The Best of Both Worlds: Combining Ranked List and Clustering

Anton Leuski* and James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
{leuski,allan}@cs.umass.edu

May 3, 1999

Abstract

An information retrieval system arranges retrieved documents into a list and ranks them in the order they are expected to be relevant. The ranked list is a well-known and widely accepted technique for information presentation for the task of finding relevant documents as quick as possible. Clustering is also a well-known and successful method for grouping similar documents together – in particular, for grouping relevant documents. In this paper we present a system that combines the ranked list with a clustering approach, accepts a modest amount of user feedback, and yields an approach that exceeds the retrieval effectiveness of a traditional ranked list, with and without relevance feedback.

1 Introduction

Finding relevant information in a collection of full-text documents is a very complex task. An information retrieval system generally retrieves a set of documents that contains both relevant and non-relevant material. Different methods for information presentation are employed to help a user separate interesting information from the rest of the data. These presentation techniques have their advantages and disadvantages. We are interested in how to combine presentation methods, hoping that by leveraging the individual strengths of each approach we will be able to help the user locate the relevant information more quickly and therefore more effectively.

The ranked list is a widely accepted approach for presenting retrieved information. An information retrieval system returns the documents arranged in the order they are expected to be relevant to the user's query: the first document is the best match to the query, the second is the next most likely to be helpful, and so on. The user of the system is expected to follow the system recommendations – start from the top of the ranked list and follow it down looking at the documents one by one. In the ideal case the user will see all the relevant documents before any non-relevant ones. Generally such a simple model breaks down and the relevant documents appear to be scattered all over the ranked list. To bring the scattered document together we also consider clustering approaches.

The use of clustering in Information Retrieval is based on the Cluster Hypothesis: "closely associated documents tend to be relevant to the same requests" [24, p.45]. Croft [11] and more recently Hearst and Pedersen [16], showed that this hypothesis holds in a retrieved set of documents. A simple corollary from this hypothesis is that if we do a good job at clustering the retrieved documents, we are likely to separate the relevant and non-relevant documents into different groups. If then we can direct the user to the right group of documents, we would increase her chances of finding the interesting information with minimal effort.

In this paper we discuss an information presentation system that combines a traditional ranked list with a clustering visualization. The clustering visualization groups similar documents together and generates

*Corresponding author

a 2- or 3-dimensional graph where each node corresponds to a document and the distance between nodes is inversely proportional to the inter-document similarity. We experimentally show that the information contained in the graph can be used to produce an alternative ranking of documents that is significantly more effective than the original ranked list.

2 Related Work

Tremendous amounts of work have been done on ranked lists over the past several decades, ranging from improving effectiveness [15] to user evaluations [18]. The basic thrust of that work is so well-known, there is little value in surveying it. Instead we focus on related work in clustering and clustering visualization.

Multiple visualization approaches for clustering have been developed in recent years. Generally these visualizations are designed to present patterns in a document set and they are considered to be browsing interfaces. The format of the presentation varies significantly from system to system. For example, Hearst and Pedersen [16] suggest a clustering system that groups the retrieved documents into some preselected number of clusters and displays them simultaneously as lists of titles. A similar presentation was developed by Leuski and Croft [19], however they do not pre-specify the number of clusters and their display looks more like the traditional ranked list.

It is common for the information organization to be presented graphically. The documents, paragraphs, and concepts are usually shown as points or objects in space with their relative position indicating how closely they are related. Allan [1, 4] developed a visualization for showing the relationship between documents and parts of documents. It arrayed the documents around an oval and connected them when their similarity was strong enough.

The Vibe system [12] is a 2-D display that shows how documents relate to each other in terms of user-selected dimensions. The documents being browsed are placed in the center of a circle. The user can locate any number of terms inside the circle and along its edge, where they form "gravity wells" that attract documents depending on the significance of those terms in that document. The user can shift the location of terms and adjust their weights to better understand the relationships between the documents.

High-powered graphics workstations and the visual appeal of 3-dimensional graphics have encouraged efforts to present document relationships in 3-space. The LyberWorld system [17] includes an implementation of the Vibe system described above, but presented in 3-space. The user still must select terms, but now the terms are placed on the surface of a sphere rather than the edge of a circle. The additional dimension should allow the user to see separation more readily.

The Bead system [9] uses a graph drawing algorithm called spring-embedding for placing high-dimensional objects in a low-dimensional space. The system puts documents in 3-dimensional space, positioning them according to the inter-document similarity. The Bead research did not investigate the question of separating relevant and non-relevant documents. The system was designed to handle very small documents – bibliographic records represented by human-assigned keywords. A similar approach was adopted by Leuski and Allan [20]. They extended the Bead spring-embedding approach and applied it to complete, full-sized documents. We adopt spring-embedding for our system.

All of this work has focused on developing the power of a ranked list or of a clustering approach. Our work, in contrast, unites the two approaches in one system and shows how their respective powers combine to yield an even more effective system. Swan and Allan [23] considered a system with both components but although their parts were tightly integrated, there was no exploration of how the two could be effectively used together. Anick and Vaithyanathan [7] also integrated clustering and ranking to facilitate browsing of the retrieval results, but did not explore effectiveness.

Hearst and Pedersen [16] also considered the effect of identifying clusters of relevant documents, and how the resulting ranking compared to an unclustered list. However, they generated discrete clusters, explored whether the user could select by keywords, and considered only that cluster as compared to the ranked list.

3 System Design

Our system consist of two parts: the ranked list and the clustering visualization. The ranked list is supplied by INQUERY [6]. The INQUERY system is based on a probabilistic model of retrieval and it neither incorporates the notion of similarity between documents nor includes document representations. Therefore, to cluster the documents we use a vector-space approach where each document is represented by a vector V such that v_i is a *tf·idf* weight of the ith term in the vocabulary:

$$v_i = \frac{tf}{tf + 0.5 + 1.5\frac{doclen}{avgdoclen}} \cdot \frac{\log(\frac{N+0.5}{docf})}{\log(N+1)} \tag{1}$$

where *tf* is the number of times the term occurs in the document, *docf* is the number of documents the term occurs in, and N is the number of documents in the collection. The distance between a pair of documents is measured by one over the cosine of the angle between the corresponding vectors (i.e, $1/\cos\theta$). That is the inverted measure of similarity between documents that is widely used in the vector-space model [22].

3.1 Spring-Embedding

To cluster the document vectors we use a spring-embedding algorithm [13]. An advantage of the algorithm is that it handles both the task of clustering and the task of generating a visual presentation of the clustering. Spring-embedding takes a set of high-dimensional vectors and generates 2- or 3-dimensional approximation of this set, attempting to preserve the relative distances between vectors. The algorithm models each document vector as an object in the 2- or 3-dimensional visualization space. The objects are connected with springs and the strength of each spring is inversely proportional to the $1/cos$ distance between the corresponding document vectors. 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-document similarity is mapped approximately onto the Euclidean distance between points.

There are many other techniques (e.g., LSI, Linear Programming, and so on) that might be used to the same effect. Our choice was motivated by the graph-drawing heritage of the spring-embedding [13] – it is supposed to generate eye-pleasing pictures – and availability of the source code. A detailed description of the algorithm can be found elsewhere [10].

If all the constraints (all the springs) are incorporated into the spring-embedding process the resulting structure is very tight and resembles a "soccer-ball." To prevent this from happening we introduce a threshold parameter into the algorithm – all springs that correspond to pairs of documents with the inter-document distances above a predefined value are weakened significantly. This process eliminates the constraints imposed on the system by these springs and at the same time it keeps the overall structure together. It allows us to generate more "interesting" spatial structures. Unfortunately, we now have a parameter that we do not know how to select. We will show, however, that the system is not very sensitive to the parameter value, so this is not a major problem.

Figure 1 gives an example of our system. It shows fifty retrieved documents arranged into a ranked list and recorded on two panels on the left and right sides of the screen. In the middle there is a 2-dimensional spring-embedded visualization of the same fifty documents in the form of fifty spheres floating in space. The document representations in the list and in the visualization are tightly linked: a click on a sphere highlights both the sphere and the corresponding document id in the list and vice versa. The document ids can be replaced by titles, but ids are shown here to save space.

This Figure 2 clearly illustrates potential advantages of the clustering visualization over the ranked list: when the relevant documents are widely scattered in the ranked list, the same documents are tightly grouped together in the visualization. The next section describes how we can take advantage of the clustering and rearrange the documents in a more effective order.

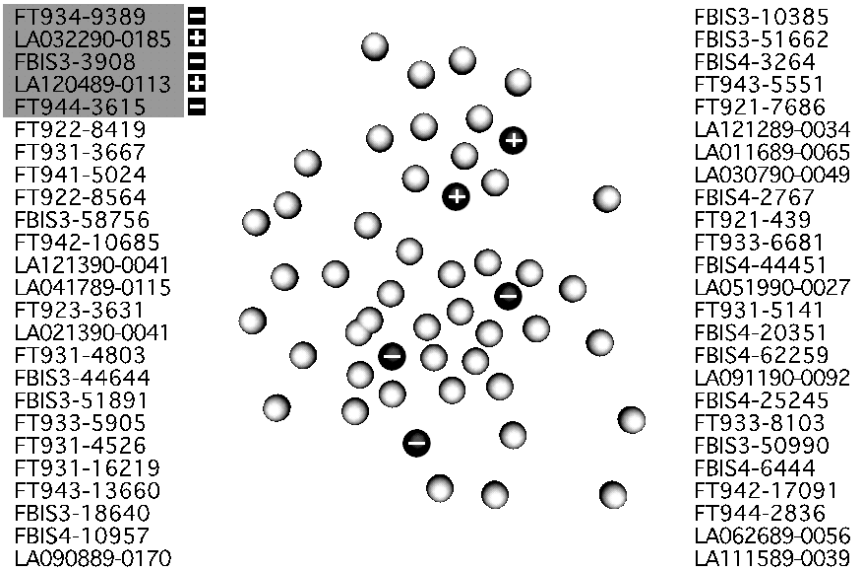

Figure 1: The combination of a ranked list and spring-embedding visualization for a group of fifty retrieved documents. The ranked list is presented on the two panels starting at the top of the left panel and continuing down and then to the top of the right panel. The 2-dimensional visualization is in the middle. We show a snapshot at the time when the user has looked at the top five documents in the ranked list. Their ids are highlighted with a gray background in the list and the corresponding spheres are shown in dark gray color. A "+" indicates a relevant document; a "-" flags non-relevant documents.

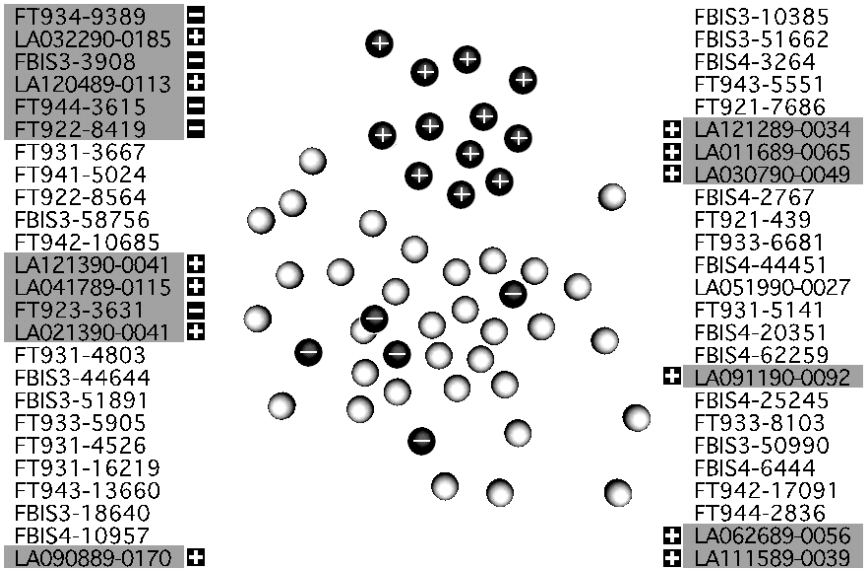

Figure 2: The same example as on Figure 1 after the user has discovered all the relevant documents in the retrieved set.

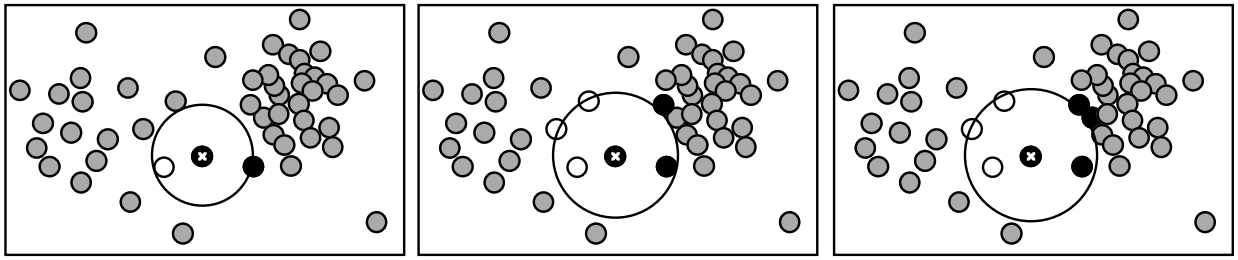

Figure 3: Shows three consecutive snapshots of how the visualization will build a static relevant proximity ranking. It begins with the first relevant document represented by the black disk in center of the picture and looks for the closest disk. We show the state as the first, second, and third relevant documents are discovered. The black disks represent the known relevant documents, the white disks – the known non-relevant, and the gray disks are the unknown documents.

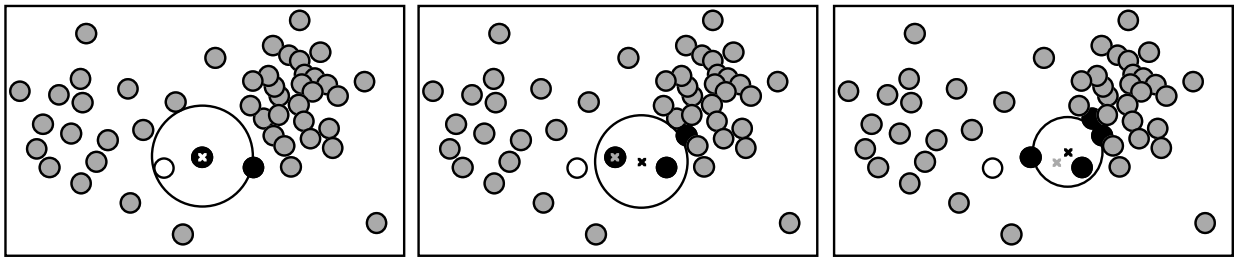

Figure 4: Shows three consecutive snapshots of how the visualization will select the documents for the dynamic relevant proximity ranking. It begins with the first relevant document represented by the black disk in center of the picture and looks for the closest disk. We show the state as the first, second, and third relevant documents are discovered. The black disks represent the known relevant documents, the white disks – the known non-relevant, and the gray disks are the unknown documents. The bright "X" represents the location of the center mass of all known relevant documents. The gray "X" shows the old location of the center of mass at the previous snapshot.

3.2 The Use of Clustering

Suppose that we know one relevant document. For example, we find this document by traversing the ranked list. Now we re-rank the rest of the documents according to their spatial proximity to that relevant document. We create a new ranking that we will call *static relevant proximity ranking*. The ranking can be shown to the user and we can compare it to the original ranked list. Our system presents the new ranking by highlighting the unknown document that is the closest to the known relevant document. After the user looks at the "suggested" document the visualization highlights the next document in the ranking and so on. This process can viewed as "feeding" the user one document at a time. Figure 3 shows an example of how the visualization builds the static relevant proximity ranking.

If the user extends the effort to assign relevance judgments to the documents as she proceeds down the new ranking we can use this information to adjust the ranking in the following way: instead of ranking the rest of the documents by their proximity to the single (first) relevant document, we re-rank the documents by their proximity to the center of mass of all relevant documents so far. Obviously this ranking may change as soon as the user discovers a relevant document. We call such a ranking *dynamic relevant proximity ranking*. Figure 4 shows an example of how the visualization builds a dynamic relevant proximity ranking.

The new ranking – either a static or dynamic relevant proximity ranking – is presented to the user in the form of *suggested* documents, highlighted in both the ranked list and the spring-embedding visualization. The system does not force the user to view the documents in the suggested order. In next section we compare

the ranking generated by the visualization to the original ranked list.

4 System Evaluation

Consider the following scenario: a user receives the top fifty documents from an information retrieval system. The documents are presented in our visualization system both in the form of the ranked list and the spring-embedding visualization. The user reads the documents starting from the top of the ranked list until she finds one relevant document. At this point our system suggests four alternative rankings to follow:

1. the original ranked list;
2. the ranking obtained by taking the relevant document found by the user to run an automatic relevance feedback process [2], modifying the original query, and rearranging the document by the similarity to the new query;
3. the static relevant proximity ranking created by the visualization;
4. the dynamic relevant proximity ranking created by the visualization (assuming that the user will keep the system supplied with relevance judgments).

Given a set of known documents and relevance judgments we will create all four rankings automatically and evaluate them using average precision.

4.1 Experimental Setup

For our experiments we used TREC [14] ad-hoc queries with their corresponding collections and relevance judgments. Specifically, TREC topics 251-300 were converted into queries and run against the documents in TREC volumes 2 and 4 (2.1GB) that include articles from Wall Street Journal, Financial Times, and Federal Register. For each TREC topic we considered four types of queries: (1) a query constructed by extensive analysis and expansion [5]; (2) the description field of the topic; (3) the title of the topic; and (4) a query constructed from the title by expanding it using Local Context Analysis (LCA) [25].

In addition, we used TREC topics 301-350 to create queries to be run against TREC volumes 4 and 5 (2.2GB) that include articles from the Congressional Records, Financial Times, and Los Angeles Times. Again, the same four different types of queries were constructed, except instead of just using the description field for the second query type, we used both the title and the description field of the topic (this was done to compensate for an error in the topic creation in TREC [15]). For each query we selected the 50 highest ranked documents and spring-embedded them in 2 and 3 dimensions.

4.2 Results – Stability of Spring-Embedding

The first question we investigate is how much the quality of the ranking generated by our system depends upon the uncertainty from the spring-embedding algorithm. There are two sources of uncertainty: the initial position of the objects before we start the embedding process and the threshold value that defines how many springs exist in the structure.

The final spring-embedding picture depends on the initial position of the objects. We experimented with individual queries by starting the embedding process from different initial placement of objects. We have observed that the spring-embedding algorithm is very stable to the initial arrangements. Table 1 shows the mean and variance of the average precision for the dynamic relevant proximity ranking. The ranking is generated for 2 and 3 dimensional embeddings for several TREC-5 queries. It is clear that 3 dimensional embeddings are less sensitive to the starting conditions. The Table suggests that our final precision numbers will be accurate to within less than a half percent of what could be reported.

The spring-embedding algorithm also has a threshold parameter associated with the embedding process (Section 3.1). Recall that the threshold defines what constraints (springs) are left in the embedding process. There are generally $(N-1)N/2$ different threshold values for N objects. Unfortunately, it is not clear how to select the threshold parameter effectively. We assume that the threshold is selected randomly and we

Table 1: Shows how the precision of different queries depends on the initial arrangements of object for the spring-embedding. The table contains the mean and variance of the precision across 40 initial random arrangements. The numbers are given for several TREC-5 title queries embedded in 2 and 3 dimensions.

Query ID	Average Precision (%)	
	2D	3D
252	37.7±1.2	32.1±0.3
253	99.9±0.0	100.0±0.0
254	52.8±0.4	60.6±0.1
257	67.2±0.2	69.0±0.0
259	81.8±0.9	87.2±0.0
261	39.2±0.4	53.7±0.0
262	30.7±8.5	31.7±0.0
263	72.9±0.4	71.0±0.3
264	24.4±0.2	22.6±0.0
273	85.6±0.4	88.3±0.2
282	51.4±0.6	45.3±0.3

average the precision across all possible values of the threshold for each query – i.e., we set a threshold, run the spring-embedding, create a ranking, measure its precision, and repeat this for the next threshold value. Then we average the precision numbers obtained that way. The variance for these values is generally close to 0.3%. This indicates that the precision of a ranking generated by the spring-embedding does not vary much with the threshold.

4.3 Results – Effectiveness

The second question we consider in this study is how the retrieval effectiveness of the rankings created from the visualization compares to the the effectiveness of the ranked list. How does the dimensionality of the visualization and the initial query properties affect the quality of the rankings?

Table 2 shows that the average precision of the rankings generated from the spring-embedding significantly exceeds precision of the ranked list – it is more effective for the user to follow the suggestions of the visualization part of the system, than to follow the ranked list. We observe on average 20% improvement in the average precision. The improvement is almost always statistically significant by two-tailed t-test with $p = 0.05$.

The addition of the user's input into the ranking generation allows us to create better orderings of documents. The difference between the static and dynamic relevant proximity rankings (Table 2) is consistently about 2-3 points in average precision, constituting an additional 5-6% improvement over the ranked list.

The document vectors occupy a very high-dimensional space. When these documents are presented with the spring-embedding algorithm some of the documents maybe shown nearby when they are actually unrelated because of the constraints imposed by fewer number of dimensions. Our intuition is that a higher dimensional visualization provides more degrees of freedom and therefore a better chance to represent the inter-document relationship accurately than a lower dimensional one. Table 2 shows that indeed the numbers for 3-dimensional visualizations are better than for 2-dimensional visualizations. The difference is about 3% and is statistically significant.

The size and the quality of the original query have a large impact on the clustering. The smallest improvements are shown for the full queries that were created by elaborate topic processing and expansion. There is little chance they will be typed in by a user and the processing to create them automatically is currently prohibitive for an interactive setting. On the other hand the rankings generated by the clustering visualizations for the shortest title queries – the style of queries that people use more often – display large amount of improvement in retrieval effectiveness.

We can also see that running the relevance feedback method and re-ranking the documents based on the

Table 2: Ranked list vs. relevant proximity rankings produced from the spring-embedding. Suppose that a user is searching from the top of the ranked list until she finds a relevant document. Four choices are considered: she continues to follow the ranked list (first column); she applies automatic relevance feedback, modifies the query, re-ranks the documents, and follows the new ranking (second column); she switches to the 2-dimensional spring-embedding and follows the system-generated static proximity ranking (third column); she switches to the 3-dimensional spring-embedding and follows the system suggestions (forth column). The fifth and the sixth columns are for the dynamic proximity rankings in 2 and 3 dimensions. Average precision numbers for each search process are shown. Percent improvement is from the ranked list. Asterisk indicates statistical significance with $p = 0.05$.

Query sets		Ranked List		Spring-Embedding			
				static proximity ranking		dynamic proximity ranking	
		Original	After RF	2D	3D	2D	3D
TREC-5	Full	36.2	35.9 (-0.7 %)	39.1 (8.0* %)	39.4 (9.0* %)	40.7 (12.4*%)	41.6 (15.0*%)
	Desc	34.1	42.6 (24.9* %)	43.4 (27.2*%)	44.5 (30.4*%)	45.8 (34.4*%)	47.2 (38.5*%)
	Title	31.0	32.0 (3.3 %)	37.3 (20.4*%)	37.6 (21.3*%)	39.0 (25.8*%)	38.9 (25.3*%)
	Title + Desc	31.0	34.5 (11.5* %)	34.1 (9.9* %)	35.1 (13.4*%)	36.1 (16.4*%)	37.2 (19.9*%)
TREC-6	Full	51.0	51.3 (0.6 %)	51.0 (0.0 %)	53.2 (4.4 %)	52.9 (3.7 %)	54.9 (7.6* %)
	Desc	42.5	46.9 (10.3* %)	49.2 (15.7*%)	53.4 (25.6*%)	52.5 (23.5*%)	55.2 (30.0*%)
	Title	40.1	42.4 (5.6 %)	49.6 (23.6*%)	51.7 (29.0*%)	53.0 (32.1*%)	54.4 (35.6*%)
	Title + Desc	47.2	41.8 (-11.5*%)	49.8 (5.6 %)	51.8 (9.8* %)	52.4 (11.0*%)	53.5 (13.4*%)
average		39.1	40.9 (4.6* %)	44.2 (12.9*%)	45.9 (17.2*%)	46.5 (18.9*%)	47.9 (22.3*%)

modified query brings much less improvement than the visualization-driven approaches. It occasionally even hurts average performance (Table 2, column 2, rows 1 and 8)

5 Discussion

An interesting question that we have just partially addressed in this paper is how this search based on the information from the ranked list and the visualization compares to automatic relevance feedback [8]. The dynamic relevant proximity ranking produced by the visualization is, in effect, a version of relevance feedback – the order of the ranking changes as the new relevance information becomes available. An alternative to this ranking would be to run automatic relevance feedback each time as we discover a new relevant document, modify the original query, run the modified query, and rearrange the ranked list. However:

1. It is known [3] that automatic relevance feedback method performs erratically with small amounts of relevant information. Whereas by using just one relevant document we have generated a new ranking that is about 17% better than the ranked list.

2. We believe that constantly rearranging of the ranked list because of the relevance feedback might prove very confusing and disorienting to the user.

3. A fair comparison would require that some type of relevance feedback be incorporated into the spring-embedding process itself as well. One approach would be to modify the spring-embedding by moving apart relevant and non-relevant documents as suggested by Leuski and Allan [21]. We are interested in incorporating these techniques into our system.

4. The relevance feedback requires an additional cognitive effort from the user – for the relevance feedback to run the user must mark documents as relevant or non-relevant. There exist anecdotal evidence that people are not likely to do so. Our dynamic relevant proximity ranking is based on the simple idea of "looking at the things that are close to the good things." It is not difficult to imagine that people might follow a similar search pattern basically doing the relevance feedback in their heads. Then explicit

relevance judgments become unnecessary. We are planning to conduct a user study to test how well people recognize and use the similarity information provided by the spatial clues in the visualization.

Another idea is to isolate the ranking generation from the visualization and incorporate it into an automatic information retrieval system to produce improved ranked lists. Then we can do away with constraints imposed on the visualization by small dimensionality of the visualization space. In other words, we can use the original distances between document vector to generate new ranking. We still see several advantages in showing the visualization to the user. First, it is very easy to explain to the user why a particular document was suggested: the visualization shows that it is the closest document to the currently discovered relevant material. Such explanation should give the users more sense of comfort and control and it has been shown that users like to have control over the search process [18]. Second, the methods for constructing both the static and dynamic relevant proximity ranking from the visualization are very simplistic. It is our belief that users may employ additional spatial clues (such as clumpiness) that are not taken into account by our processing. This is a topic for our future work.

6 Conclusions and Future Work

In this paper we presented an information organization system that combines two approaches for visualizing retrieval results: the ranked list and the spring-embedding visualization of clustering. We suggested a method of how the clustering visualization could be used to generate alternative ranking of documents – i.e., a "relevant proximity" ranking.

We showed that if the user starts with the ranked list, finds one relevant document, switches to the spring-embedding visualization and blindly follows the recommendations of our visualization system, her search will be on average 20% more effective than if she would have blindly followed the ranked list. Note that the user might actually do much better than our system suggests and what we showed is that the user is *guaranteed* to do at least 20% better than with the ranked list just by following the system suggestions.

Note that relevance feedback generally produces similar improvements in effectiveness, but it does so by re-submitting the query and presenting a new ranked list. We are achieving these results without re-issuing the query and showing better effectiveness that relevance feedback in such a setting.

We also observed that visualizations created in 3 dimensions possess better quality than 2 dimensional visualizations: they are more stable to the initial conditions in the spring-embedding and generally produce rankings with higher precision values.

This work could be extended in the following directions:

- In this study we considered only two classes of documents: relevant and non-relevant. This was caused by the lack of data of any other kind. We are looking into extending our approach into situations when the user places the relevant documents into multiple classes. That task is modeled after the interactive TREC task of "aspect retrieval."
- We plan on a user study to analyze if the people take advantage of the spatial clues such as proximity that we use to automatically generate new rankings from the spring-embedding visualization. We are also interested in what other kind of information besides simple proximity people receive from the visualization.

Acknowledgments

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

This material is based on work supported in part by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235.

References

[1] James Allan. *Automatic Hypertext Construction*. PhD thesis, Cornell University, January 1995. Also technical report TR95-1484.

[2] James Allan. Relevance feedback with too much data. In *Proceedings of ACM SIGIR*, pages 337–343, 1995.

[3] James Allan. Incremental relevance feedback for information filtering. In *Proceedings of ACM SIGIR*, pages 270–278, 1996.

[4] James Allan. Building hypertext using information retrieval. *Information Processing and Management*, 33(2):145–159, 1997.

[5] James Allan, Jamie Callan, Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongmin Shu. Inquery at TREC-5. In *Fifth Text REtrieval Conference (TREC-5)*, pages 119–132, 1997.

[6] James Allan, Jamie Callan, W. Bruce Croft, Lisa Ballesteros, Donald Byrd, Russel Swan, and Jinxi Xu. Inquery does battle with TREC-6. In *Sixth Text REtrieval Conference (TREC-6)*, pages 169–206, 1998.

[7] Peter G. Anick and Shivakumar Vaithyanathan. Exploiting clustering and phrases for context-based information retrieval. In *Proceedings of ACM SIGIR*, pages 314–323, 1997.

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

[9] Matthew Chalmers and Paul Chitson. Bead: Explorations in information visualization. In *Proceedings of ACM SIGIR*, pages 330–337, June 1992.

[10] Jonathan D. Cohen. Drawing graphs to convey proximity: An incremental arrangement method. *ACM Transaction on Computer-Human Interaction*, 4(3):127–229, Spetember 1997.

[11] W. Bruce Croft. *Organising and Searching Large Files of Documents*. PhD thesis, University of Cambridge, October 1978.

[12] David Dubin. Document analysis for visualization. In *Proceedings of ACM SIGIR*, pages 199–204, July 1995.

[13] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software–Practice and Experience*, 21(11):1129–1164, 1991.

[14] Donna Harman and Ellen Voorhees, editors. *The Fifth Text REtrieval Conference (TREC-5)*. NIST, 1997.

[15] Donna Harman and Ellen Voorhees, editors. *The Sixth Text REtrieval Conference (TREC-6)*. NIST, 1998.

[16] Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of ACM SIGIR*, pages 76–84, August 1996.

[17] M. Hemmje, C. Kunkel, and A. Willet. LyberWorld - a visualization user interface supporting fulltext retrieval. In *Proceedings of ACM SIGIR*, pages 254–259, July 1994.

[18] Jurgen Koenemann and Nicholas J. Belkin. A case for interaction: A study of interactive information retrieval behavior and effectivness. In *Proceedings of Conference on Human Factors in Computing Systems*, pages 205–212, 1996.

[19] Anton V. Leouski and W. Bruce Croft. An evaluation of techniques for clustering search results. Technical Report IR-76, Department of Computer Science, University of Massachusetts, Amherst, 1996.

[20] Anton Leuski and James Allan. Interactive cluster visualization for information retrieval. In *Proceedings of ECDL'98*, pages 535–554, September 1998.

[21] Anton Leuski and James Allan. Evaluating a visual navigation system for a digital library. *International Journal on Digital Libraries*, 1999. Forthcoming.

[22] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[23] Russel Swan and James Allan. Aspect windows, 3-d visualizations, and indirect comparisons of information retrieval systems. In *Proceedings of ACM SIGIR*, pages 173–181, 1998.

[24] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979. Second edition.

[25] Jinxi Xu and W. Bruce Croft. Querying expansion using local and global document analysis. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval*, pages 4–11, 1996.