Facilitating Network Data Exploration with Query Previews: A Study of User Performance and Preference

Egemen Tanin[14], Amnon Lotem[1], Ihab Haddadin[2],
Ben Shneiderman[14], Catherine Plaisant[4], and Laura Slaughter[3]

egemen@cs.umd.edu, lotem@cs.umd.edu, u-ihaddadin@bss2.umd.edu, ben@cs.umd.edu,
plaisant@cs.umd.edu, lauras@oriole.umd.edu

[1]Department of Computer Science, [2]Department of Systems Engineering,
[3]College of Library and Information Services, [4]Human-Computer Interaction Laboratory, and Institute for Advanced Computer Studies

University of Maryland
College Park, Maryland 20742, USA

2 / 19 / 1998

Abstract
Current network data exploration systems which use command languages (e.g. SQL) or *form fill-in* interfaces fail to give users an indication of the distribution of data items. This leads many users to waste time posing queries which have zero-hit or mega-hit result sets. *Query previewing* is a novel visual approach for browsing huge networked information warehouses. Query previews supply data distribution information about the database that is being searched and give continuous feedback about the size of the result set for the query as it is being formed. Our within-subjects empirical comparison studied 12 subjects using a form fill-in interface with and without query previews. We found statistically significant differences showing that query previews sped up performance 1.6 to 2.1 times and led to higher subjective satisfaction.

Keywords
User Interface, Direct Manipulation, Dynamic Query, Query Preview, Form Fill-in.

1. Introduction

Retrieving information from huge data warehouses in computerized environments has always been an important issue in computer science. Introduction of networked information systems and frequent changes in the amount, type, and format of data make the problems even more challenging.

A common example of information-seeking tools is search engines on the World Wide Web. Users enter keywords to a user interface that has text entry fields [10]. This form fill-in approach requires an explicit submission of a form to a search engine with three possible results:

- a small set of possibly related documents that users are looking for,
- a huge set (mega hit) of related or most probably unrelated documents that is burdensome to browse, or
- zero hits.

Users always wish for the first result in any query that they submit. Nobody likes to browse a huge set of documents only a few of which might be related to their needs. Even worse, the case of zero hits lead users to a feeling that they have done something wrong (without indicating whether a spelling mistake or lack of data is causing the problem). Often, network resources, time of the users, and processing power of the search engine are wasted.

A common problem in this approach is that the interface that is supposed to guide users to a reasonable result confuses them. It takes control away and does not give guidance that leads to a successful result.

Recent research showed that improved methods exist for more efficient querying. For example, the Butterfly System [7] and the Harvest System [2] show that efficient querying is possible on the World Wide Web. These systems also attack the problem from the information processing and system utilization sides. Veersamy and Navathe [11] address the problem of the relevance of the results of a query to the keyword set and propose a user interface solution.

Dynamic queries [1,9,12] use a direct manipulation approach to facilitate query formulation with a visual representation of query components and results. They enable a rapid, incremental, and reversible control of the query. They also give continuous feedback to users for guidance in query formulation. Figure 1 shows an example dynamic query interface.

The application of dynamic querying to networked querying environments might be useful. On the other hand, high system-resource demands make dynamic querying less applicable to huge networked information warehouses. One solution to this problem uses data aggregation in tandem with dynamic queries [5]. Another solution to this problem might be the division of the bigger problem into several smaller problems, as in *query previews* and *overviews* [3,4,8]. The paradigm is to give an overview of the database to users before the details are visualized.

A good overview should enable users to see the necessary detail in order to make the final query and understand the distribution of data in the database. The querying process is divided into steps to reduce the resources needed to form the final query. So a multi-phase incremental querying process will hopefully lead to desired results using less resources and time.

Dynamic query interfaces have been implemented for NASA's Global Change Master Directory and the web-site is in the process of being made public.

1.1. Two-Phase Querying

In the two-phase approach the designer chooses a few of the most discriminating attributes of the database to design a direct manipulation user interface. The rest of the attributes should be kept for a second phase that will also include these discriminating attributes.

When the querying environment is activated the first interface (query preview) appears immediately. Users make some decisions on this first interface and then move to the second one (query refinement) to complete the query.

1.1.1. Query Preview

The query preview is a powerful tool to define rough ranges on the data set that is being explored. It shows the discriminating attributes in the database so that any selection would lead to a smaller subset of the database. It consumes modest system resources because only a small number of attributes from the overall database are used at this phase.

To guide users in the query formulation process the preview should be supplied with aggregate information about the database (e.g., possible number of hits to the query to be formed). Distribution of data over attribute values can also be shown as a pie or bar chart.

When users select a value on any of the attributes of the preview panel the rest of the user interface (e.g., bars) should be updated in well under one second. This is called tight coupling. Therefore, for each action users take, feedback is given.

As users see the potential size of their query result before refining the rough ranges, there is less chance that they will get zero or mega hits. The system load should drop drastically because users don't waste their time with zero hit queries or consume network resources in downloading useless results.

Perhaps the greatest advantage of the query preview for users is that they only need to download the aggregate information about the data distribution at this phase. So whatever the database size, only the distribution information of the data is needed to form a preview, thereby decreasing the system resource demands. An example query preview is given in Figure 2. Three major attributes of the database were chosen for this query preview. The distribution of data over these attributes is shown with bars and the possible result set size is displayed as a bar at the bottom part of the interface.

1.1.2. Query Refinement

The query refinement phase works after the query preview and hence inherits the constraints and the data set sent from the first phase. This phase can easily be implemented as a dynamic query interface to browse the remaining data. When a desired result set is obtained, it can be saved.

If the data is not a collection of documents or values but a set of pictures the refinement phase becomes another phase in querying. Using a similar approach a third phase can be added to the first two phases (in picture databases this can be a tool to analyze a picture in more detail). Figure 3 shows a sample query refinement implemented using the dynamic querying paradigms. All of the attributes of this sample data set are displayed on this dynamic query interface. The initial query, specified on the query preview, can be refined at this level by just selecting the desired values of the attributes of the database.

1.2. User Study

Since query preview interfaces add another phase to query formulation, there is the possibility that user performance would deteriorate and that they would be annoyed by a two-phase approach. Our study analyzed the effects of the query preview on user performance and preference. We believe that query previewing guides users in forming a query. It helps narrow down the huge search space to a manageable size by giving information about the distribution of the data. For better control, our study was done in a non-networked environment but advantage of the query previews should be greater in a networked environment due to the additional delays during the downloading process.

2. Experiment

2.1. Introduction

This experiment examines the benefits of query previews when the access to the data sources is immediate, that is, all data is on a local server. Our claim is that the advantage is greatest when the users' search tasks require repetitive submissions, and a query preview can give task-related insights to the database. *Unclearly specified tasks* usually require several submissions. In such tasks the user's constraints and preferences cannot be specified in a simple way. For example, in searching for a film to view from a film library, the users' preferences might be "an award winning comedy from the last two years", but if none is available "the most popular science fiction" should be retrieved. *Clearly specified tasks* have a straight forward and an accurate definition (known-item searches), e.g. "Find the earliest film of John Wayne". Practically, the query preview cannot supply a database overview based on all the data attributes. The subset of attributes that are used in the query preview might be *relevant* or *irrelevant* to the user's tasks.

The three task types in the experiment varied in the clarity of their specifications and in the degree of relevance of the specified attributes to the query preview. Twelve subjects performed a set of tasks, once by using a form fill-in interface and once by using an interface that included a query preview and form fill-in. The time for completion of the tasks and the subjective preferences of the subjects were measured.

2.2. Hypothesis

Our hypotheses were: (1) For unclearly specified tasks, the form fill-in interface with a query preview yields faster performance than a form fill-in interface without a query preview. (2) For clearly specified tasks, the form fill-in interface without a query preview yields faster performance. (3) Users prefer query preview interfaces regardless of their performance.

The independent variable was the interface type and the treatments were:

- Form fill-in interface with a query preview
- Form fill-in interface without a query preview

We examined the two interfaces using three different types of tasks:

- Clearly specified tasks in which the query preview attributes are not relevant to the task.
- Unclearly specified tasks in which some of the query preview attributes are relevant to the task.
- Unclearly specified tasks in which all of the query preview attributes are relevant to the task.

The dependent variables were the *time to complete* the tasks in each interface (not including setup times) and the *subjective preferences* of the users.

2.3. Subjects

Twelve computer science graduate students were used as subjects. All of them use a computer almost every day and have at least five years experience in using computers. All, except one, regularly or frequently use Internet or database searching tools.

2.4. Materials

The materials include a form fill-in interface for querying a film database (including 500 films), a query preview panel for the same database, a set of tasks to be performed by the subjects, a subject background survey, and a subject preference questionnaire.

2.4.1. Form Fill-in Interface

The form fill-in interface (Figure 4) is used to perform queries on a film database. There are ten attributes for a film: category (horror, action, comedy, etc.), award winner (yes or no), rating (R, PG-13, PG, and G), year of production, length, popularity, lead actress, director, lead actor, and title. The output of a query is the list of films matching the specifications of the query. Vertical and horizontal scroll-bars can be used for scanning the list. The form fill-in interface was developed in C using the Motif library.

2.4.2. Query Preview

In the query preview (Figure 5) users select values for three attributes of the database: the category (horror, action, comedy, etc.), whether the film won an award or not, and the rating (R, PG-13, PG, and G). Multiple selections are available for each of these attributes (using check boxes). The number of films for each attribute value is shown on a separate *preview bar*. Each preview bar consists of a frame and an internal rectangle (gauge). The length of the frame is proportional to the number of films in the database that match the value of that attribute. The length of the gauge is proportional to the portion of these films which match the values specified for the other attributes as well (the number of matches appears to the left of the bar). Users formulate queries by selecting the attribute values. As each value is selected, the preview bars in the other attribute groups adjust to reflect the number of films available (this is called *tight coupling*). For example, users might be interested only in films that won awards. By selecting "Award Winners", the gauges of the preview bars of the selected categories and ratings change immediately to reflect only films with awards. The query preview bar at the bottom of the screen changes its length to illustrate the total number of films that match the current conditions.

When the "Refine" button is pressed, the query preview submits the specified partial query to the search engine and all the films that satisfy the query are downloaded for the query refinement phase. The query preview is closed and the form fill-in interface is loaded to refine the query (displaying initially all the films selected in the query preview). The query preview was also developed in C using the Motif library.

2.4.3. Tasks

The tasks given to subjects were to find a film or a list of films in the database satisfying constraints that we provided. Three types of tasks were used:

T1: a clearly specified task in which none of the query preview attributes is relevant for the task, e.g. "Find the latest film by Alfred Hitchcock" (known-item search). For that type of task, users can typically find the answer by submitting a single form fill-in query. The query preview has no advantages since its attributes are not relevant to the query;

T2: desired films are vaguely specified. In this type of task, some of the query preview attributes are relevant, e.g. "Find a PG-13 musical which was produced between 1991 and 1995, if no such film is available, find a war film from the same years with the same rating, if not, try a musical or a war film from 1970-91, and as the last possibility, try a comedy from 1970-95". This type of task is typical when users have a complex set of acceptable results, with clear preferences. To perform such a search in the form fill-in interface users must issue several queries, when the preferred choice is not available in the database. In the query preview, users can get some insight about what is available in the database and what is not. However, since not all the attributes in the specification appear in the query preview, the form fill-in is required for refining the query; and

T3: formed in a similar way to T2. A series of preferences for films are specified. In this case however, the query preview attributes are fully relevant to the task specifications. Example: "Find at least 30 films of the same category which are R rated and have no awards" (for example, in order to organize a film festival). In the form fill-in interface this task requires several queries to examine the number of films in each category. The query preview on the other hand, gives an immediate picture of the relevant categories. The form fill-in is required only to get an explicit list of the films.

For each of the above task types, six were prepared (18 tasks in total, see appendix A).

2.4.4. Subject Background Survey

The survey included 8 questions which ascertained the experience level of the subjects with computers in general and with search engines in particular.

2.4.5. Subjective Preference Questionnaire

The subjective preference questionnaire included 8 questions aimed at finding out which of the two interfaces (a form fill-in with or without a query preview) the subjects preferred and what their attitudes are toward adding query previews.

2.5. The Experiment Design

The experiment used a within-subject counter-balanced design with 12 subjects. Each subject was tested on both of the interfaces, but the order of the interfaces was reversed for half of the users. A parallel set of tasks was used on the second interface to reduce the chance of performance improvement. Each set of tasks included the three types of tasks (T1, T2, T3), with three tasks for each of these types. The order of the task types within a task set was also reversed (each of the six permutations was experienced by two subjects). The order of the tasks within each task type was fixed.

2.6. Procedure

The subjects signed a consent form, filled out a background survey, received a brief demo of the form fill-in interface and the query preview, and a 10 minute training session in which they used the two interfaces (similar tasks to the actual tasks were used). During the experiment each subject performed 18 tasks (9 in each of the interfaces). At the end of the experiment the subjects filled in the preference questionnaire. The experiment took 50-60 minutes including the training and the questionnaires.

2.7. Administration

Two experimenters were present. One administered the experiment, performed the demo, presented the tasks, and measured the task execution times. The other experimenter was a viewer who recorded notes about the way subjects coped with the tasks and about problems during the experiment. The time that the subjects spent in using each of the interfaces was recorded (successful completion time of a task). These times did not include setup times of the programs.

3. Results

3.1. Time for Completing the Tasks

Table 1 summarizes the times for completing each of the task types for novices (*clearly specified*: *T1*, *unclearly specified and partially relevant*: *T2*, *unclearly specified and fully relevant*: *T3*) for each of the interfaces (*with* and *without preview*). Figure 6 presents these results as a bar chart.

For T1 tasks the interface with the query preview yielded slower performance than the interface without a query preview ($t(35) = 2.44$, $p < 0.05$). For T2 and T3 tasks the interface with the query preview yielded faster performance than the interface without a query preview ($t(35) = 8.77$, $p < 0.05$, and $t(35) = 14.70$, $p < 0.05$, respectively). The statistical analysis used one-tailed paired two-sample t-test for means. Each task is considered separately leading to degrees of freedom of 35.

3.2. Expert Performance

Two expert users (the developers of the two interfaces) performed the same tasks (average completion times appear in Table 2). These results are meant to indicate the potential for performance after experience. No statistical analysis was performed.

3.3. Subjective Satisfaction

The subjects answered six questions about their preferences, quantifying their preferences on a 1 to 9 scale (with higher numbers indicating stronger preferences). The first question examined the general preference of subjects for using a form fill-in interface with or without a query preview (Table 3 and Figure 7).

The results showed a statistically significance difference ($t(11) = 2.82$, $p < 0.05$) for the interface with a query preview over the interface without a query preview. The rest of the questions examined more specifically what the subjects thought about the interfaces. The results (average scores, standard deviations, minimums, and maximums) appear in Table 4. Figure 8 presents these results in a histogram.

The scores for all of the questions were statistically significantly above the mid-point scale value of 5.0 ($t(11)$ = 3.86, 6.20, 7.71, 2.24, and 2.58 respectively, $p < 0.05$).

4. Discussion

Our findings support the hypothesis that for the unclearly specified tasks the interface with a query preview yields more rapid performance than the interface without a query preview. For both types of the unclearly specified tasks the improvement in performance was significant (at the level of 0.05): 1.6 times faster for *T2* tasks and 2.1 times faster for *T3* tasks. For the clearly specified tasks (*T1*), as expected, the form fill-in performed slightly better. The results with two expert users (the developers of the interface) showed similar outcomes (slightly slower for T1 tasks, 2 times faster for T2 tasks, and 2.4 times faster for T3 tasks).

4.1. Clearly Specified Tasks (T1)

As expected, users of the form fill-in interface for clearly specified tasks performed more rapidly since they were able to find the answer by submitting a single form fill-in query. The query preview had no advantage since its attributes were not relevant to the query. However, users of the interface with the query preview performed only slightly worse (10% slower). The users spent 2-3 seconds in the query preview, identified that its attributes are not relevant for the task and continued to the refinement phase.

4.2. Unclearly Specified Tasks, with Partial Relevance of the Query

Preview Attributes (T2)

Although not all the attributes in the task specification could be specified using the query preview, the insight gained from the query preview enabled users to eliminate some potential zero hit queries in advance, concentrating in the refinement phase on a much smaller set of possible queries. Apparently, the query preview enabled the users to reduce the search space significantly, and therefore find the answer more quickly.

4.3 Unclearly Specified Tasks, with Full Relevance of the Query

Preview Attributes (T3)

For unclearly specified tasks the full power of the query preview was used. The query preview enabled the users to see immediately which of the possible queries should be submitted. The users loaded the refinement phase only for submitting the query and viewing the results. The users performed the refinement phase with a high confidence that they would get the expected results. On the other hand, in the interface without a query preview, the users had no clue about which of the possible queries will give the expected result. They had to try several possible queries, submitting 5-6 queries on average until they got a satisfactory answer. Although the response time for each such query was immediate (1 second), the time for fill-ing in the specifications of each query (5-10 seconds) caused the significant differences in performance.

4.4. Performance Improvement

The following simple model for the performance time in the refinement stage can be used to explain the results:

$$performance\_time = no\_of\_queries \times query\_time$$

where:

$$query\_time = fill\text{-}in\_time + response\_time + analysis\_time$$

The *fill-in_time*, *response_time,* and *analysis_time* are the average times for fill-ing in a query, getting a response and analyzing the results, respectively. The response time is a function of several parameters such as the complexity of the query, the size of the database, the load on the database server, the number of the retrieved entities and the load on the network. The time for analyzing the results is determined by the number of retrieved elements. In our experiment the response time was short (1 second), the average analysis time was small (analysis of zero hits). Thus, the main factors were *no_of_queries* and *fill-in_time*. For the *T3* and *T2* tasks, the query preview achieved the performance improvement by reducing the *no_of_queries*, yielding a situation in which:

$$preview\_time + (no\_of\_queries_{refinement} \times query\_time) < no\_of\_queries_{form\_fill\text{-}in} \times query\_time$$

In the common situation where the access to the database would be through a network, the response time would be typically larger than one second and the performance improvement which is achieved for *T2* and *T3* tasks would be even greater.

The results show that for different types of tasks the query preview achieves different rates of performance improvement in comparison with the traditional form fill-in interface (from 0.1 times slower in *T1* to 2.1 times faster in *T3*). The performance improvement which follows from the reduction in the number of required queries depends on several parameters. One parameter is the *clarity* of the task specifications. In clearly specified tasks the number of queries required in a form fill-in interface is small, hence there is no potential for improvement. Another important parameter is the *relevance* of the query preview attributes to the task. Two additional parameters are the *significance* of the query preview attributes in pruning the search space and the *resolution* of the attribute values. For example, if rating *R* is required and almost all the films in the database are of rating *R*, this attribute, although relevant, has insignificant contribution to the performance improvement. When numeric attributes such as the year of production or length of the film are presented in a query preview, the possible values for these attributes are presented using some pre-defined resolution (for example, a 10 year resolution). Tasks which require higher resolution for an attribute than the one provided in the preview will gain a smaller benefit from the preview.

In our experiment, the query preview yielded a greater performance improvement for *T3* tasks (full relevance of the query preview attributes) than for *T2* tasks (partial relevance of the query preview attributes). That result might support the assumption that better relevance of the query attributes to the task yields a greater performance improvement. However, as other parameters might be involved, for example, *significance* of attributes, additional experiments are needed.

4.5. Learning to Use a Query Preview

We found that it was easy for users with experience in querying a database using the form fill-in interface, to learn the query preview interface and take advantage of the information it supplies. After 10 minutes of training, novice performance was only 20-30% slower than experts. However, some of the users, during training and in few cases during the experiment, continued to the refinement phase too early, skipping the examination of one of the relevant attributes. That happened when not all the task attributes could be found in the query preview. For example, when performing a task with conditions on *rating* (in the query preview), *year* (not in the query preview) and *category* (in the query preview), the fact that the *year* could not be specified in the query preview, caused some of the subjects to continue to the refinement stage without examining the information for the *category* attribute. That problem seemed to diminish with experience.

4.6. Subjective Satisfaction

The users statistically significantly preferred the interface with the query preview over the interface without it. They stated that the query preview was helpful, enabling them to search faster and learn more about the database (scores for these questions were statistically significantly above the mid-point). We believe that this subjective satisfaction comes not only from the improvement in performance time which is experienced by the subjects but also from getting better control in performing the tasks.

The suggested improvements related to user interface issues: supplying a way to clear a group of related check boxes in one step, a more immediate refreshing of the preview bars when changing attribute values in the query preview, etc.

The significant preference that subjects showed for including query previews in search engines they currently use (in addition to the objective performance improvement for two of the task types), encourages more effort in understanding and developing query preview interfaces.

5. Conclusions

5.1. Impact for Practitioners

This user study shows that query previews are powerful tools for browsing data warehouses. Query previews give an insight about the database that is being searched and guide users in the query formulation process.

Tasks that have unclear definitions generally lead to longer task completion times in regular form fill-in interfaces. Query previews are very useful in these situations. The benefits obtained depend highly on the relevance of the query attributes to the attributes used in the preview. The costs introduced by the preview are negligible with respect to the benefits (e.g., short delays in query preview load time, implementation costs, extra training for the preview, etc.).

We saw that tasks that have a clear definition (regardless of the relevance of the task to the query preview) were easily executable on a regular form fill-in interface. Query previews were not needed in these situations and, as anticipated, they introduced small delays.

In networked environments, we expect greater benefits from a query preview than the ones we observed in this experiment. As the database size gets larger, we think that the benefits of a query preview will be more appreciated by the user.

An interesting outcome of this study is that the expert user performance and the novice user performance are similar. Hence, the previews are shown to be easy to use, learn, and possibly remember. Besides, due to the application of dynamic querying paradigms (e.g., visual representation of query components, immediate feedback, reversible actions, etc.) they are also highly fault tolerant.

Most of the users preferred the query preview. This is probably due to the fact that users gain greater control and insight about the database while using a preview. Viewing the data distribution over the whole space of records was very helpful for the users. Immediate feedback that was given to users was also found to be very useful. However, relevance of the preview attributes to the most commonly used attributes should be high to maximize the benefits.

5.2. Suggestions for Future Researchers

We suggest that future experimenters explore:

- experiments in networked environments should be run,
- experiments in tandem with other interface types (e.g., dynamic querying) should be considered,
- other task types should be explored, less relevant, less clear, etc.,
- experiments with variety of users should be conducted (with more than two experts from different domains),
- other parameters that effect the query preview usage should be analyzed,
- other data types should be explored: picture, sound, non-relational, etc.,
- a concrete measure of clearness and relevance of the query should be defined, and
- scalability of the query previews with the database size should be analyzed.

5.3. Refining the Theory and Other Suggestions

This experiment is the first user study done on query previews. Previous work suggests [3,4,7] that query previews form a useful means of information exploration in networked systems. This user study on query previews shows that it is a powerful approach to database browsing for various tasks even in non-networked environments.

The experiment shows that task types play a critical role in performance. With this study a taxonomy of task types for querying with query previews was introduced (clear vs. unclear, relevant vs. irrelevant). More concrete measures for clearness and relevance are needed. We recommend that administrators of future studies define the task types in more detail.

The results obtained from this experiment support our hypotheses. We observed numerous benefits in using a query preview. However, there are people who are used to form-fill-in-only approaches. These users might continue using the form-fill-in-only approach and skip extensions to it (unless they receive the necessary amount of feedback). The extensions should enable a smooth and easy transition between the interfaces.

Acknowledgements

This work was partially supported by NASA grant NAG 52895.

References

1. C. Ahlberg and B. Shneiderman, Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays, *Proceedings of the ACM CHI '94 Conference,* 1994, pp. 313-317.

2. C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz, The Harvest Information Discovery and Access System, *Proceedings of the Second International Conference on the World Wide Web*, NCSA, 1994, pp. 763-771.

3. K. Doan, C. Plaisant, and B. Shneiderman, Query Previews in Networked Information Systems, *Proceedings of the Forum on Advances in Digital Libraries*, IEEE Society Press, 1996, pp. 120-129.

4. K. Doan, C. Plaisant, B. Shneiderman, and T. Bruns, Query Previews in Networked Information Systems: A Case Study with NASA Environmental Data, *ACM SIGMOD Record*, 6, 1, March 1997, pp. 75-81.

5. J. Goldstein and S. F. Roth, Using Aggregation and Dynamic Queries for Exploring Large Data Sets, *Proceedings of the ACM CHI '94 Conference*, 1994, pp. 23-29.

6. D. Heppe, W. S. Edmondson, and R. Spence, Helping both the Novice and Advanced User in Menu-driven Information Retrieval Systems, *Proceedings of HCI '85 Conference*, British Computer Society, 1985, pp. 92-101.

7. J. D. Mackinlay, R. Rao, and S. K. Card, An Organic User Interface for Searching Citation Links, *Proceedings of the ACM CHI '95 Conference*, 1995, pp. 67-73.

8. C. North, B. Shneiderman, and C. Plaisant, User Controlled Overviews of an Image Library: A Case Study of the Visible Human, *Proceedings of the 1st ACM International Conference on Digital Libraries*, 1996, pp. 74-82.

9. B. Shneiderman, Dynamic Queries for Visual Information Seeking, *IEEE Software*, 11, 6, 1994, pp. 70-77.

10. B. Shneiderman, D. Byrd, and W. B. Croft, Clarifying Search: A User-Interface Framework for Text Searches, *D-Lib Magazine*, January 1997, available online from the site address: http://www.dlib.org/dlib/january97/retrieval.

11. A. Veerasamy and S. Navathe, Querying, Navigating and Visualizing a Digital Library Catalog. *Proceedings of the Second International Conference on the Theory and Practice of Digital Libraries,* 1995, available online at http://www.csdl.tamu.edu/DL95.

12. C. Williamson and B. Shneiderman, The Dynamic Home Finder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System, *Proceedings of ACM SIGIR '92 Conference*, 1992, pp. 338-346.

Appendix A: Task List

Type1-1) Find a film that satisfies the following constraints given in the following preference order:

- Year 90-95
  - Popularity 5-8
    - Length 90-120 minutes
      - That is related with "Devils"

Type1-2) Find a film that satisfies the following constraints given in the following preference order:

- Year 30-40
  - Popularity 0-5
    - Length 80-100 minutes
      - That is related with "Agents"

Type1-3) Find a film that satisfies the following constraints given in the following preference order:

- Year 85-95
  - Popularity 5-5
    - Shortest available film

Type1-4) Find a film that satisfies the following constraints given in the following preference order:

- Year 82-85
  - Popularity 6-7
    - Longest available film

Type1-5) Find a film that satisfies the following constraints given in the following preference order:

- Latest film by Hitchcock Alfred

Type1-6) Find a film that satisfies the following constraints given in the following preference order:

- Earliest film by Wayne John

Type2-7) Find a film that satisfies the following constraints given in the following preference order:

- PG-13
  - Year 91-95
    - Musical
    - War
  - Year 70-91
    - Musical

        - War
    - Year 70-95
        - Comedy
        - Musical

Type2-8) Find a film that satisfies the following constraints given in the following preference order:

- Award Winning
    - Length 80-85
        - Action
        - Horror
    - Length 90-95
        - Science Fiction
        - Action
    - Length 85-130
        - Western
        - Musical

Type2-9) Find a film that satisfies the following constraints given in the following preference order:

- PG-13
    - Popularity 5-6
        - Science Fiction
        - War
    - Popularity 7-8
        - Science Fiction
        - War
    - Popularity 6-8
        - Drama
        - War

Type2-10) Find a film that satisfies the following constraints given in the following preference order:

- Award Winning
    - Year 60-60
        - Action
        - Horror
    - Year 70-70
        - Science Fiction
        - Action
    - Year 65-66
        - Comedy
        - Horror

Type2-11) Find a film that satisfies the following constraints given in the following preference order:

- PG-13
    - Popularity 7-7
        - War
        - Musical
    - Popularity 8-8
        - War

        - Musical
    - Popularity 6-7
        - Comedy
        - Horror

Type2-12) Find a film that satisfies the following constraints given in the following preference order:

- Award Winning
    - Length 90-100
        - Science Fiction
        - Action
    - Length 100-120
        - Science Fiction
        - Action
    - Length 110-130
        - Drama
        - Action

Type3-13) Find **AT LEAST 5** but **LESS THAN OR EQUAL TO 10** films and then form a collection from these films. Note that the films should be from the **SAME CATEGORY** and they should satisfy the following constraints:

- Award Winning
    - R

Type3-14) Find **AT LEAST 30** but **LESS THAN OR EQUAL TO 40** films and then form a collection from these films. Note that the films should be from the **SAME CATEGORY** and they should satisfy the following constraints:

- No Awards
    - R

Type3-15) Find **AT LEAST 5** but **LESS THAN OR EQUAL TO 10** films and then form a collection from these films. Note that the films should be from the **SAME CATEGORY** and they should satisfy the following constraints:

- No Awards
    - PG-13

Type3-16) Find **AT LEAST 3** but **LESS THAN OR EQUAL TO 5** films and then form a collection from these films. Note that the films should be from the **SAME CATEGORY** and they should satisfy the following constraints:

- Award Winning
    - G

Type3-17) Find **AT LEAST 40** but **LESS THAN OR EQUAL TO 50** films and then form a collection from these films. Note that the films should be from the **SAME CATEGORY** and they should satisfy the following constraints:

- No Awards
    - G

Type3-18) Find **AT LEAST 3** but **LESS THAN OR EQUAL TO 4** films and then form a collection from these films. Note that the films should be from the **SAME CATEGORY** and they should satisfy the following constraints:

- No Awards
    - PG