Bubbleworld

A New Visual Information Retrieval Technique

Christopher Van Berendonck **Timothy Jacobs**

Chris.Vanberendonck@defence.gov.au **Timothy.Jacobs@afit.edu**

Air Force Institute of Technology
Wright-Patterson Air Force Base
Ohio 45433 USA

Abstract

Visualisation has significant advantages over traditional textual lists for improving cognition in information retrieval. To realise these advantages, we identify a set of cognitive principles and usage patterns for information retrieval. We apply these principles and patterns to the design of a prototype visual information retrieval system, Bubbleworld. In Bubbleworld, we apply a variety of visual techniques that successfully transform the internal mental representations of the information retrieval problem to an efficient external view and, through visual cues, provide cognitive amplification at key stages of the information retrieval process. We enhance the knowledge acquisition process by providing query refinement and interaction techniques that facilitate the specification of complex search schemas and integrate these with the mechanisms to incorporate predefined ontological models. We then attempt to validate the new visual techniques through empirical user trials to gain a better understanding of its benefits.

1 Introduction

Traditional information retrieval techniques typically return a textual list of documents ranked from most relevant to least relevant based on the determination of a search algorithm (or model). Although the search algorithms are improving, resulting in better relevance ordering, they are far from perfect. Not only are large lists of irrelevant documents frequently presented to the user but also, limited clues are presented for determining how best to adjust the query to improve the result. Even if search models could be improved to return only relevant documents, current studies suggest that interaction with an information collection through foraging alone provides only part of the knowledge gathering process (Ellis 1989).

It is widely accepted that visualization of information can enhance cognition. A number of prototype visual information retrieval systems have been developed to validate this belief (Hemmje 1994, Spoerri 1993 and others); however, results are inconclusive and these systems have not found widespread use.

In this paper we introduce a new visual information retrieval technique based on the identification of a set of cognitive principles and usage patterns for information retrieval. The new technique successfully transforms the internal mental representations of the information retrieval problem to an efficient external view, thereby allowing cognitive amplification at key stages of the information retrieval process. We provide query refinement and interaction techniques that facilitate the specification of complex search schemas to enhance the knowledge acquisition process. The mechanisms to incorporate predefined ontological models are integrated into the new technique to allow automatic (or semi-automatic) schema expansion. The Bubbleworld prototype system is then user tested against a traditional text based retrieval interface for empirical comparison.

2 Cognitive Principles

To maximise the cognitive benefits of our graphical representations, we based our design on a set of cognitive strategies for information retrieval. These strategies, based primarily on the work of Scaife and Rogers (1996), are summarised here.

- Restrict the interpretations of the external model to enforce a tight coupling between it and the information retrieval domain space.
- Provide computational offloading through the careful selection of graphical representations and the associated interaction and manipulation models.
- Provide a graphical representation of the problem that maps closely to the internal mental models of most users.
- Constrain the temporal and spatial domains to enhance the correct inferences about the external representations.

3 Usage Patterns

Focusing on the information retrieval domain, we identified typical usage patterns to further guide our design. These usage patterns are derived from experiments conducted by Cunningham, McNab and Boddie (2000) using traditional informational retrieval techniques:

- Users rarely amend default settings.
- Queries tend to be short and simple.
- Clusters of common query terms are apparent.

- Users have difficulty selecting terms with the appropriate specificity.
- Users submit a high percentage of malformed or erroneous queries.
- Users submit few sequences of queries, spend short periods searching and are unlikely to repeat the visit.
- Users refine their query.
- Few, if any, documents are viewed from the returned list and those that are can be found at the top of the list.

From these usage patterns we derived a set of design corollaries considered important to visual information retrieval (Van Berendonck 2002).

4 Bubbleworld

Every information retrieval system requires a mechanism to specify a need, through a query language, and then visualise the results using an external representation. The two need not be the same system, however, most generally are. Visual techniques are powerful in that an intermediate view of the relationships between the query and the result is also possible. The rest of this section explores this concept further by developing a visual technique that does precisely this. It is based on the design corollaries and usage patterns discussed above.

4.1 Query Specification

The circle provides a simple uncluttered view that is capable of displaying nodes (or key index terms) symmetrically with optimal spacing. The nodes are spaced at $\frac{2\pi n_i}{N}$ where n_i indicates the position of the key index term and N is the number of index terms to be displayed. The benefit of the circle here is that the structure does not change as N changes, hence it will support a mutable query phrase.

The query phrase can be specified as a typed written sentence, which is then broken into keyword nodes. These nodes can be placed symmetrically around the circle to represent the keywords of the query. An example of this is shown in Figure 1. Using this view, the query can be broadened by removing nodes from the structure, or narrowed by adding nodes to the structure.

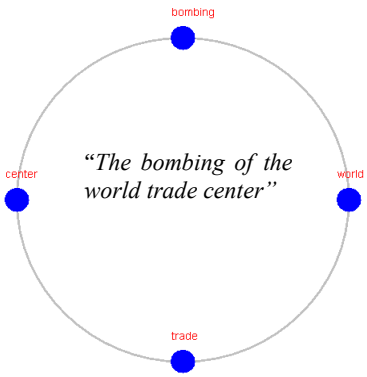


Figure 1. Simple Query Specification

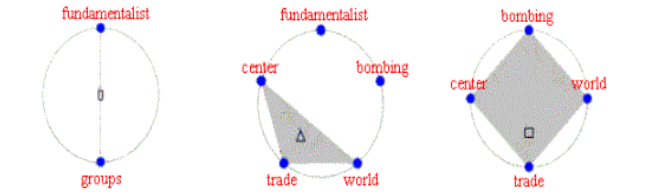
4.2 Visualising Relationships

Coding of the documents returned allows users to rely on their visual reasoning skills to quickly discern the relationships between the icons (or documents) and the keyword nodes of the structure. The key here is to identify coding principles that are understood by most users, that is, the symbols, icons and their visual organisations are intuitive and familiar. As well, the relationship between the graphical elements must also be intuitive if information is to be inferred from this relationship. InfoCrystal (Spoerri 1993) uses simple geometrical shapes such as rectangles, triangles and squares to represent a collection of documents with a specific set of keyword relationships and spatial positioning to indicate a particular characteristic of that relationship. The coding principles used by Spoerri fit well into this technique because not only are geometrical shapes (and the number of sides that make up that shape) an inherent part of early childhood learning and therefore familiar to all, the circle represents an element of this set which maintains consistency in the representation. As well, the spatial relationship between the icons and the nodes (that is, the closer the icon is to the node the greater the influence that node has on the icon) is also a familiar concept since it represents a basic physical phenomenon. The following is a list of the coding principles used in this research:

- **Shape**: The shape of the icon indicates the number of keyword nodes that the contents are associated with. That is, a circle can show relationship to one node, a rectangle to two, triangle to three, square to four and so on.
- **Proximity**: The closer the icon resides to a node, the stronger is the relationship between the icon and that node.
- **Size:** The greater the size, the higher the ranking of the icon.
- **Text**: A number resides next to each icon to indicate the number of documents associated with the icon. The node can also display a number to indicate how many documents are associated with it.
- **Colour**: Colour is used to show information on demand; such as which icons are associated to a particular node or group of nodes, which icons have previously been investigated, and so on.
- **Shading**: Regions are shaded to show which combinations of nodes are related to the icon.

An example of shape, proximity and shading techniques are shown in Figure 2. Both Figure 2(a) and Figure 2(b) show approximately equal weighting of keyword terms, whereas Figure 2(c) shows a heavier weighting on the node 'trade' with equal weighting of the terms 'world' and 'center'.

The vector model was used as the information retrieval search engine primarily because of its simplicity in design and high recall ratios achieved. That aside, it provides a convenient tool to apply two-dimensional spatial positioning on individual documents because essentially,



(a) Shading, Shape, Position Two Relationships (b) Shading, Shape, Position Three Relationships (c) Shading, Shape, Position Four Relationships

Figure 2. Demonstration of relationship coding

they are vectored into position. The equation, from Baeza-Yates and Ribeiro-Neto (1999), is shown here for clarity:

$$sim(d_j, q) = \frac{\sum_{i=1}^{t} w_{i,j} * w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} * \sqrt{\sum_{j=1}^{t} w_{i,q}^2}} \quad \textbf{(1)}$$

In this case, the $sim(d_j,q)$ represents the similarity value for document d_j when applied to query q. The result is a single value that can be used to rank documents in a descending relevance list. Of more importance here, is how each keyword index weight can be broken down into normalised vector components as shown below:

$$\alpha_{i,j} = \frac{w_{i,j} \times w_{i,q}}{\max_l w_l} \quad \textbf{(2)}$$

where $a_{i,j}$ represents the normalised scalar vector component i of document d_j and $max_l w_l$ is the maximum value over all components.

The question now is how to apply the scalar components to a vectored position. A simple but effective technique is to consider each node, n_i, to have an attractive force equal to its normalised scalar component. The icon then settles at the position of equilibrium between these nodes. This can be visualized simply by considering the icon being connected to each node via invisible springs with a tension equal to the attractive force. The icon will rest at the point were the resultant force is zero.

There are two points worth mentioning with this spatial positioning technique. First, only relative strength relationships are visible. Consider Figure 2(a) above. The position of the icon indicates an equal attractive force between 'fundamentalist' and 'groups', but there is no indication of relative strength of this force. Therefore, this technique must be supplemented with additional coding to provide a visual cue on absolute strength. This is achieved by incorporating size coding to represent absolute strength of the attractive force. This is easily implemented by relating size to $\sum a_{i,j}$, but note the complication. Documents with exact term relationships and spatial positioning, but different size must now be either differentiated or ignored. That is, either multiple icons of different sizes will try and occupy the same space, or only one icon will exist but the ranking information of the document is lost. Since trying to map many icons to the same space cannot work, for obvious visual clarity issues, the documents must be grouped into a single icon where the size of the icon represents the highest ranked document within the icon collection. This solution works for the following reasons. First, icons can still be compared against each other to determine the rank order, the only occlusion occurring is the exact number of documents that are ranked higher in one icon compared to that of another. Second, documents are still in rank order within the icon so document level ranking is not lost just relative ranking across the whole collection. Third, the probability of this occurring reduces significantly as the order of the shape increases and, as the order of the shape increases so does the probability that the ranking will be higher. In other words, higher-ranking documents tend to be located in icons with diminishing collection sizes.

The second point worth mentioning is that there is no immediate cue to which nodes provide an attractive force to the spatial positioning of the icon, and which do not. Again, an additional coding technique has to be included here. An object-in-focus technique is used. The icon in focus develops a shaded polygon with each vertex represented by a node having a positive influence on the position of the icon. In this way, the user can not only discriminate which nodes have influence on the icon, but also gains clarity from the relative positioning of the icon to each of these nodes. Figure 2 shows this technique in practice.

As the number of documents being mapped to the two-dimensional structure increases, two effects become more prominent. First, and perhaps most obvious is that the clutter of icons rapidly increases to the point that individual icons can no longer be distinguished from one another. The second point is that icons, regardless of the nodes influencing their position, can be mapped to the same spatial point. In this case, without a third dimension, the icons would perfectly overlap each other causing occlusion.

A natural solution to this problem is to treat groups of documents as repositories within single icons; however, care must be taken to maintain the integrity of the icon representation. To do this, the merging of a document into an icon is forced to obey the following simple rule:

Document d_j can merge into icon I_m containing document d_k iff:

$$\sum_{i=1}^{t} \alpha_{i,j}\phi_{i,m} = \sum_{i=1}^{t} \alpha_{i,k}\phi_{i,m} \; where \; \alpha_{i,j}, \alpha_{i,k} > 0 \quad \textbf{(3)}$$

where $\phi_{i,m}$ represents the angle from the position of Icon I_m to node n_i

The merging rule groups documents into icons that have identical term relationships and spatial positioning and excludes those that do not. Icon occlusion is not completely prevented using this method, however, the probability of two documents with differing relationship criterions being spatially mapped to the same exact point is significantly less than that of two documents with the same relationship criterions.

The problem of icon clutter remains even with the grouping scheme discussed above. This is because only perfect matches in relative relationships can be grouped.

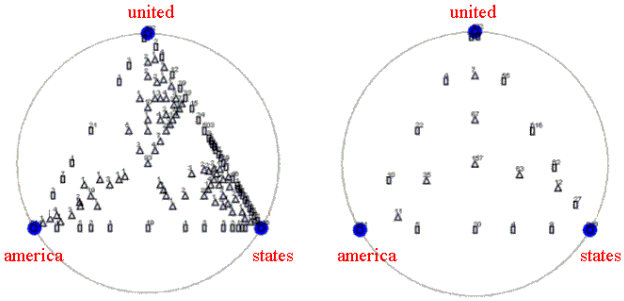
To alleviate this problem, a neighbourhood zone around each existing icon is defined. If a document containing the same nodal relationships as an existing icon is placed within that icon's neighbourhood zone, then the document can be merged as before. This essentially reduces the restrictions of the document merging rule above to incorporate near hits. The merging rule therefore becomes:

$$\left| \sum_{i=1}^{t} \alpha_{i,j} \phi_{i,m} - \sum_{i=1}^{t} \alpha_{i,k} \phi_{i,m} \right| \leq r \qquad (4)$$

where $\alpha_{i,j}, \alpha_{i,k} > 0$

and r is the radius of the neighbourhood zone.

The size of the neighbourhood zone is known here as the C-Factor, for the consolidation of icons. The diagram in Figure 3 shows an example with and without the use of neighbourhood zones. The use of neighbourhood zones does distort the spatial positioning of the documents but individual fidelity is traded for clarity. It is not expected that this loss in fidelity will significantly alter the users ability to reason about the relationships of the documents to the nodes. It will, however, increase the repository size of the icon (number of documents merged into a single icon) as less and less detail is displayed.



(a) Without Neighborhood Zones (b) With Neighborhood Zones (high C-Factor)

Figure 3. Neighbourhood Zoning of Icons

4.3 Filtering through schemas

Perhaps the most fundamental argument for a visual information retrieval system is the ability to visualise and navigate search schemas whilst provide cognitive off-loading to the external representation.

Schemas are visualised here as a partial representation of structure nested on a particular parent object The contents (documents) of a parent bubble are allowed to replicate through a node join into a child bubble causing them to undergo a transformation into new icons. This is the filtering process that occurs as the document collection transgresses across the schemas. The filtering can be achieved in a number of different ways. First, and perhaps most obvious, is the boolean filter. In this technique, documents only replicate through the node if the documents in the parent bubble have a syntactic match to the node. For example, S_2 of Figure 4 would have documents from S_1 that have the keyword 'President'.

The second method of filtering is phrase matching. This is similar to boolean filtering except replication of documents through the node is restricted to particular phrases being located within the documents. For example, "white house" would restrict replication to only those documents with the phrase "white house" located in the text. The third method of filtering is proximity matching which is an extension of the previous two methods. In this case, a proximity distance between the key words of the phrase can be specified to remove some of the syntactic restrictions that phrase matching places on the outcome. For example, 'Computer Networks' with a specified proximity distance of one, would allow documents containing the phrase 'Computer Support Networks' to be replicated through the node. To visualise this, a period between the terms would specify the proximity distance, so 'Computer .. Network' would specify the proximity distance as two, 'Computer … Network' as three and so on.

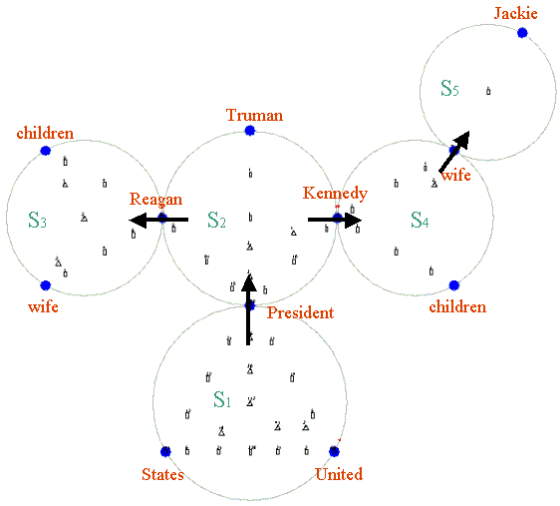


Figure 4. The formation of collection sets

4.4 Interaction Model

With any medium to large-scale collection size, one of the most important considerations is how best to hide details through high level abstract views of the data. This, however, relies on an intuitive interaction model to allow the user to 'drill down' into the data until the primitive view is reached (in this case, the document itself). In this section, the interaction model is discussed further.

The interaction model is limited to three distinct environments: the exploratory view, the icon view and the document view. The exploratory view provides the overview of the document collection depicted by the schemas that are defined. That is, not all documents are seen at all times, but only those that fit into the current need of the user. The icon view provides a means to further explore the repository collection of an icon, and the document view is a view of the document itself. Figure 5 depicts the three interaction views.

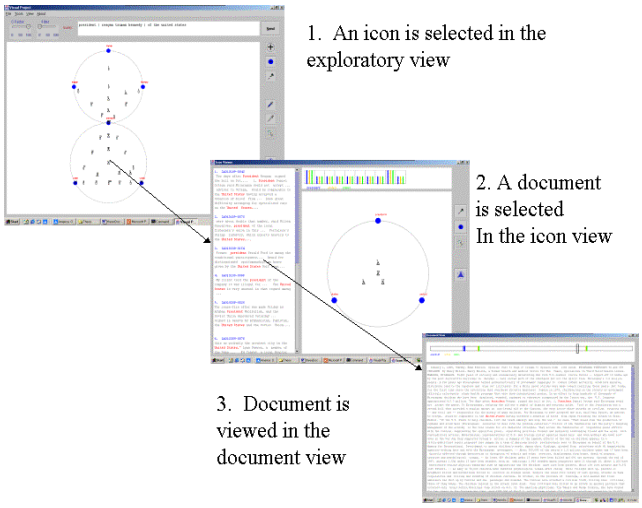


Figure 5. The three interaction views

The interaction link between the exploratory view and the icon view is the icon itself, so by selecting the icon the user moves from the exploratory view to the icon view.

At this stage the collection view is limited to the repository of documents within the icon. The size of this subset collection, however, may still be quite significant especially if the initial collection size is large. Therefore, just as it was not appropriate to view individual documents in the exploratory view, it may not be appropriate to view all the documents individually in the icon view. To overcome this, the icon view is designed as an abstract manipulation layer that sits between the view of the collection and the view of the document.

The abstraction layers work as follows. The icon view layer is initially created around the icon selected in the exploratory view and only the nodes that had influence on the icon are reconstructed. An example of this is shown in Figure 6(b). In this view, additional nodes can be added (or removed) to break open the icon(s) and redistribute them, as shown in Figure 6(c). When a particular icon of interest is identified (by relative positioning) then it too can be selected causing an additional layer of abstraction to be created around that icon, as shown in Figure 6(d). The document collection contained within this newly selected icon is passed through the layer, and the view is reconstructed around this new icon. This manipulation phase can go on for an indefinite number of abstract layers, and the layers themselves can be transitioned up or down as they are created. This creates a browse and filtering environment, which can help locate a particular document for viewing.

The document view is reached when transgressing an arbitrary number of abstraction layers leads to a document of interest. For example, in Figure 6(d), since each icon contains only one document in its repository, selecting either would cause a transition from the icon view to the document view. In this case, only two abstraction layers were needed. At this stage, and because of the interactive filtering that has occurred, it is probable that the document being viewed is of some relevance, and therefore of some interest to the user. Unfortunately, like any information retrieval system, there are no guarantees that the document is indeed relevant since the complexity of the lexical grammar of our language allows for redundancy of meaning using otherwise similar groups of words. Therefore, the document viewer represents the end of only one search path and perhaps the beginning of another. The search/browsing continues by transgressing from this point up the icon view abstraction layers until other icons of interest are located.

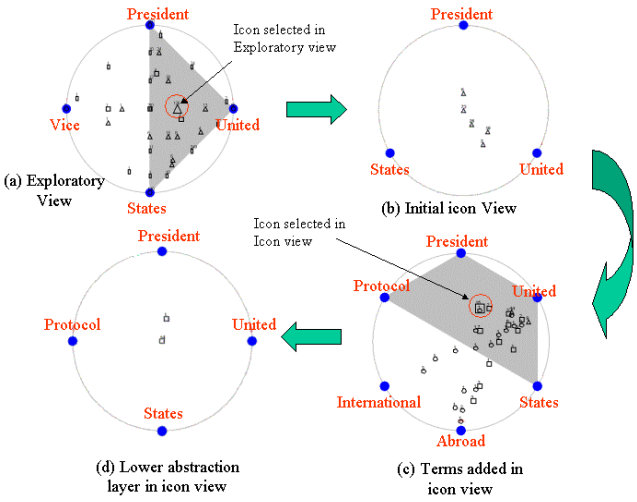


Figure 6. Transitions through the Icon View abstraction layers

4.5 Ontology Models

Exploratory browsing is a fundamental technique used in retrieving textual information from large information sources. The process of browsing is complex, however, and it is not always intuitively obvious to the user on how to proceed at any given point. For instance, if the user is not an expert in the domain they are searching, they may have some difficulty in expressing their need in the language provided to them by the system they use. Even if they could express this need, the user may wish to organise related information into common schemas, but exactly how these schemas should be specified may not be known.

Ontology modelling has been identified as an important tool to be used in knowledge based systems and information retrieval systems, since they provide semantic structure to an otherwise unstructured information environment (Lenat and Guha 1990, Gaines 1997). Of importance for this research is how this semantic structure can be integrated into the need specification process.

Consider the example of an aircraft structure. A common method of displaying this type of information might be as a connected tree diagram shown in Figure 7. In this diagram, each sub-category is represented as a child to the parent category with the relationships specified by lines and arrowheads.

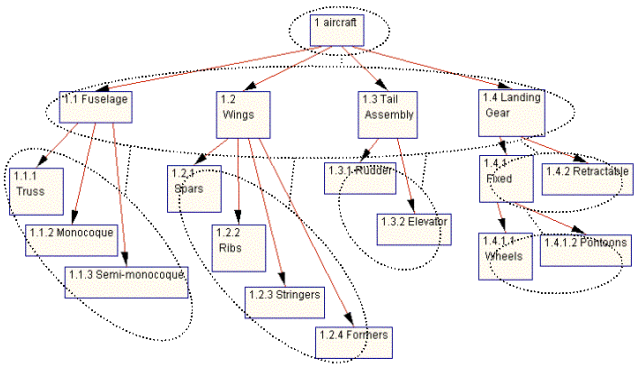


Figure 7. Aircraft Ontology with Identified Categories

To integrate this diagram into the bubble schema, ovals are identified as bubbles and the connection between the ovals as the nodes. Figure 8 shows the completed transformation of the aircraft ontology into this representation. Since ontology models can be easily converted into this form, it follows that automatically expanding a user's schema is also possible. If ontology models can be pre-defined based on expert knowledge of the domain, then automatic schema expansion can provide a means to guide the user into areas of the collection that may otherwise have been hidden from the search.

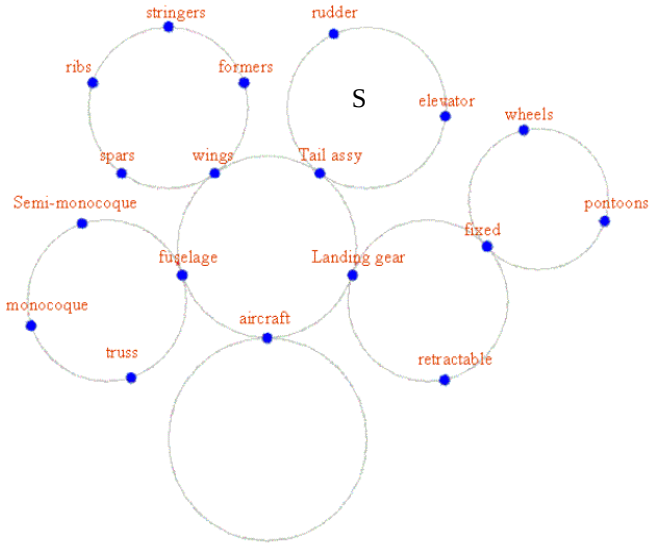


Figure 8. Example Transformation of the Ontology Model of an Aircraft

One of the many benefits of ontology models is that they are specified a priori, based on expert domain knowledge of the information collection. This means that not only can structural information about a particular model be specified, but the opportunity to specify semantically equivalent terminology is also possible. Adding a thesaurus capability to the model is essentially adding the OR portion to the boolean model previously described. Consider the aircraft ontology example again in Figure 8. If 'empennage' is added to the thesaurus collection of 'Tail assy' then the documents found in the set *S* would satisfy the boolean equation: *Aircraft AND (Tail assy OR empennage).*

5 Evaluation

New information retrieval systems are generally tested against standard retrieval strategies in controlled experiments. The experiments are generally set around standard information retrieval test collections (such as TREC), and performance is evaluated based on conventional techniques such as precision and recall. There are some misgivings with this technique that cause doubt as to the validity of the results. For instance, the test queries are generally constructed to represent high specificity and syntactical matching resulting in 'information tasks' rather than 'navigational tasks'. Here, 'information tasks' refer to those problems that can generally be solved with a single query. 'Navigational tasks' are those that require multiple queries to formulate enough about the schema of the problem such that it can be answered.

Attempting to test visual information systems in this manner has many problems. First, in most experimental environments in which these systems are tested, only naïve or briefly trained test subjects are used. Therefore, many of the benefits of the visual information retrieval system tend not to be effectively utilised which leads to poorer performance and satisfaction ratings than simple textual interfaces. All new tools require some lead-time to use effectively. Second, many visualisation techniques are designed as exploration tools that 'help' navigate users through the mass of information available. To this end, the tools tend to perform better in solving 'navigational tasks' rather than 'information tasks', which contradicts current precision and recall evaluations. The difficulty here is how can 'navigational tasks' be specified into a query without it becoming an 'information task'? The queries themselves either become designed around the textual interface or simply become non-representative of real world problems. Third, the metrics that should be used for the evaluation are not always clear. For instance, when evaluating the performance of the visual information retrieval system, should the visual retrieval technique be removed from the visual interface, or should the system as a whole be evaluated? Should the system be evaluated against usability, task performance or both, and how does this affect the results when directly comparing systems?

5.1 Query Tasks

In formulating the queries for the experiment, it is important to understand the types of queries that make up reasonable representations of 'real world' situations. For instance, asking a 50-85 word query on a search engine when users are reluctant to type more than 3-5 words at a time is not reasonable. Equally, queries that tend to have perfect syntactical matching with the documents generally occur less than 20% of the time (Furnas, Landauer, Gomez and Dumais 1987), so although this is

useful sometimes, again it is not a frequently occurring situation.

For this experiment, queries were broken down into four types, based on syntactic matching and term specificity. Syntactical matching is how well the syntax of the words of the query match the document considered relevant to that query. Term specificity is how well the query semantically specifies the problem. The four query types are listed and described below:

- **Type 1: Good term specificity and good syntactical matching.** Requires limited browsing. Usually the document is well scored and appears high in the retrieved relevance list.

- **Type 2: Poor term specificity and good syntactical matching.** Returns good hits on the documents from the query but quite often scores poorly because of poor specificity. Queries of this type generally fall lower in the relevance list and require some browsing to find.

- **Type 3: Good term specificity and poor syntactical matching.** Good description of the problem, however, only minor matches are made with the document. Again, scoring on this document is poor unless some additional techniques are used to match semantically similar terms.

- **Type 4: Poor term specificity and poor syntactical matching**. Could be the first step in the information gathering phase. Documents returned are generally not relevant but may help the user to refine the query into another type.

Type 1 queries are not frequent and generally only occur if the user knows the problem domain well. These queries are well specified with good syntactic matching.

Type 2 queries represent a typical query that a user might issue if they know their subject matter. That is, the query is not well specified but the keywords of the query will provide a good syntactic match. The problem with this type of query is that syntactic matching does not guarantee semantic matching, and if the keyword being matched is popular in the collection, the returned relevance will either be poor or hidden by irrelevant documents.

Type 3 queries represent perhaps the most common type issued if the user had a good understanding of what they wanted, but had little knowledge of the specific terms that may be used. That is, the problem is generally well specified but there is poor syntactic matching of the keywords.

Type 4 queries are those that have a very poor syntactic match and are not at all well specified. This type of query is generally produced when users begin with a vague idea of what they are searching for and with no domain knowledge to help with syntactic matching. Needless to say, this type of query generally never produces a good relevance list for the user. It may, however, help the user to re-specify the query iteratively until the query is of one of the other three types. For example, a user specifying the term ‘wireless’ may not have a good relevant list of documents returned, but in browsing, the same user may discover the terms ‘radio’ or ‘transistor radio’ and re-query the system with more success.

5.2 Experiment

A prototype information retrieval system was implemented to compare the Bubbleworld technique against a representational text-based information retrieval system. The graphical interface was implemented in the JAVA programming language (version 1.3.0) and deployed on a standard desktop computer (Intel PIII, 500MHz 256K RAM PC running Windows 2000 OS). The Bubbleworld GUI is shown in Figure 9.

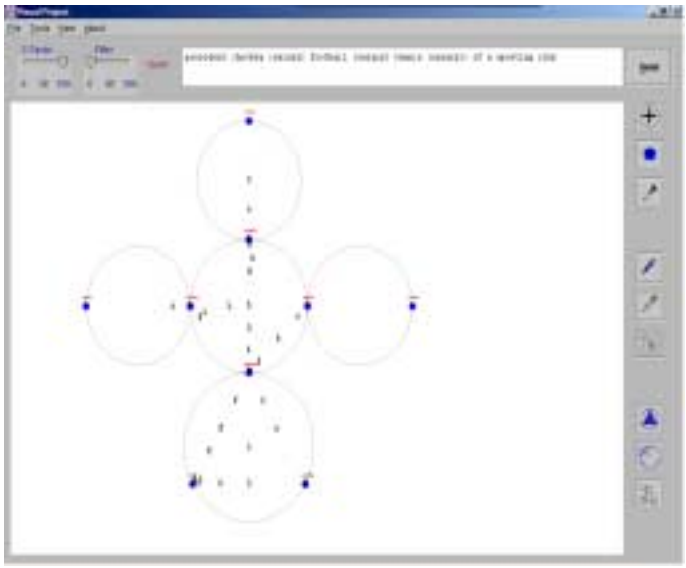


Figure 9. The Bubbleworld GUI

For this experiment, a purpose built textual interface was incorporated within the graphical interface software. This had a similar form and function as a traditional textual interface that one might reasonably expect to see used commercially. This included relative matching indication (based on the same vector model search engine), ranked summary document listing, and keyword highlighting, but did not include phrase matching or boolean filtering.

Integrating the textual and graphical interface provided a controlled experiment, where the independent variables are essentially the interfaces themselves. This mitigated the risk of uncontrolled (and undesirable) variables having an effect on the experiment and perhaps biasing the results obtained.

The document collection used for the experiment was a subset of the LATIMES data available on disk 5 of the TREC data collection. The collection of 4457 documents was indexed a priori to produce 74516 unique keywords. This collection was selected based on its neutral subject content and its unilateral relevance. The size of the collection represented a trade-off between the time required to index the collection, system performance (based on the test platform described above), and a reasonable representation of an information source.

Twenty test candidates were obtained from volunteer students in the Engineering and Management School of the Air Force Institute of Technology (AFIT). Each candidate was selected for either the textual interface component of the experiment or the graphical interface component of the experiment based on 'walk in' random selection. Prior to the experiment commencing, the candidate was required to view a short video describing the purpose of the experiment, functions of the interface they were about to use, and how the experiment was to be conducted. They were then allowed to experiment with the interface until they were ready to proceed. This generally took a few minutes for the textual interface and 5-10 minutes for the graphical interface.

The test candidates where required to answer twelve information retrieval questions across the four query types discussed above. They achieved this by using their selected interface to find document(s) that best answered the question; the queries they chose to use was up to the candidate. They were told that both time and accuracy were of equal importance. The questions were presented in sequential order and timed from initial presentation until the candidate chose to move to the next question. Accuracy was measured by matching the candidates recorded document number to predetermined correct solutions.

5.3 Results

Figure 10 shows the direct comparison between the average time results of candidates using the textual interface versus the average results of the candidates using the graphical interface on a per question basis. Question 12 was removed from the experiment because of an inter-dependency problem with how Question 10 and 11 were answered.

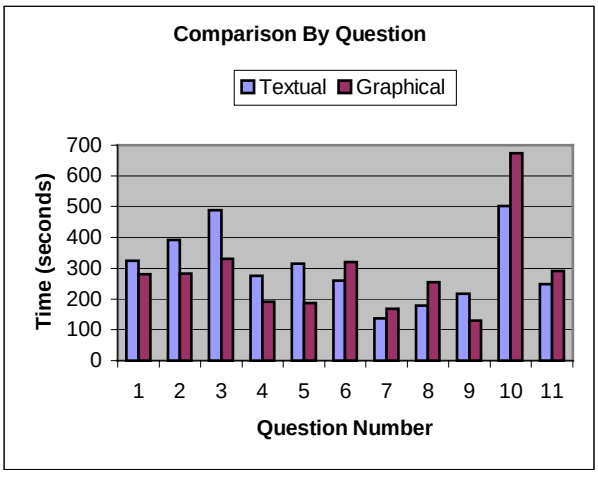


Figure 10. Comparison of Average Times by Question

Figure 11 shows the direct comparison between the average time results of candidates using the textual interface versus the average results of the candidates using the graphical interface for each query type.

Testing at the $\alpha = 0.05$ level of significance, the two-tailed test statistic for average times between the samples suggests that there is not sufficient evidence to indicate a difference in the mean between the two results. Therefore, there is no statistically significant difference between the two sample mean times. The results show, however, that there is a significant decrease in time required to answer type 1 and type 3 queries for the graphical interface when compared to the textual interface, a comparatively equal result for type 2 queries and a significant increase in time to answer type 4 queries.

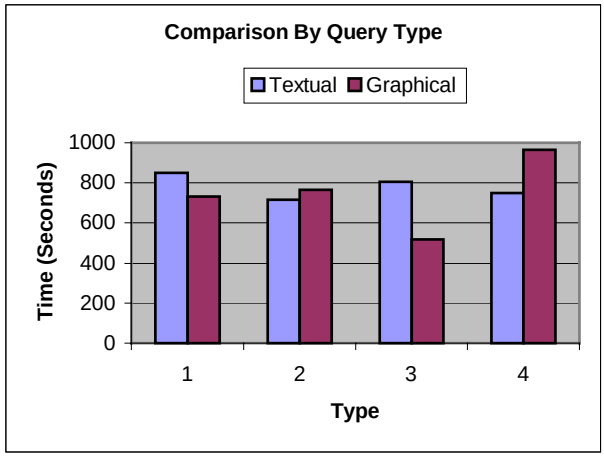


Figure 11. Comparisons of Average Times by Query Type

The results of the type 1 queries are interesting. These queries represent those that have good term specificity and good syntactic matching (as described above) which generally allowed the documents to rank well in the relevance list. It was initially thought that documents ranking high in the document list would be found quicker than selecting icons from the graphical display, however this was not the case. The reason for this was that documents matching the type 1 queries were displayed with higher order icons (i.e. squares, pentagons), which are easily distinguishable across the collection. Therefore, the number of documents that were scanned for content (by the user) was generally fewer than in the textual interface. For example, in the ranked list of the textual interface, a document with five keyword hits may rank lower than a document with four resulting in the later document being scanned before the former.

The type 2 queries showed similar results for both the graphical and the textual interface. Type 2 queries are those with poor specificity but good syntactic matching. This was to be expected since the relevant documents are easily 'lost' in the noise of the collection. For instance, in the textual interface, the document may rank low in the relevance list especially if the small number of keywords that match are popular in the collection. A similar problem occurs in the graphical interface. The document can easily get lost in the noise of lower order icons (say rectangles) that have large document collections within them. With experience, users may have been able to reduce the search time by considering spatial location of the icon more closely than they did.

Type 3 queries showed the most significant reduction in time between the graphical interface and that of the textual interface. Type 3 queries are those that contain good specificity but poor syntactical matching. Again,

this generally resulted in the relevant document being ranked poorly across the collection. This type of query generally required the user to manipulate the query, perhaps incrementally, until keywords are struck that allow the document to rank higher in the relevance list. The results showed the users of the graphical interface were able to see the results of this incremental change more clearly than those using the textual interface. That is, words that had little or no relevance to the document collection could easily be distinguished and therefore replaced with other words until the document ranked well enough to be promoted to higher order icons, thereby becoming more visible. Those using the textual interface did not have this luxury and relied on having to scan a large list of documents for every incremental change of the query, resulting in considerable frustration and fatigue for the user.

The last of the query types, Type 4, provided interesting results as well. These queries required multiple schemas to piece together the overall solution. This type of query resulted in a 28% average increase in time for those candidates using the graphical interface to those using the textual interface, which by any means is a significant result. The reason for this came down to inexperience by the user. The users did not generally attempt constructing multiple schema queries in the graphical interface, or if they did, it was not done effectively. In fact, most of the users that attempted this problem in the graphical interface tried to solve the problem using a single schema (single bubble). By guessing the answer they attempted to remove the browsing component of the problem and go directly to the search component. The problem with this technique is that it cluttered the search space and made it difficult for the user to ascertain what was relevant, and what was not.

Figure 12 shows the direct comparison between the average accuracy results of candidates using the textual interface versus the average results of the candidates using the graphical interface by question. Figure 13 shows the same comparison but this time by question type.

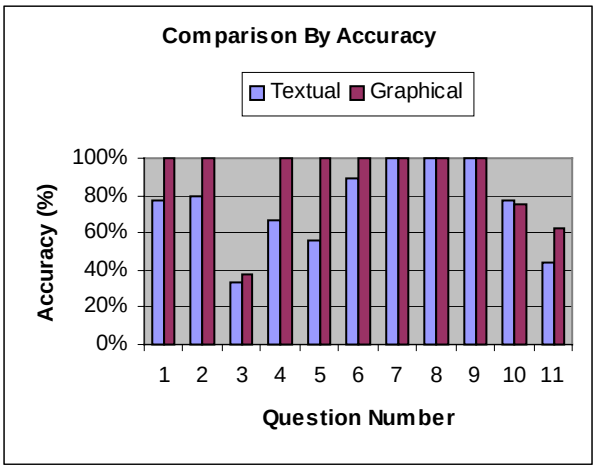


Figure 12. Comparison of Accuracy by Question Number

Testing at the $\alpha = 0.05$ level of significance, the two-tailed test statistic for the average accuracy values between the samples suggests that there is sufficient evidence to indicate a difference in the mean between the two sample mean accuracies. Furthermore, it is a lower tail alternative providing support that the mean accuracy for the results of the graphical interface is statistically superior to that of the results for the textual interface

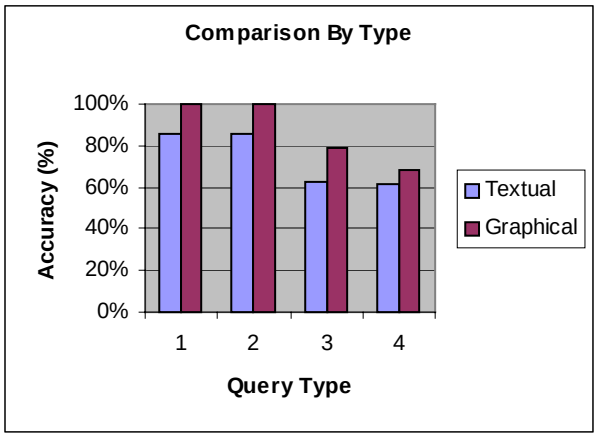


Figure 13. Comparison of Accuracy by Question Type

It is clear that the accuracy of all query types for the graphical interface is significantly higher than that of the textual interface. Of particular interest here is the fact that both Type 1 and Type 2 queries show an accuracy of 100% for the graphical interface. This shows that with good syntactic matching, the relevant documents are promoted well enough to higher order icons to be found. The only variance here was the time taken to find the document.

5.4 Further Discussion

A reasonable conclusion to draw from the results above is that users of the graphical interface appear to be more willing to continue searching for the solution as long as they think they have a chance to find it. The graphical interface promotes this by showing the whole collection space (as pertained to the query). While relevant icons are still not searched, the user is willing to continue. Contrast this to the textual interface where, after searching through tens of documents, the user is not sure whether continuing the search would help, or whether there really is no answer.

Another observation made with the study is that users tend to experiment more with the queries in the graphical interface. The user logs revealed that, on average, users of the graphical interface issued 46 queries across the 12 questions compared with 32 queries for the textual interface. This was probably due to the fact that users could obtain more meaningful feedback from the graphical display than from the textual list and were therefore more willing to try new queries. This is one explanation as to why the accuracy varied between the two systems so significantly. The fact that the average time difference between the two groups did not change significantly, even with an increase in the number of queries issued, suggest that users were able to make quicker inferences from the results of each query than they were from the textual list.

As discussed before, there is a significant increase in the average time (+28.5%) to solve Type 4 queries in the

graphical interface over the textual interface. Although it was thought that the graphical interface would outperform its counterpart in these queries, the results are not surprising. Constructing the nested queries represents the most difficult task required of the user, and knowing just how to nest the queries effectively takes a little experience. Investigating the user logs, of the 10 candidates that used the graphical interface, only five attempted nested queries as instructed in the training video, and of those, only two did so effectively. The rest attempted to answer the question in a similar fashion to those using the textual interface. That it, guess the answer and search for documents using a single schema approach. The two candidates that did use the nested queries effectively answered the question correctly and did so with a 16.9% improvement in time over the overall average for the group using the graphical interface. If Type 4 questions are removed from the analyses of mean average times and the two groups again compared, then there is a real mean average decrease in time of 15% from the group using the graphical interface to the group using the textual interface. From this discussion, it is reasonable to postulate that an increased effort in training (more than the 25 minutes afforded in this experiment) would result in a reduction in the mean average time of the users. Given that the test candidates can be considered expert computer users, it is unlikely that the same additional training in the textual interface would result in any increased performance.

6 Summary and Conclusion

Bubbleworld integrates a variety of visual techniques for exploring large document collections to locate relevant information. Visual techniques are selected and implemented based on a set of cognitive principles and usage characteristics tailored to information retrieval. Preliminary experimental results indicate that these techniques lead to minor improvements in retrieval time and significant improvements in retrieval accuracy over textual document retrieval lists. As our preliminary results are not entirely conclusive, we intend to apply Bubbleworld to common information retrieval tasks to gain a better understanding of its benefits.

7 Acknowledgment and Disclaimer

This work is funded in part by the United States Air Force Office of Scientific Research. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defence, or the United States Government.

8 References

BAEZA-YATES, R. and RIBEIRO-NETO, B. (1999): Modern Information Retrieval. Addison-Wesley.

CUNNINGHAM, J, MCNAB, R. and BODDIE, S. (2000): A transaction log analysis of a digital library. *International Journal on Digital Libraries*, v 3, no 2, p 152-169.

ELLIS, D. (1989): A behavioural model for information retrieval system design. *Journal of Information Science*, 15:237-247.

FURNAS, G., LANDAUER, I., GOMEZ, L. and DUMAIS, S. (1987): The Vocabulary Problem in Human-System Communication. *Communication of the ACM*, 30 (11):964-971.

GAINES, B.(ED) (1997): Using Explicit Ontologies in Knowledge-based System Development. *A special issue, International Journal of Human-Computer Studies*, vol 46, no 2/3.

HEMMJE, M., KUNKEL, C. and WILLETT, A. (1994):. LyberWorld – a visualization user interface supporting fulltext retrieval. *In Proc. Of the 17th Annual International ACM SIGIR Conference*, pages 249-259, Dublin, Ireland.

LENAT, D. and GUHA, R. (1990): Building Large Knowledge Bases, Addison-Wesley.

MORSE, E., and LEWIS, M. (1997): Why information visualizations sometimes fail. *Proceedings of IEEE International Conference on Systems Man and Cybernetics*, Orlando, FL.

SCAIFE, M. and ROGERS, Y. (1996): External Cognition: how do Graphical Representations work?. *International Journal of Human-Computer Studies*, 45(2):185--213.

SPOERRI, A. (1993): InfoCrystal: A visual tool for information retrieval & management. *In Proc. of Information Knowledge and Management '93*, pages 11-20, Washington, DC.

TREC, Text Retrieval Conference Home page, www.trec.nist.gov/.

VAN BERENDONCK, C. (2002): Bubble World, a novel visual information retrieval technique. Thesis. Air Force Institute of Technology, Dayton.