retrieval technique. We used the method developed by Metzler et al. [23] for our baseline (see section 2.1). This method had the best title run as measured by mean average precision and had the second best geometric mean average precision for both title and description runs submitted to TREC's 2005 Robust track [39]. We achieved larger relative performance improvements during initial experiments with a weaker baseline.

		Reexamine non-relevant				Do not reexamine non-relevant				Iter.
		Greedy		Breadth-like		Greedy		Breadth-like		Rel.
	Baseline	Regular	Biased	Regular	Biased	Regular	Biased	Regular	Biased	FB.
All 150 topics										
AM Avg. Prec.	0.262	0.175*	0.226*	0.260	0.269	0.224*	0.281	0.303*	0.323*	0.322*
Pct. Change		-33%	-14%	-1%	3%	-14%	7%	16%	23%	23%
GM Avg. Prec.	0.130	0.122	0.151*	0.157*	0.169*	0.160*	0.193*	0.197*	0.216*	0.220*
Pct. Change		-6%	16%	21%	30%	23%	49%	52%	66%	69%
Baseline's 50 poorest performing topics										
AM Avg. Prec.	0.036	0.079*	0.091*	0.083*	0.101*	0.108*	0.119*	0.114*	0.134*	0.129*
Pct. Change		119%	151%	130%	179%	197%	228%	215%	270%	255%
Baseline's 50 middle performing topics										
AM Avg. Prec.	0.202	0.160*	0.190	0.190	0.196	0.218	0.261*	0.251*	0.275*	0.256*
Pct. Change		-21%	-6%	-6%	-3%	8%	29%	24%	36%	27%
Baseline's 50 best performing topics										
AM Avg. Prec.	0.548	0.285*	0.396*	0.505*	0.509*	0.346*	0.461*	0.544	0.560	0.580*
Pct. Change		-48%	-28%	-8%	-7%	-37%	-16%	-1%	2%	6%

Table 2: Arithmetic mean (AM) and geometric mean (GM) average precision for all 150 topics and the arithmetic mean average precision for the 150 topics grouped into three disjoint sets based on the baseline's average precision for that topic. Results with a $\star$ are different from the baseline at a statistically significant level ($p < 0.05$) as measured by a two-sided, paired, randomization test with 100,000 samples.

		Reexamine non-relevant				Do not reexamine non-relevant				Iter.
		Greedy		Breadth-like		Greedy		Breadth-like		Rel.
	Baseline	Regular	Biased	Regular	Biased	Regular	Biased	Regular	Biased	FB.
Precision at 20 documents										
Arith. Mean	0.374	0.254*	0.330*	0.372	0.387	0.282*	0.358	0.395*	0.415*	0.411*
Pct. Change		-32%	-12%	-1%	3%	-25%	-4%	6%	11%	10%
Geo. Mean	0.120	0.095*	0.116	0.121	0.130	0.104*	0.128	0.132*	0.143*	0.137*
Pct. Change		-21%	-3%	0%	8%	-13%	6%	9%	19%	14%
Precision at 100 documents										
Arith. Mean	0.225	0.163*	0.206	0.219	0.236	0.204*	0.246	0.250*	0.274*	0.277*
Pct. Change		-28%	-8%	-3%	5%	-10%	9%	11%	22%	23%
Geo. Mean	0.122	0.106*	0.128	0.125	0.137*	0.129	0.152*	0.145*	0.163*	0.162*
Pct. Change		-13%	5%	3%	12%	6%	25%	19%	34%	33%

Table 3: Arithmetic mean and geometric mean of the precision at 20 and 100 documents. Results with a $\star$ are different from the baseline at a statistically significant level ($p < 0.05$) as measured by a two-sided, paired, randomization test with 100,000 samples.

		Reexamine non-relevant				Do not reexamine non-relevant				Iter.
		Greedy		Breadth-like		Greedy		Breadth-like		Rel.
	Baseline	Regular	Biased	Regular	Biased	Regular	Biased	Regular	Biased	FB.
Arith. Mean	0.687	0.741*	0.750*	0.747*	0.746*	0.806*	0.809*	0.808*	0.811*	0.823*
Pct. Change		8%	9%	9%	9%	17%	18%	18%	18%	20%
Geo. Mean	0.603	0.688*	0.703*	0.695*	0.700*	0.763*	0.765*	0.764*	0.767*	0.779*
Pct. Change		14%	17%	15%	16%	26%	27%	27%	27%	29%

Table 4: Arithmetic mean and geometric mean of the recall at 1000 documents. Results with a $\star$ are different from the baseline at a statistically significant level ($p < 0.05$) as measured by a two-sided, paired, randomization test with 100,000 samples.

We also tested iterative relevance feedback with an iteration size of 1, which is Aalbersberg's incremental feedback [1]. An iteration size of 1 performed as well as an iteration size of 10, with the larger iteration size yielding a negligibly larger AMAP (0.322 vs. 0.321).

All the find-similar runs that avoid reexamination of non-relevant documents perform better than the corresponding runs that do reexamine non-relevant documents. An interface that supports find-similar may need to provide a mechanism to help the user avoid reexamination of non-relevant documents. If a user has to keep track of judgments, it would seem that find-similar and traditional multiple item relevance feedback should be able to co-exist in the same retrieval system.

While both the greedy and breadth-like browsing patterns show significant improvements in GMAP over the baseline, following a breadth-like browsing pattern is superior to the greedy browsing pattern. Table 2 shows that the greedy browsing pattern in particular has difficulty with the better performing topics. As section 2.4 noted, the work by Klöckner et al. [14] motivated the two browsing patterns we used, but the performance of the greedy pattern influenced our design of the breadth-like browser. A user that follows a greedy browsing pattern will be harmed by the find-similar feature on better performing topics. The breadth-like browsing pattern avoids using find-similar while the retrieval quality of a list is high. We leave for future work the question of whether find-similar can be used to improve an already high quality retrieval.

Query-biased similarity shows consistently better performance than regular similarity. The query-biased similarity helps the greedy browsing pattern perform over 20% better than with regular similarity as measured by AMAP and GMAP on all 150 topics. Query-biased similarity also helps the breadth-like browser but to a lesser degree.

Given a search topic, a perfect document-to-document similarity method for find-similar makes the topic's relevant documents most similar to each other. We can characterize this notion of relevant documents being more similar to each other by measuring the distance from all relevant documents to all other relevant documents. For each topic, we constructed a directed graph as follows. Each relevant document is a node in the graph. There is an edge from each node to each other node. The weight of an out edge is the rank of the target document in the ranked list produced by applying find-similar to the source document. Given this graph, we compute the all pairs shortest paths (APSP) to obtain the shortest distance from every node to every other node. This distance is the number of documents that need to be examined to navigate from one relevant document to another using find-similar. We used the Boost Graph Library's implementation of the Floyd-Warshall APSP algorithm [30]. The distribution of distances is highly skewed. The distance to some documents is so large that they are "out of reach" via find-similar. A single topic can greatly skew the average, too. Therefore we use the median rather than the mean to handle these skewed distributions. The median of the median distances is 70.8 for regular and 33.0 for query-biased similarity. It appears that query-biased similarity creates a tighter grouping of relevant documents than does regular similarity. This result, that query-biased similarity better clusters relevant documents, echoes the results of Tombros' work on query sensitive similarity [35].

The find-similar and feedback runs show a much greater improvement in GMAP than in AMAP. Table 2 highlights this difference and shows that the majority of the improvement comes from improving the poorer performing topics. For the poorest performing topics, the baseline has an AMAP of 0.036, and on average, 1 document in 28 is relevant. On these same topics, find-similar raises this ratio to 1 in 7 with an AMAP of 0.134. Besides having a large relative performance improvement for poorly performing topics, find-similar can provide performance gains that should be noticeable by the end user.

Being able to improve precision early in a ranked list may influence user adoption of a retrieval tool such as find-similar. Table 3 shows that find-similar can achieve improvements over the baseline in precision at 20 and 100 documents. While not shown, the best find-similar run also obtained a statistically significant 7% increase in P@10 (arithmetic mean) over the baseline.

For find-similar's best run, its P@100 arithmetic mean improvement of 22% is comparable to its AMAP improvement of 23%. For this same run, the P@100 geometric mean improvement of 34% is nearly half that of the 66% improvement in GMAP. A fair amount of the GMAP performance may come from improving very poorly performing topics with feedback on low ranking relevant documents. For some poor performing topics, if users are unwilling to dig deep into the ranked results, they may be unable to use feedback to help their search.

Table 4 shows that all of the find-similar runs increase recall at 1000 documents and the best performance is comparable to iterative relevance feedback. Retrieval techniques that cluster or reorder the top N results cannot increase the recall at N [13, 19]. Interestingly, the different similarity and browsing types do not significantly impact recall at 1000 documents.

5. CONCLUSION

We found that find-similar, as a feedback-like search tool, has the potential to improve document retrieval. The best performance improvement attained by find-similar matched that of an implementation of iterative relevance feedback. Find-similar achieved a 23% improvement in the arithmetic mean average precision and a 66% improvement in the geometric mean average precision. The geometric mean emphasizes the poorer performing topics.

We found differences in performance for find-similar along the dimensions of document-to-document similarity, reexamination of documents, and the browsing pattern. First, we discovered that a query-biased similarity performs significantly better than using a document alone as a query for find-similar. We demonstrated the greater clustering power of query-biased similarity using an all pairs shortest path analysis that we believe is novel. Secondly, interfaces supporting find-similar as a search tool will likely need to help the user avoid reexamining already examined documents. Finally, a user's browsing pattern can substantially affect the performance of find-similar. Between two simulated browsing patterns, we found that a breadth-first like pattern works better than a greedy, depth-first like pattern. Both patterns show significant improvement in the geometric mean average precision over a strong baseline retrieval.

Given the potential of find-similar, future work should include user studies to determine if users can obtain simi-

lar improvements. While one could create more elaborate browsing patterns, our preference would be to implement a find-similar interface and study users. Future work should also examine in greater detail the many ways of computing document-to-document similarity. Analyzing the ability of similarity methods to cluster relevant documents could continue to be done with batch style experiments.

6. ACKNOWLEDGMENTS

Thanks to Don Metzler for providing his dependence model code and guidance on the use of dependence models. Thanks to Trevor Strohman, Hema Raghavan, and the anonymous reviewers for their helpful feedback.

This work was supported in part by the Center for Intelligent Information Retrieval and in part by the Defense Advanced Research Projects Agency (DARPA) under contract number HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] I. J. Aalbersberg. Incremental relevance feedback. In *SIGIR*, pages 11–22, 1992.

[2] J. Allan. *Automatic Hypertext Construction*. PhD thesis, Cornell University, 1995.

[3] C. Buckley. trec_eval. `http://trec.nist.gov/trec_eval/trec_eval.8.0.tar.gz`.

[4] I. Campbell. *The ostensive model of developing information needs*. PhD thesis, University of Glasgow, 2000.

[5] H. Chen and S. Dumais. Bringing order to the web: automatically categorizing search results. In *CHI*, pages 145–152, 2000.

[6] P. R. Cohen. *Empirical methods for artificial intelligence*. MIT Press, 1995.

[7] W. B. Croft. What do people want from information retrieval? *D-Lib Magazine*, Nov. 1995.

[8] S. T. Dumais and D. G. Schmitt. Iterative searching in an online database. In *Proc. Human Factors Soc. 35th Annual Mtg.*, pages 398–402, 1991.

[9] K. Eguchi. Adaptive cluster-based browsing using incrementally expanded queries and its effects (poster abstract). In *SIGIR*, pages 265–266, 1999.

[10] M. Hancock-Beaulieu, M. Fieldhouse, and T. Do. An evaluation of interactive query expansion in an online library catalogue with a graphical user interface. *Journal of Documentation*, 51(3):225–243, 1995.

[11] D. Harman. Relevance feedback revisited. In *SIGIR*, pages 1–10, 1992.

[12] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *SIGIR*, pages 76–84, 1996.

[13] M. Iwayama. Relevance feedback with a small number of relevance judgments: incremental relevance feedback vs. document clustering. In *SIGIR*, pages 10–16, 2000.

[14] K. Klöckner, N. Wirschum, and A. Jameson. Depth- and breadth-first processing of search result lists. In *CHI*, page 1539, 2004.

[15] J. Koenemann and N. J. Belkin. A case for interaction: a study of interactive information retrieval behavior and effectiveness. In *CHI*, pages 205–212, 1996.

[16] R. Krovetz. Viewing morphology as an inference process. In *SIGIR*, pages 191–202, 1993.

[17] V. Lavrenko and W. B. Croft. Relevance based language models. In *SIGIR*, pages 120–127, 2001.

[18] Lemur Toolkit for Language Modeling and IR. `http://www.lemurproject.org/`.

[19] A. Leuski. Relevance and reinforcement in interactive browsing. In *CIKM*, pages 119–126, 2000.

[20] H. Lieberman. Letizia: An agent that assists web browsing. In *IJCAI-95*, pages 924–929, 1995.

[21] D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. *IPM*, 40(5):735–750, 2004.

[22] D. Metzler and W. B. Croft. A markov random field model for term dependencies. In *SIGIR*, pages 472–479, 2005.

[23] D. Metzler, F. Diaz, T. Strohman, and W. B. Croft. UMass robust 2005 notebook: Using mixtures of relevance models for query expansion. In *TREC 2005 Notebook*, 2005.

[24] C. Olston and E. H. Chi. Scenttrails: Integrating browsing and searching on the web. *ACM Trans. Comput.-Hum. Interact.*, 10(3):177–197, 2003.

[25] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR*, pages 275–281, 1998.

[26] Pubmed, `www.pubmed.gov`. "Related articles": `www.nlm.nih.gov/bsd/pubmed_tutorial/m5002.html`.

[27] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System*, pages 313–323. Prentice Hall, 1971.

[28] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 18(2):99–145, 2003.

[29] M. Sanderson. Accurate user directed summarization from existing tools. In *CIKM*, pages 45–51, 1998.

[30] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost Graph Library*. Addison Wesley, 2001.

[31] A. Spink, B. J. Jansen, and H. C. Ozmultu. Use of query reformulation and relevance feedback by excite users. *Internet Research: Electronic Networking Applications and Policy*, 10(4):317–328, 2000.

[32] A. Spink, D. Wolfram, B. J. Jansen, and T. Saracevic. Searching the web: The public and their queries. *JASIST*, 52(3):226–234, 2001.

[33] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language-model based search engine for complex queries (extended version). Technical Report IR-407, CIIR, UMass, 2005.

[34] R. H. Thompson and W. B. Croft. Support for browsing in an intelligent text retrieval system. *Int. J. Man-Machine Studies*, 30:639–668, 1989.

[35] A. Tombros. *The effectiveness of query-based hierarchic clustering of documents for information retrieval*. PhD thesis, University of Glasgow, 2002.

[36] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *SIGIR*, pages 2–10, New York, NY, USA, 1998. ACM Press.

[37] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *TOIS*, 9(3):187–222, 1991.

[38] E. M. Voorhees. The trec robust retrieval track. *SIGIR Forum*, 39(1):11–20, 2005.

[39] E. M. Voorhees and H. T. Dang. Draft: Overview of the TREC 2005 robust retrieval track. In *TREC 2005 Notebook*, pages 105–112, 2005.

[40] R. W. White, J. M. Jose, C. J. van Rijsbergen, and I. Ruthven. A simulated study of implicit feedback models. In *ECIR*, 2004.

[41] W. J. Wilbur and L. Coffee. The effectiveness of document neighboring in search enhancement. *IPM*, 30(2):253–266, 1994.

[42] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *SIGIR*, pages 334–342, 2001.