
User Modeling and User-Adapted Interaction **14:** 259–288, 2004.

Interactive Information Retrieval Using Clustering and Spatial Proximity

ANTON LEUSKI[1]⋆ and JAMES ALLAN[2]
[1]*University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Suite 1001 Marina del Rey, CA, 90292, USA*
[2]*Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003 USA*

(Received: 12 March 2002, accepted in revised form 27 November 2002)

Abstract. A web-based search engine responds to a user's query with a list of documents. This list can be viewed as the engine's model of the user's idea of relevance—the engine 'believes' that the first document is the most likely to be relevant, the second is slightly less likely, and so on. We extend this idea to an interactive setting where the system accepts the user's feedback and adjusts its relevance model. We develop three specific models that are integrated as part of a system we call Lighthouse. The models incorporate document clustering and a spring-embedding visualization of inter-document similarity. We show that if a searcher were to use Lighthouse in ways consistent with the model, the expected effectiveness improves—i.e., the relevant documents are found more quickly in comparison to existing methods.

Key words. clustering, information organization, information retrieval, information visualization

1. Introduction

The bulk of research in Information Retrieval (IR) explores techniques for improving the ability of an automatic system to return relevant documents in response to a searcher's query. The work has typically been carried out in laboratory batch evaluations, with static queries, collections, and relevance judgments. Researchers have made enormous strides in understanding how to handle queries automatically, and in a wide range of IR-related tasks. Perhaps the most visible collections of such research are the annual Text REtrieval Conference (TREC) (Harman and Voorhees, 1999; 2000; 2001).

We are interested in extensions of these ideas to interactive environments. Although it is very important that the underlying search technology be as accurate as possible, how the information is presented and how the user interacts with a system are also critical. The TREC workshops have incorporated an investigation into interactive retrieval every year (Allan et al., 1998; Hersh and Over, 2001), though results from those evaluations have not always been clear.

⋆Most of the work on this article was carried out while the author worked at the Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts at Amherst.

All approaches to IR develop some sort of a model of what documents are likely to be relevant to the user's query. When we move from a static environment to an interactive setting, the models should be richer in that they ought to incorporate more information about the user's interaction with the system. In this study we focus on extending the model of the user's idea of relevance—a user relevance model—to adapt to the user's interaction as it occurs, capturing the user's intents better than a classic static model.

We will define the notion of user relevance models and show how they are related to traditional ranked list and relevance feedback techniques. We will develop three specific models that are integrated as part of a system we call Lighthouse. The models incorporate document clustering and a spring-embedding visualization of inter-document similarity. We will show that if a searcher were to use Lighthouse in ways consistent with the model, the expected effectiveness will improve—i.e., the relevant documents will be found more quickly in comparison to existing methods. We discuss how Lighthouse has been implemented to incorporate the best of these models and to help (though not require) a searcher to follow that model.

We start in Section 2 by defining the notion of a user relevance model (URM) and then develop the specific instances in Section 3. In Sections 4 and 5 we describe our evaluation paradigm and the laboratory evaluation that we use to show the improvements from our models. In Section 6 we evaluate one of the URMs that is based on document clustering and show that our technique matches classic relevance feedback in effectiveness, while providing better information to the searcher about what is happening. In Section 7 we evaluate two URMs based on inter-document similarity and show that one of them is significantly better than relevance feedback—as well as providing a more transparent system for the searcher. We conclude the work in Section 8.

2. User Relevance Model

Bookstein (1983) argues that information retrieval should be envisioned as a process, in which the user is examining the retrieved documents in sequence and the system can and should gather feedback to adjust the retrieval. We adopt a similar notion while looking at organizing the documents retrieved by a search engine in response to a user's query. The document organization is a process that induces an order on the retrieved documents and the users are expected to follow that order while examining the results.

This document ordering reflects the system's model of what the user considers relevant information. It is expected that the user will find the documents that are located at the beginning of the ordering more relevant than the documents that follow them. We call the system's representation of the user's desires (made visible by the document ordering) the *user relevance model* (URM).

A URM can be viewed as a mapping between each unexamined document d and a numeric value: $F(\mathcal{D}_t, d)$. Here $\mathcal{D}_t$ is the current state of the document set and

t is the current time step. The current state of the document set is the basis of the URM and includes all information about the search: the original query, all retrieved documents, whether the documents were examined by the user, and what relevance label if any were assigned to the documents by the searcher. The document with the highest values of $F(\mathcal{D}_t, d)$ is expected to be the most relevant and it should be examined first. We call this mapping function the *user relevance model function*. Thus, a URM application is straightforward – at every time step, a system that employs the URM points the user to the document with highest value of $F(\mathcal{D}_t, d)$.

Note that a URM is a 'dynamic' entity and the values for the URM's function $F(\mathcal{D}_t, d)$ change as soon as there is a change in the state of the document set $\mathcal{D}_t$. For example, suppose the user judges a document to be relevant to the request. She labels the document as relevant, modifying the document set representation $\mathcal{D}_t \rightarrow \mathcal{D}_{t+1}$. If the URM function takes into account the document relevance labels, it creates a new set of values of $F(\mathcal{D}_{t+1}, d)$ – the URM *adapts*.

One example of a URM is the ranked list of documents returned by the search engine. In the ranked list the documents are ordered by their probability of being relevant: the user is expected to start at the top of the ranked list and proceed down the list examining the documents one-by-one. We call this URM the *ranked list user relevance model*. The URM function is equivalent to the query-document similarity function that is used by the search engine to rank the documents in the first place:

$$F(\mathcal{D}_t, d) \equiv \text{Rank}(Q, d), \quad \forall t$$

where Q is the user's query. The URM function is independent of the current state of the retrieved document set, so there is no adaptation in the model. This is an example of a *static* user relevance model.

The interactive query-based relevance feedback approach (Aalbersberg, 1992) defines another document ordering: the documents are ordered by probability of being relevant. The user is supposed to start from the top and examine the documents until the first relevant document is found. That document is used to modify the query, the unexamined documents are reordered by probability of being relevant to the new query, and the process continues. We call this *dynamic* ordering the *relevance feedback user relevance model*.

The most widely used approach to relevance feedback is the Rocchio method (Rocchio, 1971). Given the original query and a set of judged documents, a new query is constructed after a new relevance judgment is made. The weight of each term in the new query is a weighted sum of the weights for that term in the old query, in the known relevant and known non-relevant documents:

$$\boldsymbol{Q}_t = \alpha \cdot \boldsymbol{Q} + \beta \cdot \frac{1}{|\mathcal{R}_t|} \sum_{\forall x \in \mathcal{R}_t} \boldsymbol{x} + \gamma \cdot \frac{1}{|\mathcal{N}_t|} \sum_{\forall x \in \mathcal{N}_t} \boldsymbol{x} \tag{1}$$

Here $\boldsymbol{Q}$ and $\boldsymbol{Q}_t$ are the vectors of terms for the old and new queries, where each vector element corresponds to a term in the document set vocabulary; $\mathcal{R}_t$ and $\mathcal{N}_t$ are the sets of all known relevant and non-relevant documents at time t, and $\boldsymbol{x}$ is the vector

of terms for document x. Parameters α, β, and γ – called *Rocchio coefficients* – control the relative impact of each component.

The URM function for the relevance feedback user relevance model is defined as follows:

$$F(\mathcal{D}_t, d) \equiv \text{Rank}(Q_t, d)$$

where the content of the query Q_t is determined by Equation 1.

The idea of the user relevance model function is very similar to the concept of the document ranking function that constitutes the essence of most search engines. The main difference is that the URM function is 'dynamic': the value $F(\mathcal{D}_t, d)$, and therefore the current ordering of documents, may depend on what documents were examined and what relevance judgments were assigned by the time t.

Both ranked list and relevance feedback are well-known approaches in information retrieval. The ranked list approach is widely-used by most of the academic (Allan et al., 1997) and commercial (e.g., http://www.google.com/) search engines. It is intuitive, clearly-understood by searchers, and has been shown to perform well on multiple occasions (Harman and Voorhees, 1997; 1998; 1999). However, it does not support any system-user interaction beyond creating the initial query.

Traditional knowledge-intensive techniques for adaptive interactive systems generally rely on individual user models learned by observing user's behavior and they only become useful after substantial amount of training (Cypher, 1991). Other approaches require initial encoding of domain and user knowledge that is costly, difficult to acquire, and can be rarely extended on a different domain (Benyon and Murray 1993; Sullivan and Tyler, 1991). Adaptive information access methods often rely on specialized retrieval structures such as semantic indexing (Frisse and Cousins, 1989), neural networks (Chen, 1995), and relevance networks (Chen and Mathe, 1995). However, none of these techniques has been shown effective in TREC experimental settings. Large collections of heterogeneous information and a set of very diverse information requests create a formidable challenge for adaptive techniques.

Our system requires less sophisticated knowledge representation and it is very similar to the traditional relevance feedback approach in IR (Salton, 1989). Relevance feedback methods improve the retrieval effectiveness by modifying the original query using the user's relevance judgments. Our system is based on ideas refined by the most successful relevance feedback implementations (Allan et al., 1998; Buckley and Salton, 1995).

Relevance feedback has been shown to perform well in both batch-oriented (Buckley and Salton, 1995) and incremental settings (Aalbersberg, 1992; Allan, 1995). However, existing anecdotal evidence supported by experimental studies suggests that casual searchers find relevance feedback confusing and unpredictable (Koenemann and Belkin, 1996). Because of its closed and difficult to explain mechanics relevance feedback still struggles to find it acceptance in every day search tasks.

In this paper we consider three alternative URMs. Each of these URMs stores a representation for the state of the document set. Each defines a function that ranks the unexamined documents in the order they should be considered by the searcher. They all accept user's relevance judgments and adapt their document set representation to take advantage of the user feedback. Each model is strongly related to classic relevance feedback, but is made less opaque for the user by tightly pairing it with an interface tool that is designed to visualize the current state of the model. In the next section we describe the systems and define the URMs.

3. Defining three URMs

In this paper we focus on the task of helping a user to locate relevant documents among the retrieved results. The Cluster Hypothesis of Information Retrieval states that documents that are similar to each other tend to be relevant to the same query (van Rijsbergen, 1979). A corollary of this hypothesis is that if we find one relevant document, some of the other relevant documents will be similar to it. Our approaches use the inter-document similarity information to help the searcher.

3.1. DOCUMENT CLUSTERING URM

The first system – the *clustering* system – partitions the retrieved document set into groups of similar documents. Figure 1 shows the top 50 documents retrieved by the Google search engine in response to the query 'Samuel Adams'. The documents

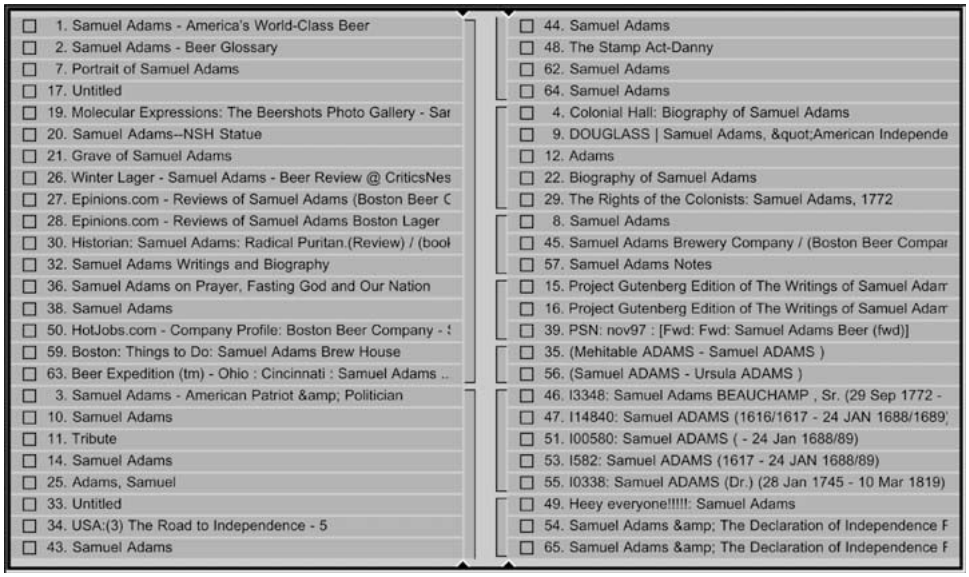

Figure 1. Clustering visualization of 50 documents for the 'Samuel Adams' query.

are shown as a list of titles. Each document title is preceded with its original rank assigned by the search engine. Note that the last document in the set has the rank of 65. Our system visualizes only the web pages it can access at the time of the query. When our system fails to download a retrieved page in the time allowed, it tries to access the next document in the ranked list. In our example the system failed to access 15 web pages in the top portion of the list returned by the Google engine.

On the picture the list is divided into two columns, it flows starting in the left and continues in the right column. The document titles are arranged into clusters and the clusters are indicated by the bars or 'handles' that appear in the gutter between the columns. For example, the first 17 documents in the left column form the first cluster. There are 8 clusters on the picture. The documents inside each cluster are ordered by their rank and the clusters themselves are ordered by the rank of the highest ranked document. We call this representation the *clustered list* by analogy with the ranked list. Several past studies adopted similar approaches (Hearst and Pedersen, 1996; Leuski and Croft, 1996).

By analogy with the ranked list we assume that a user will start at the top of the clustered list and follow it down examining document after document. The only difference is that we assume that the user stops examining a cluster and switches to another cluster if she finds too many non-relevant document. When she reaches the end of the clustered list, the searcher returns back to the top of clustered list and continues this process until all retrieved documents are examined. We call the resulting document ordering the *clustering user relevance model*.

We define the clustering URM function as follows:

$$F_{\mathrm{cl}}(\mathcal{D}_t, d) = \theta_1 \cdot \sum_{x \in \mathcal{R}_t} \mathrm{sim}_{\mathrm{cl}}(x, d) + \theta_2 \cdot \sum_{x \in \mathcal{N}_t} \mathrm{sim}_{\mathrm{cl}}(x, d)$$

where $\mathcal{R}_t$ and $\mathcal{N}_t$ are the sets of all examined relevant and non-relevant documents at time step t and $\mathrm{sim}_{\mathrm{cl}}(x, d)$ is the binary similarity between two documents:

$$\mathrm{sim}_{\mathrm{cl}}(x, d) = \begin{cases} 1, & \text{if } x \text{ and } d \text{ are in the same cluster} \\ 0, & \text{otherwise} \end{cases}$$

If two documents have the same score $F_{\mathrm{cl}}(\mathcal{D}_t, d)$, the clustering URM prefers the document placed higher in the clustered list. The parameters θ_1 and θ_2 determine when the user should switch from one cluster to another.

For example, suppose $\theta_1 = 1$ and $\theta_2 = -1$, then at the beginning all documents have the same score ($F_{\mathrm{cl}}(\mathcal{D}_t, d) = 0$). The URM suggests that the user examines the top document in the top cluster. If the document is relevant, then all other documents in that cluster receive a score of 1, and the URM will suggest the next document from the same top cluster. That will continue until there are no more documents in the cluster or the number of examined non-relevant documents in the cluster exceeds the number of examined relevant document, i.e. all remaining documents in that cluster will have a negative score. In both cases, the URM will suggest the top document in the second ranked cluster.

At this point, it is worth mentioning, that the URM we have defined is by no means the only URM that could use document clustering. A URM could randomly select a cluster (so they would have an expected ranking), could rank clusters by their average similarity to the query, could require that a searcher examine a complete cluster before moving on, etc. Our intent is to show how those models can be evasluated so that their parameters can be optimized, and so that preferable models can be found.

There are four experimental questions we consider in relation to the document clustering system and the URM we have defined:

1. What clustering algorithm should we use to partition the documents?
2. How should documents be arranged in the clustered list? We select one representative document from each cluster and place it at the top of the cluster's list. This document is supposed to be the most helpful to the user in establishing the overall relevance value for the cluster. Ideally, by looking at the representative document the user should be able to decide whether to examine the cluster or skip it and go to the next one. The rest of the documents inside the cluster are kept in their original order – they are ordered by the probability of being relevant to the user's request. This should insure an effective ranking (van Rijsbergen, 1979, p. 88). The clusters are ordered using the original rank of the highest ranked document in each cluster.
3. When should the user stop examining a cluster and switch to the next cluster in the list? What will be the best values for the weights θ_1 and θ_2?
4. Is the clustered URM a better model of the user's concept of relevance than the ranked list URM? Will a system that uses this model be more effective in locating relevant information in the retrieved document set than the ranked list or the interactive relevance feedback?

We address each of these questions in more detail in Section 6 on clustering experiments.

A problem with clustering is that it requires a hard decision about which cluster a document should be assigned to. In the next section we change the presentation so that clusters are not explicitly created. Instead, a visualization portrays document relationships in such a way that tight clusters are easily recognizable and that relationships between and within clusters are also visible.

3.2. SPRING-EMBEDDING VISUALIZATION

In the clustering system described in Section 3.1 the inter-document similarity is a binary function – two documents are either similar or not, they are either in the same cluster or not. Our second document organization system relies on a more accurate and continuous representation of the similarity. Figure 2 shows the documents visualized as spheres floating in space and positioned in proportion to inter-document

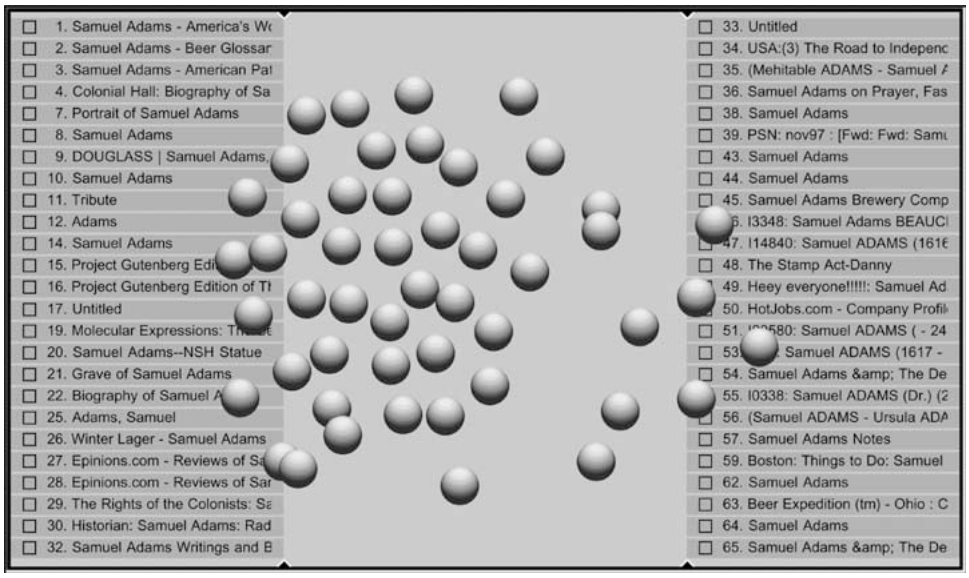

Figure 2. Spring-embedding visualization of 50 documents for the 'Samuel Adams' query in two dimensions.

similarities. Documents with spheres located close together have a similar content and a pair of spheres placed far apart corresponds to a pair of very different documents.

To arrange the document spheres in the visualization space we use a multidimensional scaling (MDS) algorithm called *spring-embedding*. An MDS algorithm accepts a matrix of inter-object dissimilarities and attempts to create a set of points in a Euclidean space such that the distances between the points correspond to the dissimilarities between original objects as closely as possible. A number of such algorithms exist. Our choice was motivated by the graph-drawing heritage of spring-embedding (Fruchterman and Reingold 1991; Swan and Allan, 1998) – it is supposed to generate eye-pleasing pictures – and the availability of the source code.

The spring-embedding algorithm models each document as an object in 2- or 3-dimensional visualization space. The objects repel each other with a constant force. They are connected with springs and the strength of each spring is proportional to the similarity between the corresponding documents. This 'mechanical' model begins from a random arrangement of objects and due to existing tension forces in the springs, oscillates until it reaches a state with 'minimum energy' – when the constraints imposed on the object placements by the springs are considered to be the most satisfied. The result of the algorithm is a set of points in space, where each point represents a document and the inter-point distances are proportional to the inter-document dissimilarities.

We extend the idea of using inter-document similarity to design a URM for the spring-embedding visualization. There is a significant difference between the clustering and spring-embedding visualizations that we have to take into account

while designing the URM. The ranked list, interactive feedback, and clustering URMs have a well-defined starting point: the top of the ranked list. What is the starting point for someone navigating the spring-embedding visualization? We address this problem by allowing the URM to start from the first relevant document in the ranked list. We consider this document and all the other documents that precede it in the list as examined and the relevance judgments available to the model. This simulates a user who started looking for relevant information in the ranked list and switched to the spring-embedding visualization after she found the first relevant document.

In this paper we consider two different URMs for the spring-embedding visualization that take into account the distances between the document representations. Other models are of course possible (Leuski, 2000).

3.2.1. *Relevance Proximity URM*

The first URM ranks the documents by their average proximity to the known relevant documents. Initially that distance is equivalent to the distance between the document and the top ranked relevant document in the ranked list as per the starting condition. As the user examines the documents, more relevant documents get discovered and labeled. The URM is adjusted by modifying the sum:

$$F_{\text{rel}}(\mathcal{D}_t, d) = -\frac{1}{|\mathcal{R}_t|} \cdot \sum_{x \in \mathcal{R}_t} \text{dist}(x, d)$$

where $F_{\text{rel}}(\mathcal{D}_t, \text{d})$ is the URM function, $\mathcal{D}_t$ is a representation of the current state of the document set that includes the relevance judgments about the examined documents at time step t, d is an unexamined document, $\text{dist}(x, d)$ is the distance or dissimilarity between two documents, $\mathcal{R}_t$ is the set of all examined relevant documents at time step t. We call this model the *relevance proximity* URM.

Note that the relevance proximity URM may behave differently depending on whether $\text{dist}(x, d)$ is the distance between two document vectors or it is the distance between two spheres in the visualization space. Generally, it is impossible to find a set of objects in the visualization space such that the inter-sphere distances are precisely equal to the distances between the corresponding document vectors. When the documents are visualized with the spring-embedding algorithm some of the documents may be shown nearby when they are actually unrelated because of the constraints imposed by using fewer dimensions.

It is likely that the URM that uses distances between document vectors – which are more accurate – will be more effective than the one that uses inter-sphere distances. However, the document closest to the known relevant documents in the document vector space may not appear so in the visualization space. The document ranking made by the former URM may appear on the screen as breaking the intuition behind the relevance proximity URM. Thus the URM that uses inter-sphere distances may prove to be more intuitive to the users.

3.2.2. *Wizard URM*

The design of the second URM builds on the idea of the traditional Rocchio relevance feedback scheme (Equation 1). By analogy with the Rocchio approach our URM ranks the unexamined documents using a weighted sum of the document similarity to the query, average similarity between the document and all examined relevant documents, and average similarity between the document and all examined non-relevant documents:

$$\begin{aligned} F_{\mathrm{Roc}}(\mathcal{D}_t, d) = \theta_0 &+ \theta_1 \cdot \mathrm{querysim}(d) + \\ &+ \theta_2 \cdot \frac{1}{|\mathcal{R}_t|} \sum_{x \in \mathcal{R}_t} \mathrm{sim}(x, d) + \\ &+ \theta_3 \cdot \frac{1}{|\mathcal{N}_t|} \sum_{x \in \mathcal{N}_t} \mathrm{sim}(x, d) \end{aligned}$$

where $\mathrm{querysim}(d)$ is the similarity between the document and the original query, $\mathrm{sim}(x, d) = 1/\mathrm{dist}(x, d)$ is the similarity between two documents. Note that F_{rel} is equivalent to F_{Roc} with $\theta_0 = \theta_1 = \theta_3 = 0$ and $\theta_2 = 1$.

We go beyond the Rocchio approach by making an heuristic observation that different arguments in the Rocchio-based representation should be weighted depending on how many documents were examined. For example, the similarity of a document to the query is more important at the beginning of the search, when few relevant documents are known, than further into the process when the document relevance information is plentiful. We accommodate this intuition by dividing the search process into three distinct phases and considering a separate Rocchio-alike URM function for each phase:

$$F_{\mathrm{wzd}}(\mathcal{D}_t, d) = \sum_i A_i(|\mathcal{R}_t| + |\mathcal{N}_t|) \cdot F_{\mathrm{Roc},i}(\mathcal{D}_t, d)$$

where the coefficients $A_i(\cdot)$ – called *application coefficients* in machine learning – are smooth functions of the number of the examined documents (Berliner, 1979):

$$A_i(x) = \exp\left(-\frac{(x - \mu_i)^2}{\sigma^2}\right)$$

where μ_i defines the center of the influence area for the ith 'subfunction' $F_{\mathrm{Roc},i}(\mathcal{D}_t, d)$ and σ defines its width. We call this model the *wizard* URM.

We set the centers for application coefficients ($A(\cdot)$) of each area to the beginning, middle, and end of the ordering. The width was set to one quarter of the distance between the centers. For example, for 50 documents $\mu = 1, 25, 50$ and $\sigma = 6$. The parameter vectors for $F_{\mathrm{Roc,i}}(\mathcal{D}_t, \mathrm{d})$ ($\theta_{i,\cdot}$) were determined during a preliminary training phase described elsewhere (Leuski, 2000).

There are two experimental questions we consider in relation to the spring-embedding visualization, relevance proximity, and wizard URMs:

1. Both the ranked list and interactive relevance feedback URMs provide the baseline for our experiments. How do the relevance proximity and wizard URMs compare to the ranked list and relevance feedback URMs? Can a system that uses either of these URMs help a searcher find the relevant documents faster than our baselines?
2. The inter-document similarity does not necessarily have metric properties (we discuss the document similarity function in Section 5.1) and it is not possible to map it precisely onto inter-sphere distances in the visualization. Some of the document spheres may be shown nearby when the corresponding documents are unrelated because of the constraints imposed by the visualization space. Our intuition is that a higher dimensional visualization will provide more degrees of freedom and therefore it has a better chance to represent the inter-document relationships accurately than a lower dimensional one. How does the dimensionality of the visualization affect the quality of the URM?

We address each of these questions in more detail in Section 7. In the next section we discuss our evaluation strategy and provide some additional motivation for both the clustering and the spring-embedding systems.

3.3. SUMMARY

All three URMs discussed in this section – the clustering, relevance proximity, and wizard models – can be used without the corresponding clustering and spring-embedding interface tools. For example, one can envision a system that presents the retrieved documents to the searcher one at a time. It requires the searcher to make a relevance judgment about each document and uses the URM to select the next document for presentation. Such a scenario has the same problem as the traditional query-based relevance feedback – it is completely opaque and confusing to the user.

The main advantage of our approach is that each URM is coupled with an interface that visualizes all the information used by the model. For example, the relevance proximity URM ranks documents by their average similarity to the known relevant documents. The spring-embedding visualization clearly displays that information by mapping the similarity onto inter-sphere distances on the screen. Suppose the system color-codes the individual document spheres using green for relevant and red for non-relevant documents. The inner workings of the URM are now transparent – 'the next document recommended by the system is the one closest to that group of green spheres.' Thus the interface serves an important goal in the system-user interaction – it explains and motivates the choices made by the system.

A similar approach is possible for the query-based relevance feedback. We can create a system that will display each new term the relevance feedback adds to the query and we can display the relative weight of each term in the individual documents in the attempt to capture and present all the information used by the Rocchio formula (Koenemann and Belkin, 1996). However, we believe that an interface dealing with

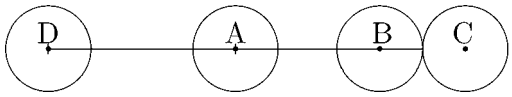

Figure 3. Example of how an implicit version of relevance proximity URM will adapt. The interval that goes through the centers of spheres A and D indicates that point D and the point where B and C touch are equidistant from A.

dozens or even hundreds of individual terms can be no less confusing than an interface that does not explain anything at all.

A second, more ambitious goal of the visualization is to eliminate the need for the searcher to assign relevance judgments to individual documents. So far we have assumed that the searcher will always assign relevance judgments. The document organization system defines and actively maintains an *explicit* form of the user relevance model, i.e. there is a ranking function with well-defined numerical parameters encoded in the system. At the same time all the information used by the URM is presented on the screen. If the user successfully recognizes and applies the inter-document similarity data from the picture, she can browse the retrieved set as effectively as if she were guided by the URM. Thus the visualization can help the searcher to define and maintain an *implicit* version of the relevance model herself, i.e, the knowledge of where to look for the next relevant document. The model is adjusted every time the user examines a document. For example, consider Figure 3. Suppose document A is examined and it is relevant. The user examines document B, because it is the closest document to A. If B is relevant, then the user will examine C as it is the closest document to both A and B. Otherwise, the user will examine D as the next closest document to A alone.

We consider the question of how well a searcher can recognize and interpret the informational clues displayed by the spring-embedding visualization system in Section 7.2.

4. Evaluating a URM

To evaluate a URM we imagine a searcher who always follows the URM's advice – every time she examines the document with the highest rank assigned by the URM function. Given a standard set of document collections with queries and relevance judgments, we simulate such a user interacting with individual systems that employ the URMs discussed in this article. Specifically, this simulation works as follows:

1. Using a search engine, run a query on the collection and select the top 50 retrieved documents.
2. Rank all unexamined documents using the URM function $F(\mathcal{D}_t, d)$.
3. Pick a document with the highest value $F(\mathcal{D}_t, d)$, mark it as examined, and record it.

4. Assign a relevance label to the document using the relevance judgments from the experimental dataset.
5. If there are no more unexamined documents quit, otherwise go to step 2 to select the next document.

The result of each simulation is an ordering of the documents. We compare multiple document ordering using *average precision* (van Rijsbergen, 1979). To compute the average precision we examine the document ordering starting at the beginning and following it until the end. Every time there is a relevant document in the ordering we take a note of the precision at that position, i.e. the proportion of relevant documents among those we have seen. At the end of this process we compute the average of the recorded precision values – the average precision. It determines how quickly the searcher can locate all relevant documents while following the given ordering. Here 'speed' is measured in the number of examined documents. In the best possible ordering all relevant documents are located before any non-relevant ones. Such an ordering has average precision equal to one. The average precision of the worst possible ordering with all relevant documents placed after all non-relevant ones is

$$P_{\text{worst}} = \frac{1}{|\mathcal{R}|} \sum_{i=1}^{|\mathcal{R}|} \frac{i}{i + |\mathcal{N}|}$$

5. Experimental Setup

For our experiments we used the data provided by the Text REtrieval Conference (TREC) (Harman and Voorhees, 1997). Specifically, we used the ad-hoc queries with their corresponding collections and relevance judgments supplied by NIST assessors. We converted TREC topics 251–300 and 301–350 into queries and ran them against the documents in TREC volumes 2 and 4 (2.1 GB) and TREC volumes 4 and 5 (2.2 GB) respectively. To run the queries we used INQUERY – a search engine developed at the University of Massachusetts at Amherst (Allan et al., 1998). For each TREC topic we considered four types of queries:

1. a query constructed by extensive analysis and expansion (Allan et al., 1997);
2. the description field of the topic;
3. the title of the topic;
4. a query constructed from the title by expanding it using Local Context Analysis (LCA) (Xu and Croft, 1996). A query of this type has size and complexity between the corresponding queries of the first and second types.

Different queries constructed for the same topic result in different retrieved sets. That allowed us to study what effect both the quality of the query and its size have on the information organization. The complete discussion of this effect can be found elsewhere (Leuski, 2001b).

For each query we selected the top 50 documents returned by the search engine. There were 400 document sets with 50 documents each. These formed our main experimental data set.

The ranked list was generated by the INQUERY system during the retrieval process (Allan et al., 1998). We implemented the relevance feedback URM following the strategy outlined in Section 2:

$$\boldsymbol{Q}_t = \alpha \cdot \boldsymbol{Q} + \beta \cdot \frac{1}{|\mathcal{R}_t|} \sum_{\forall x \in \mathcal{R}_t} \boldsymbol{x} + \gamma \cdot \frac{1}{|\mathcal{N}_t|} \sum_{\forall x \in \mathcal{N}_t} \boldsymbol{x} \tag{2}$$

The Rocchio coefficients were set to the default values of the INQUERY feedback subsystem ($\alpha = 0.5$, $\beta = 4$, and $\gamma = -1$). We also limited the number of terms that are added to the original query $\boldsymbol{Q}$. Our experiments showed that performance of the relevance feedback URM can be improved if we use only the highest ranked 100 terms to expand the query.

5.1. DOCUMENT REPRESENTATION

The INQUERY system's retrieval model neither incorporates the notion of similarity between documents nor assumes the construction of document representations. To compute inter-document similarities we employ the vector-space model for document representation (Salton, 1989) – each document is defined as vector V, where $v_i = w_i$ is the weight in this document of the i-th term in the vocabulary. The term weight is computed following the INQUERY weighting formula (Allan et al., 1998), which combines Okapi's *tf* score (Robertson et al., 1995) and INQUERY's normalized *idf* score to compute the weight:

$$v_i = \frac{tf}{tf + 0.5 + 1.5\dfrac{doclen}{avgdoclen}} \cdot \frac{\log\left(\dfrac{colsize + 0.5}{docf}\right)}{\log(colsize + 1)}$$

where *tf* is the number of times the term occurs in the document, *docf* is the number of documents the term occurs in, *doclen* is the number of terms in the document, *avgdoclen* is the average number of terms per document in the collection, and *colsize* is the number of documents in the collection. The similarity between a pair of documents is measured by the cosine of the angle between the corresponding vectors (Salton, 1989). In this paper we use one over the cosine ($1/\cos\theta$) to define the dissimilarity (or the *distance*) between a pair of documents.

6. Clustering Experiments

In this section we consider the following four experimental questions introduced in Section 3.1 based on the clustering URM:

1. What clustering algorithm should be selected?
2. How should documents be arranged in the clustered list? What document has to be the cluster's representative (the top document in the cluster)?
3. When should the user stop examining a cluster and switch to the next cluster in the list?
4. Is the clustered list a more effective presentation for locating relevant information in the retrieved document set than the ranked list or the interactive relevance feedback?

6.1. CLUSTERING ALGORITHM

The first experimental question in this section is the comparison of different clustering algorithms. Our system uses a hierarchical agglomerative clustering algorithm to cluster the documents. Such an algorithm creates a hierarchy of clusters – it builds a tree where each node is a cluster of objects and the clusters corresponding to the node's immediate children form a complete partition of that cluster (Mirkin, 1996). On input the algorithm receives a set of objects and a matrix of inter-object distances. It starts by assigning each object to its own unique cluster – the leaves of the future tree. The algorithm iterates through the cluster set by selecting the closest pair of clusters and merging them together forming a new cluster that replaces them in the cluster set. A node corresponding to this new cluster is created in the tree and the selected pair of clusters become its children. That procedure is executed until all objects are contained within a single cluster, which becomes the root of the tree.

This is a general algorithm that is instantiated by choosing a specific distance function for clusters. Indeed, the distance between a pair of singleton clusters is well-defined by the original distance matrix. If one of the clusters contains more than one object, the inter-cluster distance is determined by a specific heuristic. For example, the *single linkage* method defines the distance between two clusters as the smallest distance between two objects in both clusters. The *group average* on the other hand defines the distance between two clusters as the average of distances between the cluster members.

Lance and Williams (1967) have shown that many different clustering methods can be derived from the following equation and computed quite efficiently:

$$d_{k,i\cup j} = \alpha_i \cdot d_{k,i} + \alpha_j \cdot d_{k,j} + \beta \cdot d_{i,j} + \gamma \cdot |d_{k,i} - d_{k,j}| \tag{3}$$

here, $d_{k,i\cup j}$, the distance between the cluster created by merging ith and jth clusters and an arbitrary cluster k is defined as a nonlinear function of distances between the individual clusters. The coefficients for the most commonly used methods are presented in Table I.

In this study we consider six different clustering techniques based on the generalized agglomerative algorithm (Table I): centroid, complete linkage, group average, single linkage, Ward, and weighted group average.

Table I. Lance-Williams coefficients for some agglomerative clustering methods. n_i denotes the size of the ith cluster.

Method	α_i	α_i	β	γ
Centroid	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	$\frac{-n_i \cdot n_j}{(n_i + n_j)^2}$	0
Complete linkage	0.5	0.5	0	0.5
Group average	$\frac{n_i}{n_i + n_j}$	$\frac{n_j}{n_i + n_j}$	0	0
Single linkage	0.5	0.5	0	−0.5
Ward	$\frac{n_i + n_k}{(n_i + n_j + n_k)}$	$\frac{n_j + n_k}{(n_i + n_j + n_k)}$	$\frac{-n_k}{(n_i + n_j + n_k)}$	0
Weighted group average	0.5	0.5	0	0

The hierarchical agglomerative clustering algorithm produces a hierarchy of clusters. We are interested in an approach which presents the user with a partition of the document set – a set of clusters that divides the retrieved material into the groups of similar documents.

To create a partition of the document set from a cluster hierarchy we 'cut' the hierarchy at some level, i.e., stop the clustering algorithm before it reaches the root of the tree. The clusters present in the set at that moment form the required partition. The problem is to decide at what point to make the cut. For example, the Scatter/Gather research (Hearst and Pedersen, 1996) fixes the number of clusters in the document set. We, on the other hand, set a threshold on the similarity distance between clusters – while iterating through the cluster set the algorithm stops as soon as the distance between the closest pair of clusters exceeds the threshold. If the threshold is kept constant from session to session, the density of the clusters becomes the system's invariant. The user will always know what minimal degree of similarity to expect from the documents placed in the same cluster.

To select the threshold value we conducted our experiments following a basic two-way cross-validation scheme. We divided our experimental data in half: training and testing data. We selected the threshold using the data from the former data set and evaluate the performance on the rest of the data. We then repeated the experiments switching the roles of the data sets (Leuski, 2001a). We set the clustering URM coefficients θ_1 and θ_2 to 1 and −1, respectively (see Section 3.1). The documents were arranged in the clustered list and the documents in each cluster were ordered by their original rank. The highest ranked document in each cluster was selected as the cluster's representative.

As outlined in Section 4, we implemented our system using one of the clustering algorithms. We then follow the advice given by the clustering URM (i.e., stay with a cluster as long as it has sufficient number of relevant documents) to generate a 'ranking' and calculate its average precision. We do the same with the other clustering algorithms to compute their effectiveness.

We observed that the group average and Ward algorithms result in the best performance (see Table II). The difference between these two algorithms is not statistically

Table II. Performance (percent average precision) of the clustering URM on document see partitions created by six different clustering algorithms.

Single link	Compl. link	Group avrg.	Weight. avrg.	Centr.	Ward
45.90	47.24	47.39	46.73	46.34	47.28

significant*. Also the differences between the group average algorithm and both weighted average and complete linkage are not statistically significant. The single linkage method is a clear 'loser' in this competition.

6.2. PRESENTING CLUSTERS

We consider four different alternatives for selecting the representative document. The first, a rather obvious choice is to use the document that is the best representation of the cluster – the cluster centroid, or the document that is the most similar to the actual centroid. The second alternative we consider is the highest ranked document in the cluster. Our intuition is that if this document is non-relevant than the rest of the cluster is very likely non-relevant. The third choice is to use the lowest ranked document: if that document is relevant than it is very likely that the rest of the cluster is also relevant.

The documents at the top of the ranked list most likely are relevant and the documents at the bottom of the list most likely are non-relevant. Lewis (1992) speculated that the best way to find the boundary between the relevant and non-relevant material in the list is to examine the documents in the middle. The last candidate for the cluster representative is the medium ranked document – the document whose original rank is the median of the cluster.

The choice of the highest ranked document in a cluster as the cluster's representative is the most effective for the task of locating the relevant material (Table III). For this experiment we used the group average clustering algorithm and the clustering URM coefficients θ_1 and θ_2 were set to 1 and -1, respectively. We observe an almost 4% drop in average precision while selecting the document closest to the cluster's center. The difference is statistically significant. Our explanation is that the highest relevant document allows the user to quickly discard non-relevant clusters – if even the highest

Table III. Performance (percent average precision) of the clustering URM on document set partitions created by the group average algorithm using four different types of cluster representative documents.

Centroid	Highest ranked	Lowest ranked	Medium ranked
45.76	47.39	43.78	44.70

*In the clustering experiments we used the paired two-tailed t-test to measure the statistical significance. The cutoff level is set to 5% ($p < 0.05$).

ranked document in the cluster is non-relevant, then it is very likely that the rest of the cluster is also non-relevant.

6.3. STOPPING CRITERIA

Recall from Section 3.1 that the clustering URM function $F_{\text{cl}}(\mathcal{D}_t, d)$ has two parameters θ_1 and θ_2. The ratio of these values determines when the searcher should switch from examining one cluster to another. For example, if $\theta_1 = 1$ and $\theta_2 = -1$ the values of $F_{\text{cl}}(\mathcal{D}_t, d)$ for unexamined documents in the cluster will drop below zero as soon as the user sees more non-relevant than relevant documents in the cluster. The documents from the next cluster in the list will have a higher score – it is time to start looking at the next cluster in the list. Note that if the first document in a cluster is non-relevant, the user should go immediately to the next cluster in the list.

We clustered the documents using the group average clustering algorithm and used the highest ranked document in each cluster as the cluster's representative.

The six columns in Table IV correspond to six different value settings for the relevant and non-relevant weights (θ_1 and θ_2) in F_{cl}. The column titles are presented in the form θ_1:θ_2. There is a clear maximum in performance of the total average for $\theta_1 = 1$ and $\theta_2 = -1$. Increasing or decreasing the value for the non-relevant weight θ_2 from -1 leads to lower values of average precision. All pairwise differences between the maximum ('1:−1') and the other parameter sets are statistically significant.

6.4. CLUSTERING VS. RANKED LIST

The last experimental question in this section is to compare the performance of the clustering URM F_{cl} with the performance of the ranked list and relevance feedback

Table IV. Performance (percent average precision) of the clustering URM on document set partitions created by the group average algorithm. Average precision values are shown for different ratios of relevant and non-relevant weights (θ_1 and θ_2) in $F_{\text{cl}}(\mathcal{D}_t, d)$.

Relevant to non-relevant weight ratio					
1:0	1:−0.2	1:−0.5	1:−1	1:−2	1:−4
43.22	46.97	47.18	47.39	47.14	47.09

Table V. Performance (percent average precision) of F_{cl} search strategy on document set partitions created by the group average algorithm. The table also includes the precision values for the ranked list and interactive relevance feedback search strategies. The interactive relevance feedback used top 100 terms for query expansion.

RL original	RF	Group	Average v. RL
41.98	47.17	47.39	+13.61%

URMs. The last two columns of Table V show the average precision values for F_{cl} and the percentage difference from the ranked list ('RL'). The search strategy worked with clustering structures built by the group average algorithm. The highest ranked document was used as the cluster representative. The search strategy function weights were set to $\theta_1 = 1$ and $\theta_2 = -1$.

We observe 13.61% improvement over the ranked list. This difference is statistically significant. The difference between F_{cl} and the search strategy for interactive relevance feedback ('RF') is small and not statistical significant.

6.5. CONCLUSIONS

In this part of our study, we have shown that we can substantially improve the speed at which relevant documents are found, if document clusters are incorporated into the system's model of how to predict relevance. We have developed a model for evaluating several strategies a searcher might follow when trying to find relevant documents in a clustered set (i.e., where to start and when to switch to a different cluster). Our results indicate that some approaches are better than others, but that most are better than a simple ranked list.

7. Spring-embedding Experiments

Our second set of experiments in this paper deals with the spring-embedding visualization system and with relevance proximity and wizard URMs.

1. How do the relevance proximity and wizard URMs compare to the ranked list and relevance feedback URMs? Can a system that uses either of these URMs be more successful in helping a searcher to find the relevant documents than our baselines?
2. How does the dimensionality of the visualization affect the quality of the URM?

7.1. RESULTS

We evaluated the performance of both relevance proximity and wizard URMs on the same data sets we used for the clustering experiments. Table VI summarizes the results of our experiments. Note that in Section 6 our simulations began their exploration from the top of the ranked list. Here the simulation process started from the highest

Table VI. Performance (average precision) of the relevance proximity 'F_{rel}' and wizard 'F_{wzd}' URMs in the original vector space and spring-embedded visualization space. We did not run the wizard URM in the visualization space because it designed to work in the original vector space.

Dimension	RL	RF	$F_{rel}(\mathcal{D}, d)$	$F_{wzd}(\mathcal{D}, d)$
Vector space	39.12	49.29	48.23	53.82
3D			47.86	–
2D			46.54	–

ranked relevant document (as if the searcher started in the ranked list and then jumped to the visualization after discovering the first relevant document).

The results confirm our intuition that a higher dimensional visualization will provide more degrees of freedom and therefore it has a better chance to represent the inter-document relationships accurately than a lower dimensional one. We observe a small drop in average precision when the relevance proximity URM is moved from the high-dimensional document space into a smaller number of dimensions. The drop in precision is less when three dimensions are used (−0.8%) instead of two (−3.5%) and it is only statistically significant for the difference between the URM effectiveness in the original document vector space and in the two-dimensional visualization space.

There is a small improvement (2.8%) if the search strategy is moved from the two-dimensional embedding space to the three-dimensional. This difference is statistically significant by two-tailed t-test with $p < 0.05$.

The wizard URM is a clear winner in these experiments. It shows 37.6% and 9.2% improvement over the ranked list and interactive relevance feedback URMs, respectively. Both these differences are statistically significant. A more detailed analysis of these results that shows that the relevance proximity and wizard URM's performance is noticeably higher for short queries can be found elsewhere (Leuski, 2000).

7.2. USER STUDY

The idea of searching for relevant information by examining the document that is the closest to already discovered relevant material seems simple. We assume that given an accurate visual representation of inter-document similarities the user can effectively locate the relevant documents without any aid from the system.

Thus, the next question of our study is: 'How effective in locating the relevant information will the user be when given the spring-embedding visualization of the retrieved set?' We hypothesize that the notion of spatial similarity in the spring-embedding visualization is an intuitive and accurate metaphor for representing inter-document relationships. We expect that the user's implicit URM will be similar to our relevance proximity URM in both procedure and effectiveness. We use the relevance proximity URM as our baseline in these experiments because it is more intuitive than the Wizard URM – the Wizard URM maintains substantially more state than we expect a person to be able to keep track of.

To test these hypotheses we have implemented a computer-based user study. It was designed to simulate a user looking for relevant information in the visualization (i.e., the document title list was removed from the system). Each participant in this study had to solve a number of information-foraging problems. Every problem consisted of a set of 50 spheres floating in space: one green, a few red (maybe none), and all the rest white. (This is the same initial condition adopted in the experiments considered in the previous section.) The participants were told that the 'true' color

of the white spheres were either red or green. The true color of a white sphere could be discovered – the sphere could be 'opened up' – by double-clicking on the sphere with the mouse pointer. The participants were asked to find all the green spheres as quickly as possible, trying to avoid opening red spheres. The participants received a small time penalty for opening a sphere – the sphere was animated for several seconds before showing its true color. They were also prohibited from double-clicking on a sphere while another was opening. This was done to discourage the users from clicking the spheres in random order. At the same time it crudely simulated the delay that would have been experienced by a person while reading and judging the document.

The participants were told that spheres of the same color (e.g., green spheres) tend to appear in close proximity to each other (similar spheres generally group together) but not necessarily so. The last hint was a direct corollary from the Cluster Hypothesis as the spheres represented the documents, and the color, the document relevance value. A green sphere indicated a relevant document and a red one indicated a non-relevant document. However, the participants were not told the meaning of the spheres. We believe this design eliminates a high uncertainty that is generally connected with query formulation and passing relevance judgments (Koenemann and Belkin, 1996; Swan and Allan, 1998) and allows us to isolate the navigation properties of the visualization which are the focus of our study.

The problems were presented in two and three dimensions. The three-dimensional effect was created by using a 3D-rendering engine. To improve the depth perception, a simulated fog effect was added to the picture. The participants were able to rotate, slide, and zoom the set of spheres. The application interface of a two-dimensional presentation was equivalent to that of a three-dimensional one except that the user saw a flat structure on the screen (i.e., he or she could still rotate and zoom the 2-dimensional structure).

Each participant was presented with ten problems. We divided the problems into two equal groups. The problems in one group were shown in two dimensions, the problems in the other – in three dimensions. The dimensions in which each group of problems was shown alternated between users. We also varied the order in which the groups were presented and the order in which the problems inside each group were presented. This was done to account for a possible learning effect. Before each group of problems was shown to a participant he or she was given two training problems to familiarize herself with the application interface. The participants filled out two questionnaires about their experience with the system – one after the first five problems and another at the end of the study. The results from those questionnaires are summarized in the next section.

The study was designed to be completely supervision-free. The software was written in Java and it was available on the Web. We advertised the study in local newsgroups and in information retrieval mailing lists on the Internet. 40 people expressed their interest in the study by accessing the software; 20 of them completed it, spending on average one hour and thirty minutes with the system.

7.3. USER STUDY RESULTS

To create the problems we randomly selected ten topics from TREC topics 251–350 (see Section 5). We used the 'title' versions of the corresponding queries to retrieve the 50 top ranked documents for each topic. These documents were visualized and presented to the users in 2 and 3 dimensions. At the beginning of each problem the spheres corresponding to the highest ranked relevant document and the non-relevant documents that precede it in the ranked list were shown in color. The rest of the documents were shown in white – i.e., as if starting after the first relevant document in the ranked list was found. This was supposed to provide the users with the starting point in their exploration.

The users examined the white spheres in sequence. The order in which each user double-clicked the spheres defined the user's search strategy or the user's implicit relevance model. To distinguish it from the explicit relevance proximity URM discussed in previous experiments (F_{rel}) we call the latter the *algorithmic* search strategy. We calculated the average precision for the user's model and averaged it across all users and all problems. Table VII shows that the algorithmic URM produced better average precision numbers than the users' selections. Note, though, that the table also indicates that the users do better by using the visualization than by blindly following the ranked list.

The differences between the users' performance in both dimensions and the ranked list, between the users' performance and the algorithmic URM in both dimensions, and between the users' performance in individual dimensions are statistically significant by two-tailed t-test with at least $p < 0.01$.

The algorithmic search strategy has a higher performance when working with a 3-dimensional representation of the document set than with a 2-dimensional

Table VII. Users' performance navigating the visualizations of ten randomly selected document sets. The numbers are averaged across all selected document sets. Average precision numbers, percent improvement over the ranked list search strategy, percent improvement over the relevance proximity URM in the corresponding dimension, and percent improvement of using 3D over 2D are shown. We also show the significance level for each difference by two-tailed *t*-test.

URM		Average precision	Improvement		
Ranked list		42.9			
Algorithmic strategy	2D	59.1			
	3D	61.4			
Users	2D	55.8	vs. Ranked list	(+30.1%)	$p < 5 \cdot 10^{-8}$
			vs. Algorithmic strategy in 2D	(−5.7%)	$p < 5 \cdot 10^{-4}$
	3D	53.2	vs. Ranked list	(+24.1%)	$p < 5 \cdot 10^{-6}$
			vs. Algorithmic strategy in 3D	(−13.3%)	$p < 5 \cdot 10^{-12}$
			vs. Users in 2D	(−4.6%)	$p < 0.01$

representation. The users on the other hand show much better results in 2 dimensions than in 3 dimensions. From this observation and also from the comments we have collected during the study we conclude that the users have a much harder time establishing proximity relationships and navigating the 3-dimensional visualization.

We were interested in comparing the users' browsing strategy with the ordering defined by the relevance proximity URM. Each time the user selected a sphere to examine, the algorithmic URM ranks the unexamined (white) spheres by the spatial proximity to the current cluster of examined green (relevant) spheres and assigns a rank number to the user's choice. Note that in this situation the algorithm will select the highest ranked sphere. If both the algorithm and the user select the same sphere, the user's choice is ranked as one. Figure 4 shows the rank of the users' choice as each successive sphere is selected. The X-axis is the index of the examined sphere. We show both the average rank and the error bar indicating the standard deviation. We also show the average and standard deviation for the number of green spheres

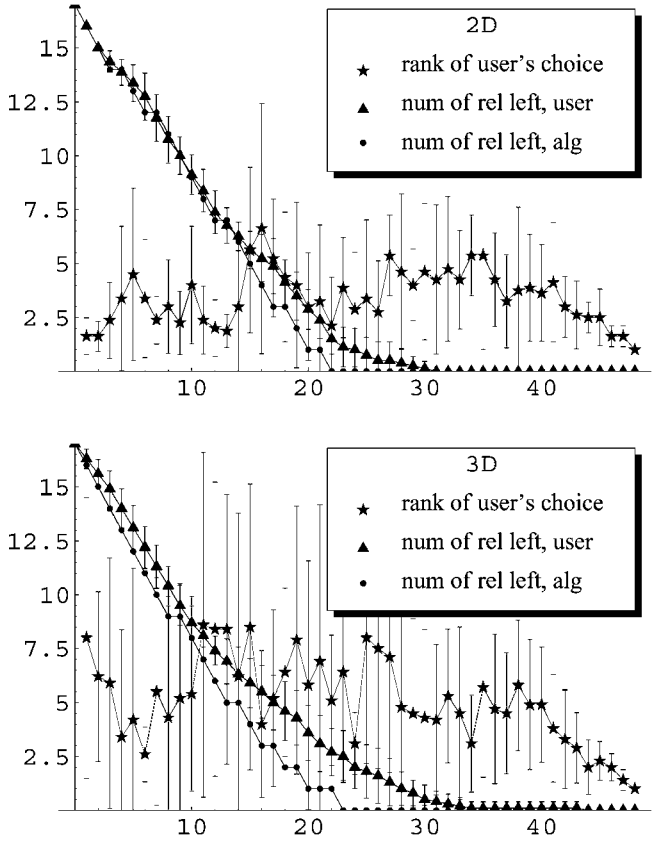

Figure 4. Comparison of the users' average document ordering to the algorithmic URM in 2 and 3 dimensions for one document set. The X-axis is the number of the examined documents. We show the number of green spheres remaining unexamined by the algorithm ('num of rel left, alg'), the same number for the users ('num of rel left, user'), and the rank of the user choice ('rank of user's choice').

remaining unexamined by the user at each step, and the same number for the algorithm. For example, at the beginning there are 18 unexamined spheres ($x = 0$). In two dimensions both the users and the algorithm succeed in locating a green sphere on first pick – the number of unexamined green spheres drops to 17 for $x = 1$. The users always select a green sphere – the standard deviation is zero. We also see that the sphere selected by the users on that step has an average rank of 1.5. It means that the users almost always picked up either the first or the second closest white sphere to the first green sphere.

Figure 4 presents the algorithm and users' behavior for one of the document sets. We have observed similar effects on the rest of the data. Comparing the plots in 2 and 3 dimensions we see that:

1. the algorithm is more successful in 3 dimensions than it is in 2 dimensions – the plot line for the number of green spheres left after the algorithm's pick descends faster in the bottom figure (3D).
2. the users are more successful in 2 dimensions than they are in 3 dimensions – the plot line for the number of green spheres left after the users' pick descends faster and trails the same line for the algorithm closer in the top figure (2D).
3. the users' ordering is more similar to the algorithmic URM in 2 dimensions – the average rank of the users' choice is smaller.
4. the users' choices have less variance in 2 dimensions than in 3 – the error bars are generally shorter in the top figure (2D).
5. the users' ordering is not significantly different from the algorithmic URM at least at the beginning of the search – the average users' choice is always less than one standard deviation apart from the algorithm's first choice for the first 14 examined documents.
6. when the users' ordering diverges sharply from the algorithmic URM choices, the users' selection is more likely to be a red sphere. Examples of that divergence appear as sharp 'peaks' on the user's choice graph in the top figure at $x = 5$ and $x = 15$. That mostly red spheres are chosen is clear in the same graph because the number of green spheres (relevant) left stays noticeably higher for the user than for the algorithm.

We asked the users to fill out short questionnaires about their experience with the system comparing 2D with 3D. Specifically, we asked the users to assign a 'grade' between 1 and 5 measuring how easy it was to use each presentation and if, in their opinion, the system did a good job at organizing the objects. The average grades in Table VIII show that the users preferred the 2D presentation over the 3D one. They overwhelmingly found 2D visualization easier to use and they were generally satisfied with the system's arrangement of the green and red spheres.

We conclude that both our hypotheses are supported. The users have no difficulty grasping the idea of spatial proximity as the metaphor for inter-object similarity.

Table VIII. Users' responses to a number of questions comparing 2D with 3D visualizations. The answers were given as 'grades' between 1 and 5. The average grade is shown for each question. The higher numbers are better ('easier', 'more satisfied').

Question	2D	3D
How easy was it to **understand how to use** the system?	4.4	3.4
How easy was it to **learn to use** the system?	4.6	3.6
How easy was it to **use** the system?	4.3	2.7
Are you satisfied with the system's organization of data? Does the system's placement of the objects makes it easier to find the green spheres?	3.4	2.7
Are you satisfied with your performance in finding the green spheres?	3.6	3.1

Their browsing strategy is very similar to the notion of selecting the spheres that are close to the known green spheres. In fact, one of the participants explicitly stated he was trying to pick up the 'white ball that is the closest to the average of the green set.' Generally the users were very successful in following this tactic. However, the more spheres that were examined, the more difficult it became to correctly identify the similarity between the cluster of green spheres and the remaining white spheres. This task was even more difficult in three dimensions. The users constantly pointed out that correct identification of the inter-sphere distances in 3D required frequent rotations of the structure, thus making the visualization task more difficult. We believe these effects account for the observed differences in precision between the algorithmic and user document orderings.

7.4. USING THE WIZARD

Based on the experiments described in this paper, we created Lighthouse – an interface system for a web-based search engine. Lighthouse integrates a ranked list, clustering, and a spring-embedding visualization. A detailed description of the system can be found elsewhere (Leuski and Allan, 2000a,b). Lighthouse uses a wizard tool based on the wizard URM that accepts the searcher's relevance judgments, computes its estimations of relevance for the unexamined documents, and visualizes this information on the screen.

For example, if Lighthouse's URM estimates that a document is relevant it highlights the corresponding document sphere and title using some shade of green. The intensity of the shading is proportional to the strength of the system's belief in its estimation – the more likely that the document is in the category, the brighter the color. The same shade of color is used to highlight the document title backgrounds. Additionally, the length of that highlighted background is proportional to the strength of the system's belief in its estimation.

Figure 5 shows a screenshot of the Lighthouse system displaying the same top 50 documents returned by the Google search engine in response to the query 'Samuel Adams' we presented in Figures 1 and 2. The spring-embedding visualization presents the documents in three dimensions. The dark ovals at the bottom of the picture

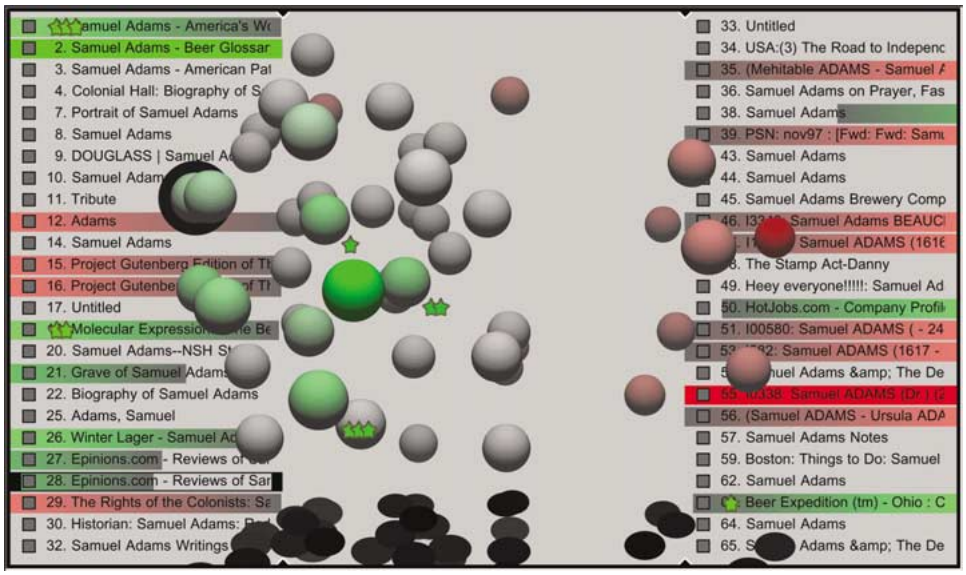

Figure 5. Lighthouse visualization of documents for the 'Samuel Adams' query in three dimensions.

are the shadows of the spheres. We use the shadows, perspective projection, and fog effect to create a sense of the depth in the picture.

Green indicates relevant documents, e.g., the document ranked 2 is marked as relevant (note that the title background has complete and uniform green fill) and the corresponding document sphere is on the left side of the visualization. We use red to mark non-relevant documents. The document ranked 55 by Google is marked as non-relevant. The color completely fills the title background. The corresponding sphere is on the right side of the picture. The Lighthouse wizard URM estimates that many of the documents should be assigned to the non-relevant category, e.g. documents 12, 15, 16, 29, and 35 have a light-red background with smooth gradient fill. The lightly-shaded red spheres corresponding to those documents occupying the right side of the picture. The documents with the ranks 1, 19, 21, 26, etc. are estimated to be relevant – they have light-green title backgrounds and document spheres. The user has selected document ranked 28. Its title has a black outline and the corresponding sphere located in the top left quarter also has a thick black outline. The URM estimates this document far less likely to be in the relevant category than the document with rank 1: green color hardly fills one half of the title background and the sphere has a lighter shade of green.

Lighthouse displays the top three wizard URM choices with the green stars attached to the document titles and spheres. The stars are placed diagonally to the right and bottom of the spheres. Three stars indicate the document ranked first by the wizard (document 1). Two (document 18) and one star (document 63) point to the second and third choices respectively. As the user examines more documents and assigns the categories to them, Lighthouse recomputes its estimations based on the wizard URM and adjusts the colors in the visualization.

The highlighted backgrounds in the left column are aligned on the left side and the highlighted backgrounds in the right column are aligned on the right side. Note that a gray sphere and no highlighting for the document title reflect that the document can be equally likely assigned to any one of the categories.

Table VI showed very clearly that the wizard URM is the most effective one for a user interested in finding relevant documents. However, the user study just described suggests that searchers could not always follow such a strategy on their own: they have progressively more difficulty determining where to look next as the number of examined documents grows.

This situation made creating a Lighthouse wizard very obvious. A searcher can enable the wizard to have it highlight the next three documents that the algorithm would choose (see Section 7.1). If the searcher were to enable the wizard and follow its suggestions blindly, the result would be identical to the algorithmic approach of F_{wzd}.

The Lighthouse wizard can also be extended to multiple aspects of relevance – multiple simultaneous URMs. Lighthouse supports the capability of marking documents with one or more tags that can be thought of as aspects of relevance. The wizard can be asked to identify unmarked documents likely to belong to one of those aspects (by that aspect's URM), making it easier for a searcher to exhaustively recover all documents on a single aspect. In addition, the searcher can set the wizard to suggest documents that do not match *any* of the currently marked aspects, allowing a searcher to identify the various aspects or to find the one that is of interest.

Lighthouse is one of several clustering-based visualization systems that have been created for research use. For example, Hearst and Pedersen (Hearst and Pedersen, 1996) considered a clustering system for visualizing retrieved documents. They did not investigate the question of how the clustering can help a user to locate relevant documents. Their system breaks the retrieved results into a fixed number of document groups, while Lighthouse sets the clustering parameters based on inter-document similarity.

The Bead system (Chalmers and Chitson, 1992) uses the same Multidimensional Scaling algorithm as Lighthouse for visualizing the document set. That system was designed to handle very small documents – bibliographic records represented by human-assigned keywords. The Bead research did not evaluate the system.

There exist many other approaches for clustering and document visualization. An extended review of the related work in both areas can be found elsewhere (Leuski, 2001b).

8. Conclusion

We believe that ranked lists, strict clustering, and the visualization of inter-document similarities combine to provide a highly useful interactive IR environment. A user may find that many queries are satisfied by a simple ranked list, that pure clustering is sometimes ideal, and that more complex inter-relationships may be made

transparent using the spring-embedding visualization, if it is required. We have shown that clustering and visualization are both useful.

We have done that by constructing user relevance models that capture the feedback provided by a searcher as do the simpler models of relevance that are typical in past IR research. We have shown how these models can capture a wide range of methods for a system to model a user's notion of relevance and have evaluated three styles of URM. We showed that a searcher that follows the 'advice' of the wizard URM (or a searcher that uses the same strategy) is significantly more likely to find relevant material quickly than one using classic retrieval techniques.

Our motivation for exploring URMs and building the Lighthouse system to incorporate them was to support interactive IR. We are continuing our work in that direction, exploring additional ways that Lighthouse and similar technologies can be exploited and included in URMs (Frey et al., 2001). We would like to carry out full user studies to compare URMs to real user activities and models of relevance, but have not done so yet. User studies are expensive and time consuming, and should not be undertaken until the problem being investigated is sufficiently well understood. The experiments in this study provide 'simulation' evaluations of users (i.e., users that follow an algorithmic model) and provide us with an improved understanding of the problem, making it possible for us to eventually carry out a user study that is valuable as well as interesting.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by the National Science Foundation under grant number IRI-9619117, and in part by the National Science Foundation Cooperative Agreement number ATM-9732665 through a subcontract from the University Corporation for Atmospheric Research (UCAR). Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

Aalbersberg, I. J.: 1992, Incremental relevance feedback. In: *Proceedings of ACM SIGIR*. pp. 11–22.

Allan, J.: 1995, Automatic Hypertext Construction. Ph.D. thesis, Cornell University.

Allan, J., Callan, J., Croft, B., Ballesteros, L., Broglio, J., Xu, J. and Shu, H.: 1997, 'INQUERY at TREC-5'. In: *Fifth Text REtrieval Conference (TREC-5)*. pp. 119–132.

Allan, J., Callan, J., Croft, W. B., Ballesteros, L., Byrd, D., Swan, R. and Xu, J.: 1998, INQUERY does battle with TREC-6. In: *Sixth Text REtrieval Conference (TREC-6)*. pp. 169–206.

Benyon, D. and Murray, D.: 1993, Developing adaptive systems to fit individual needs. In: *Proceedings of the 3rd International Workshop on Intelligent User Interfaces*. pp. 115–121.

Berliner, H.: 1979, On the construction of evaluation functions for large domains. In: *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*.

Bookstein, A.: 1983, Information retrieval: A sequential learning process. *Journal of the American Society for Information Science* **34**(5), 331–342.

Buckley, C. and Salton, G.: 1995, Optimization of relevance feedback weights. In: *Proceedings of ACM SIGIR*. pp. 351–357.

Chalmers, M. and Chitson, P.: 1992, Bead: Explorations in information visualization. In: *Proceedings of ACM SIGIR*. pp. 330–337.

Chen, H.: 1995, Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society for Information Science* **46**(3), 194–216.

Chen, J. R. and Mathe, N.: 1995, Learning subjective relevance to facilitate information access. In: *CIKM*. pp. 218–225.

Cypher, A.: 1991, Programming repetitive tasks by example. In: *Proceedings of the ACM Conference on Computer Human Interaction*. pp. 33–39.

Frey, D., Gupta, R., Khandelwal, V., Lavrenko, V., Leuski, A. and Allan, J.: 2001, Monitoring the news: a TDT demonstarion system. In: *Proceedings of the first International HLT Conference*.

Frisse, M. E. and Cousing, S. B.: 1989, Information retrieval from hypertext: Update on the dynamic medical handbook project. In: *Proceedings of the ACM Conference on Hypertext*. pp. 199–212.

Fruchterman, T. M. J. and Reingold, E. M.: 1991, Graph drawing by force-directed placement. *Software–Practice and Experience* **21**(11), 1129–1164.

Harman, D. and Voorhees, E. (eds.): 1997, The Fifth Text REtrieval Conference (TREC-5). NIST.

Harman, D. and Voorhees, E. (eds.): 1998, The Sixth Text REtrieval Conference (TREC-6). NIST.

Harman, D. and Voorhees, E. (eds.): 1999, The Eighth Text REtrieval Conference (TREC-8). NIST.

Harman, D. and Voorhees, E. (eds.): 2000, The Ninth Text REtrieval Conference (TREC-9). NIST.

Harman, D. and Voorhees, E. (eds.): 2001, The Tenth Text REtrieval Conference (TREC-2001). NIST.

Hearst, M. A. and Pedersen, J. O.: 1996, Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In: *Proceedings of ACM SIGIR*. pp. 76–84.

Hersh, W. and Over, P.: 2001, The TREC-9 interactive track report. In: *The Ninth Text REtrieval Conference (TREC-9)*. pp. 41–50.

Koenemann, J. and Belkin, N. J.: 1996, A case for interaction: A study of interactive information retrieval behavior and effectivness. In: *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*. pp. 205–212.

Lance, G. N. and Williams, W. T.: 1967, A general theory of classificatory sorting strategies: 1. Hierarchical Systems. *Computer Journal* **9**, 373–380.

Leuski, A.: 2000, Relevance and reinforcement in interactive browsing. In: *Proceedings of Ninth International Conference on Information and Knowledge Management (CIKM'00)*. pp. 119–126.

Leuski, A.: 2001a, Evaluating document clustering for interactive information retrieval. In: *Proceedings of Tenth International Conference on Information and Knowledge Management (CIKM'00)*. pp. 41–48.

Leuski, A.: 2001b, Interactive Information Organization: Techniques and Evaluation. Ph.D. thesis, University of Massachusetts at Amherst.

Leuski, A. and Allan, J.: 2000a, Details of Lighthouse. Technical Report IR-212, Department of Computer Science, University of Massachusetts, Amherst.

Leuski, A. and Allan, J.: 2000b, Lighthouse: showing the way to relevant information. In: *Proceedings of InfoVis'2000*.

Leuski, A. and Croft, W. B.: 1996, An Evaluation of Techniques for Clustering Search Results. Technical Report IR-76, Department of Computer Science, University of Massachusetts, Amherst.

Lewis, D. D.: 1992, An evaluation of phrasal and clustered representations on a text categorization task. In: *Proceedings of ACM SIGIR*. pp. 37–50.

Mirkin, B.: 1996, *Mathematical Classification and Clustering*. Kluwer.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M. and Gatford M.: 1995, Okapi at TREC-3. In: D. Harman and E. Voorhees (eds.): *Third Text REtrieval Conference (TREC-3)*.

Rocchio, Jr., J. J.: 1971, Relevance feedback in information retrieval. In: G. Salton (ed.): *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, Inc., pp. 313–323.

Salton, G.: 1989, *Automatic Text Processing*. Addison-Wesley.

Sullivan, J. W. and Tyler, S. W. (eds.): 1991, *Intelligent User Interfaces*. ACM.

Swan, R. and Allan, J.: 1998, Aspect Windows, 3-D Visualizations, and indirect comparisons of information retrieval systems. In: *Proceedings of ACM SIGIR*. pp. 173–181.

van Rijsbergen, C. J.: 1979, *Information Retrieval*. London: Butterworths. Second edition.

Xu, J. and Croft, W. B.: 1996, Querying expansion using local and global document analysis. In: *Proceedings of ACM SIGIR*. pp. 4–11.