5. User Interaction

Users interact with IR engines in many ways. They formulate queries or routing requests. They review the results (if any) returned by the engines. They refine their original requests. They generate “profiles” reflecting their interests and preferences. They build training sets, and train IR engines to classify documents. They set parameters to guide the engines, e.g., filtering thresholds, cluster sizes or numbers.

Much of this material is covered elsewhere in this report, e.g., relevance feedback for query refinement, high-speed clustering methods for interactive clustering, the variety of query capabilities provided by Web IR and research engines, etc. But much of this discussion is conducted from the perspective of the IR engine, and its developer. Here, we will consider user interaction from the point of view of the user, and the researchers who are trying to make the user’s interactions more convenient and effective.

5.1 Displaying and Searching Retrieved Document Sets

Most IR engines return retrieved data in the form of a list of documents, ranked according to similarity to the topic or query for which they were retrieved, or the probability of relevance to the given topic/query. To make it easier for the user to scan this list, it is normally presented as a list of document surrogates, i.e., each document is represented by its title, or a short summary, each perhaps associated with its computed similarity or probability of relevance.

However, the number of retrieved documents may be very large, especially when the document collection is the huge number of pages comprising the Web, and all the databases to which Web pages act as gateways. Moreover, in many cases, the precision is low, due to the limitations of the existing technology on which IR engines are based, and the inexperience of human users. Hence, the relevant documents retrieved (if any) may be far down on the list returned to the user. Some systems may give the user the ability to limit the number of documents returned, either by setting a similarity/probability threshold, or by specifying the maximum number of documents to be returned. However, limiting the number of documents returned won’t improve the precision; if the precision is low, an arbitrary cutoff point may simply prevent the system from returning the relevant documents the user wants to see.
Another problem with a simple ranked list is that it gives few clues to which documents are closely related. These relationships depend, in general, on many attributes, e.g., many document terms, as well as external attributes such as author, date of publication, etc. In other words, a ranked list only represents one dimension. The user would like to see documents positioned according to many dimensions. (Of course, this applies equally to the original collection against which the query was executed.)

An alternative is to organize the retrieved document set so that the user gets the “big picture” quickly and visually, and can zero in rapidly on the desired documents, regardless of how far down the ranked list they are. The big picture also enables the user to see which documents are closely related.

Veerasamy et al. [SIGIR ’96] [SIGIR ’97] display the filtering set as a matrix. The rows correspond to key words from the query. The columns correspond to retrieved documents, ordered by rank, i.e., the leftmost column corresponds to the highest ranking document, the 2nd column corresponds to the document in rank 2, etc. The elements of the matrix are small vertical bars; the height of the bar for query word (row) \( i \), and retrieved document (rank) \( j \), is the weight of word \( i \) in document \( j \). The effect is that “one gets an immediate idea of how the different query words influence the document ranking.” One can see immediately which query terms are well-represented in high-ranking documents, and which are not. This may lead the user to modify the query by adding new words, dropping ineffective words, re-weighting query terms, etc. If two terms are closely related to each other and to the intended topic in the user’s mind, but exhibit low and dissimilar distributions in the filtering set, this becomes immediately obvious to the user; she may be able to improve filtering and ranking, by modifying the query to specify the words as a phrase, or specifying that they must satisfy a proximity condition. In this way, documents in which the key words co-occur in close proximity are favored, and receive higher ranking.

In their SIGIR ‘97 paper, Veerasamy et al. describe a carefully controlled experiment to measure the effectiveness of their filtering set visualization technique. The nature of the experiment, and the measures defined for interactive measurement are as interesting and significant as the results themselves. And the experimental results tell us as much about
the human task of making relevance judgments as about the value of the visualization tool. They used a portion of the TREC data, and ten TREC information topics (queries). Veerasamy et al. used the INQUERY 2.1p3 engine as the common IR engine. They controlled for precision, on the assumption that the task of recognizing a relevant document is significantly different (and harder!) than the task of recognizing a non-relevant document. Hence, each user was given, for each topic, a high-precision and a low precision document set. (The high precision set was the first 60 documents retrieved by INQUERY. The low precision set consisted of the documents ranked 90 to 150 by INQUERY.) To control for the effect of the visualization display, the high precision set was further divided into an even-ranked set (ranks, 2, 4, etc.) of 30 documents, presented to the user with the visualization tool available, and an odd ranked set (ranks 1, 3, etc.) of documents, presented to the user without the visualization tool. A similar division was made in the low precision set. For a given topic, each user was tasked to judge the relevance of the documents in each of four sets: high-precision even rank, high-precision odd rank, low-precision even rank, and low-precision odd rank. They were told that they were testing the effectiveness of the visualization tool. They were not told that all four sets came from the same collection, and not told that some were high precision, some low precision. Finally, they were given a monetary incentive to judge relevant accurately, and to judge quickly; the users who ranked best in a score that measured both accuracy and speed in completing a task received a sum of money.

Veerasamy et al. defined several measures of human interactive effectiveness: Interactive precision is defined as the proportion of documents judged relevant by the user that were also judged relevant by the TREC judges. Interactive recall is defined as the ratio of documents judged relevant by the user to documents judged relevant by the TREC judges. Accuracy is defined as the number of correct relevance judgments minus the number of incorrect judgments. Correctness means agreement with the judgment of the TREC judges, both with regard to relevance and non-relevance. Hence, in all cases the judgment of the TREC judges was treated as absolute “truth.” So, the difference between interactive precision and recall, and their more traditional counterparts is that the interactive versions measure a user’s relevance judgments rather than an IR system’s judgments.
The Veerasamy experiment showed that users can “identify document relevance more accurately with the visualization tool than without.” The effect of the visualization tool on accuracy is about the same for high precision and low precision document sets. The visualization tool also improves the time required to judge relevance (about 20% improvement), but this effect is much more pronounced for low precision sets than for high precision sets. Finally, the experiment showed that the visualization tool produced a significant improvement in interactive recall (and in the speed of identifying relevant documents as well), but only a minimal improvement in interactive precision. However, users achieve a much higher absolute accuracy for low precision document sets than for high precision document sets independently of whether they use the visualization tool, showing that their ability “to identify non-relevant documents as non-relevant is much higher than their ability to identify relevant documents as relevant.” In other words, non-relevant documents are easy to recognize, while it takes extra effort to identify a document as relevant. This effect is much stronger than the influence of the visualization tool.

Note by the way, the interplay of interactive recall, interactive precision, and accuracy. The improvement in interactive recall means that the visualization tool is helping users to correctly recognize a higher proportion of the actual relevant documents. The corresponding minimal improvement in interactive precision means that the improvement increase in relevant documents identified is counterbalanced by a proportionate increase in non-relevant documents falsely identified as relevant. Hence, the concurrent improvement in accuracy must mean that the visualization tool substantially helps the users in correctly classifying documents as non-relevant, classifying more non-relevant documents correctly as non-relevant, and fewer relevant documents incorrectly as non-relevant.

Hearst’s TileBar display paradigm [ACM SIGCHI, 1995] may be compared and contrasted with the Veerasamy approach. In Hearst’s display, each row corresponds to a retrieved document. Each document is represented as a series of adjacent non-overlapping segments called tiles. The order in which tiles are displayed in a row is the order in which they occur in a document. As explained in the section on query-document similarity, tiles are multi-paragraph segments such that each tile is about some sub-topic of the document, and the boundary between successive tiles represents a change in topic,
as measured by the $tf*idf$ similarity measure. In the tilebar display, each tile is represented as a small rectangle. Its shading on a grey scale from white to black represents the sum of the frequencies of all the query terms, white representing the complete absence of the query terms, and black representing a heavy concentration of the query terms. Hence, the user can see at a glance whether a given document is largely about the given query (much black throughout the row), whether it has passages relevant to the given topic (isolated black sections separated by much white), whether it has passages that may be about the given topic (grey sections), and so on. The user can see not only how much of the document is relevant, but also where the relevant passages are, e.g., at the beginning of the document, in the middle, etc. Similarly, the user can see at a glance which of the set of documents displayed are most likely to contain relevant passages, or to be largely about the given query.

As a further refinement, the user can see the document set displayed relative to several sets of query terms. For example (the example is Hearst's), the user may be interested in “computer-aided medical diagnosis.” She may supply three sets of query terms, one set relating to medicine and patients, a second related to tests and diagnoses, and a third related to computer software. The TileBars display for a given document will show three rectangles for each tile, arranged vertically one above another. The degrees of shading of the rectangles for a given tile immediately tell us how much the tile is about each of the sub-queries. If all three rectangles for a given tile are black or dark grey, there is a good chance that the corresponding passage is about all three of the specified sub-topics. On the other hand, if the dark rectangles for one sub-topic are in completely different tiles from the dark rectangles for another sub-topic, then the document is less likely to be relevant to the user’s topic, although it might score high on a conventional document similarity ranking. For example, the document might discuss both software and medical diagnosis, but the references to software might have nothing to do with its application to medical diagnosis.

Note that the Hearst display, unlike the Veerasamy display, is not a term-by-document matrix, or even a tile-by-document matrix. Indeed, a tile “column” would be meaningless, since each document is composed of its own unique set of tiles. But documents, i.e. rows, can be compared with respect to the distribution and shading of their respective tiles. Moreover, the display of each document in Hearst’s display
represents not merely term occurrence as with Veerasamy, but local term co-
occurance within the document’s tiles.

Both Veerasamy and Hearst give the user a visual display of each of a set of
individual documents. The user can study the properties of an individual document, or
compare documents within a set. By contrast, another way to give the user an overview
of the filtering set is to cluster the documents. Instead of seeing a (perhaps very long)
list high-ranking documents, the user sees a modest number of document sets, each set
clustered by some measure of content similarity that one hopes corresponds to topic
similarity. Each group is identified by key-words, phrases, or other labels, that (again, one
hopes) tell the user what topic(s) each cluster is about.

Clustering a very large filtering set retrieved from an even larger set such as the Web
imposes certain requirements. First, pre-processing (which can serve to speed up
clusteringsee the section on clustering above) is impossible. The original collection,
e.g., the internet, is far too large (and dynamic) to pre-cluster. The filtering set itself also
cannot be pre-processed because its content is not known until the IR engine executes a
user query. Second, the clustering must be fast; specifically, it must not add substantially
to the time required for filtering by the IR engine, which for Web retrievals is typically
a matter of seconds. If clustering adds minutes or hours, the additional time would
usually far outweigh the benefit of clustering. Third, the cluster labels should enable the
user to pick out the best cluster(s) very rapidly. Finally, selecting the best cluster(s)
should substantially improve the precision of the user’s search, i.e., the effective
precision if the user examines the documents in the best cluster(s) first, should be much
better than if the user merely searched down the ranked list returned by the IR engine.

STC clustering, described above in the section on clustering, has been developed with
just those requirements in mind. It is linear-time, not as good as the constant-time and
almost-constant-time methods described in the section on clustering, but probably as
good as can be achieved without pre-processing. Moreover, it is incremental, which
means that clustering can proceed while the data is being retrieved and documents are
being returned to the user. By the time the last document in the filtering set arrives,
the clustering can therefore be nearly done. (This assumes of course, that the
documents can be clustered as fast as they arrive at the user’s site, which is in fact the
case, in the Web filtering test reported by the STC developers, Zamir et al. [SIGIR ‘98]
No actual test with real users was conducted in the reported research, so it remains to be determined how effective the cluster labels (strings of consecutive words shared by the documents in a cluster) prove to be identifying cluster topics and topic relevance to a real user. However, intuitively, strings of words should prove more informative than individual key words, and could always be supplemented with titles (where applicable) and statistically derived terms. In any case, the STC study made use of the experience of other researchers, who did provide document clusters to real human users. These experiences indicated that a user could select the “best” cluster first about 80% of the time. Hence, the STC researchers calculated precision on the assumption that the user was able to rank the topic clusters by number of relevant documents. On this assumption, they compared STC with several other linear time clustering methods, and one classic $O(N^2)$ method. STC was the clear winner. However, it should be stressed that this was a comparison of cluster quality, i.e., the best STC clusters contained more relevant documents than the best clusters produced by the other methods, not a comparison of the user ability to select the best clusters. It should also be noted that the queries employed by the researchers were generated by the researchers themselves, the queries were executed via real Web engines against the actual Web, and relevance judgments were assigned to the retrieved data by the researchers. Thus, the queries and test data were not a very large standardized test set such as the TREC data so widely employed in IR research.

### 5.2 Browsing a Document Collection

The term “browsing” implies that the user starts searching the data without a clear-cut end goal in mind, without clear-cut knowledge of what data is available, and very likely, without clear-cut knowledge of how the data is organized. She may have a rough goal in mind or perhaps no goal at all, or many possible goals. If she has a rough goal at all, it isn’t clearly defined enough to be formulated as a query. She searches the data as fancy takes her, formulating and modifying goals, as she encounters data or categories of interest.

The browsing method depends, in the first place, on whether or not the collection to be browsed has been manually indexed, i.e., whether or not human indexers have assigned subject categories to each document. A very popular example of manual indexing is the Web service, Yahoo (discussed below in the section on Web IR engines). However, much
IR research has been devoted to accessing collections that have not been manually indexed. Let us consider first some browsing techniques that can be applied to collections that are only indexed by IR engines.

The information space browsing paradigm allows the user to visualize a vector space (such as the spaces discussed in an earlier section), and move around freely in that space (or what comes to the same thing, to manipulate the space itself). The human user is accustomed to moving around in a three-dimensional space in the real world. She is also accustomed to moving a cursor around in the two-dimensional space of computer monitor screen, using a device such as a mouse. However, to apply the “movement in space” metaphor to IR browsing, several problems must be surmounted.

The most obvious difficulty is that the number of dimensions in a typical IR vector space is much greater than two or three. Even with dimension reduction techniques such as LSI, the number of dimensions may be 50 to 200, far more than a human can readily visualize. Hence, a number of key dimensions, e.g., especially important query or document terms must be selected. If more than three dimensions are selected, the additional dimensions must be mapped into visual characteristics other than spatial coordinates. Each document is located in space by its spatial coordinates, and represented by a visual object, often called a glyph or icon. The additional dimensions can be represented by such visual characteristics as color, shape, texture, degree of opacity, etc. [Ebert, CIKM'95] Note that while each of these characteristics can vary according to a linear scale, opacity can effectively act as a filter. That is, on an opacity scale, a glyph varies from opaque to transparent. But a transparent (or near transparent) object disappears from the screen; hence, it will be effectively filtered out.

Viewing a glyph-based information space, and browsing in such a space, is significantly enhanced by the Stereoscopic Field Analyzer (SFA) [Ebert et al., IEEE Graph, 1997]. It is trivial to represent two spatial dimensions on a two-dimensional computer monitor screen. Three dimensions can be represented by the use of perspective, and manipulated, e.g., rotated and translated, via mouse control. SFA improves on these techniques in three ways. First, it provides a true 3-D stereo effect, by rendering the information space twice, once for each eye, and viewing the space through Liquid Crystal Shutter Glasses. Second, the user is given a tracking control, equipped with buttons. By moving this control with her hand in actual physical space as she sits in front
of the monitor, and pressing the buttons to grip and release the information space, the user can manipulate the entire 3-D space, both rotating it in 3 dimensions so that the space may be effectively viewed from any direction, and translating it, i.e., moving the entire space up, down, left, right, away from the user, and toward the user (the latter two movements corresponding to zooming out and zooming in). Third, the user is given another hand control to be manipulated with her other hand. This control can be used for finer manipulation, such as sweeping out a section of the space for closer examination, or pointing at a particular glyph (which may represent a single document or a cluster of documents).

One inherent limitation of the SFA/Information Space approach is that only three dimensions can be manipulated and browsed directly with the two manual controls. This is not merely a limitation of SFA. It is also a limitation of the human perceptual capability. We live and perceive in a three-dimensional world. However, SFA provides flexibility by permitting the user to specify which of the document attributes are to be mapped into each of the three spatial dimensions. The user can also specify what range of attribute values is to be mapped into the corresponding axis of the information space display. This capability can be used for such purposes as filtering out uninteresting ranges, e.g., a large cluster of documents near the origin of coordinates. Other attributes can, of course, be mapped into other visual cues as noted above: color, shape, texture, etc. These dimensions can not be manipulated with the 3-D tracker, but could be controlled separately, e.g., by graphical sliders.

A completely different approach to browsing, the “scatter/gather” method [Cutting et al., SIGIR’92], is based on a different metaphor, that of alternating between consulting the table of contents of a book (to get an overview of what is available), and consulting its index (to find the page or section dealing with a specific, narrow topic). The “table of contents” is generated (conceptually) by clustering the document collection. The labels or summaries that identify each cluster form the table of contents. The hope is that the documents that cluster together will be about a common topic, and that the label will identify that topic (or topics) to the user. This is called the “scatter” phase, because the documents, initially comprising a single collection, are scattered into multiple clusters. Then, the user scans the cluster labels (the “table of contents”) and selects the cluster(s) that interest her most. This selection process is the “gather” phase, because the user is...
gathering the selected clusters together into one document collection, a subset of the original collection. Next, the system clusters (scatters) again, but this time the clustering is applied to the subset collection. Hence, the clustering will be finer-grain, identifying sub-topics within (and perhaps across) the topics selected by the user. Hence, a new finer-grain table of contents is produced. Once again, the user selects (gathers) clusters (topics) of particular interest. The user repeats this scatter/gather process until she has narrowed her focus down to one or more specific topics for which she wants to read or scan the actual documents. Or perhaps, summaries, or abstracts, or the cluster labels themselves, at a fine level of detail, are sufficient to tell her what she wants to know. At any stage in this scatter/gather sequence, the user can employ an alternative focused search strategy, e.g., a key-word or boolean query to select particular documents from a cluster representing a topic of interest to the user. This corresponds to looking up a specific term or narrow topic in an index, the second part of the metaphor of alternating table of contents overview and index lookup.

At any level of detail, the user can back up to a higher level, and select different topics to pursue, initiating a new gather/scatter/gather sequence.

Various techniques can be used to generate labels for a cluster. Cutting et al. use a cluster digest. The digest of a given cluster $C$ consists of the $m$ most central documents in $C$, and the $w$ most central words in $C$. The most central documents are those most similar to the cluster centroid (which they call the cluster profile). The highest weighted terms can be selected either from the cluster profile, or from the profile of its most central documents. The centroid (or profile) of the cluster is the normalized sum of the term vectors describing the documents of which the cluster is composed. Cosine similarity is used to compute the similarity of a document in the cluster to its profile. Term weight for a given term in a given document is computed as the square root of the term frequency.

Since scatter/gather requires “on the fly” re-clustering (scattering) of clusters selected (gathered) interactively, a rapid clustering method and algorithm is essential. Cutting et al. use buckshot and fractionation, two linear time clustering methods described above, in the section on heuristic clustering. These algorithms are used to find cluster centers rapidly. Each document is assigned to the closest center. Then, various refinement techniques are applied, e.g., once each document has been assigned to a center, the
centers can be re-computed, and then each document can once again be assigned to the closest center. This process can be repeated indefinitely. Other refinements include splitting clusters that fail some simple coherency criterion, and joining clusters that have a sufficient number of topic (highly weighted) words in common. Finally, since even a linear time clustering method can be too slow for interactive clustering if the collection to be clustered is large, they use computationally expensive pre-processing, specifically the computation in advance of a cluster hierarchy before runtime, to achieve constant-time clustering [Cutting et al., SIGIR’93] during an interactive scatter/gather session. This constant-time method is discussed in the section on heuristic clustering above.

5.3 Interactive Directed Searching of a Collection

In contrast to browsing, “directed” searching means that the user has a specific information need. (This is the usual assumption of both ad hoc querying and routing.) “Interactive” directed searching means, of course, that instead of merely formulating and kicking off a single query and examining the results returned by the IR engine, the user engages in an interactive process, either to formulate the original query, or to refine the query on the basis of the initial results returned.

Relevance feedback, discussed above in the sections on Query Expansion and Query Refinement, is the classic method of improving a query interactively. Here, a variation of relevance feedback, and the use of clustering for query refinement are discussed.

Aalbersberg [SIGIR ‘92] proposes a simplified form of interactive relevance feedback that he calls “incremental relevance feedback.” Most IR systems that support relevance feedback perform the reformulation of the query automatically, concealing the mechanics and often the reformulated query from the user. However, in conventional systems, each time the user executes the (original or reformulated) query, she sees a set of N retrieved documents, typically 10 to 20, from which she must select those she judges relevant. After she has judged the N documents for relevance, she requests an automatic reformulation of the query, and execution of the reformulated query. In Aalbersberg’s system, the user is not aware of her query being reformulated at all. The user sees one document at a time. She designates that document as relevant or not. If it is relevant, its title is added to a “Results” window of relevant documents. She then sees another document, and again judges its relevance, and so on. After the user has viewed
and judged $N$ documents, the titles of that fraction $N_r$ that have been judged relevant are in the *Results* window. Hence, she has the sense that she is merely judging a series of documents that the system believes may be relevant to her original query (or the information need that query is intended to represent). However in actual fact, the first document she sees is the highest ranking document in the list returned after executing the original query. Thereafter, each time the she judges the relevance of the current “best” document, this judgment is immediately used to reformulate the current query. This reformulated query is immediately executed, invisibly to the user. The next document submitted to the user for relevance judgment is the highest ranking document returned for the reformulated query.

Aalbersberg uses the Rocchio formula for query reformulation. However, this formula takes on a very simple form in Aalbersberg’s system, since each stage involves modifying the query vector by either adding a single document vector (if the current document is judged relevant), or subtracting a single document vector (if the current document is judged non-relevant). In either case, the document vector represents the single document the user has just judged, multiplied by the appropriate constant ($B$ or $C$) from the Rocchio formula.

Above, the use of clustering for browsing a document collection, the so-called *scatter/gather* method, was discussed. Earlier, in the section on clustering, the use of hierarchical clustering for a directed search was discussed. By drilling down through the hierarchy, the user can focus on the small number of documents, in a cluster at the lowest level of the hierarchy, about the topic that concerns him. Roussinov *et al.* [SIGIR ‘99] suggest another interactive use of clustering: to help the user refine and reformulate his query.

The scheme is to submit a simple natural language query to a Web IR engine (Roussinov *et al.* use Alta Vista). Their system fetches the 200 highest ranking documents from the list returned by the IR engine. These documents are automatically clustered, using an unsupervised clustering technique. (Roussinov *et al.* use the self-organizing map technique. Another, possibly better technique would be the STC clustering method discussed earlier in the section on Incremental Clustering. The essential characteristics of the clustering method is that it must be unsupervised, and that it must generate labels for each cluster that can aid the user in rapidly identifying the content of a given cluster.
Speed is another important characteristic of a clustering method for on-line clustering of retrieved results. However, Roussinov et al. are only clustering the top 200 documents, so they don’t need as fast a method as Zamir et al. who use STC to cluster a much larger filtering set.) The system then displays for the user the cluster labels and “representative terms associated with each cluster.” The user selects from this display those labels and terms that seem relevant to his original query (or to the current information need the query was intended to express). The selected terms and labels may also suggest additional terms that belong in the query. He types these additional words or phrases. The system then uses the selected and typed terms and labels to create a set of new or reformulated queries, which it then submits to the IR engine. Multiple iterations of this process are supported.