Natural Search Pointers – A Query Formulation Method for Structured Information Search

Marek Lipczak, James Blustein, and Evangelos Milios
Faculty of Computer Science, Dalhousie University, Halifax NS B3H 1W5, Canada
{lipczak, jamie, eem}@cs.dal.ca

***Abstract*—Despite a wide variety of new solutions, structured information search still has only one practical approach – form-based interface. A key limitation of this interface is poor handling of iterative search. While browsing the results users have to memorize all new search constraints, go back to the form, and enter them into the appropriate fields. To overcome this obstacle we created Natural Search Pointers – a structured information search interface, which formulates search queries based on information highlighted by a user while browsing the search results. NSP can be used as an extension of any standard form-based interface for consumer-oriented database search engines. Comparison of traditional form-based interface and its NSP extension shows that in iterative search tasks NSP makes finding information faster and more convenient.**

I. Introduction

Search, specifically search for information having structure and semantic meaning has been deeply studied for years. One of the main areas of interest is query formulation. Despite a variety of new approaches (e.g., Natural Language Interfaces to Databases [1]) most commercial systems still use only one query formulation technique – form-based interface. Such traditional interfaces take the most important elements of the structure of the database being searched, and present them to users as an interactive search form. For example, search interface of Yahoo! Travel (Fig. 1) allows users to search for hotels using two text fields for home city and destination, two date boxes defining arrival and departure date, and two pull-down lists for number of guests. This one-size-fits-all approach is used in most consumer-oriented search systems (e.g., on-line travel agencies, e-libraries, web auctions, and e-shops). The importance of this type of databases was recently noticed by Google, which launched the Google Base[1] project [2]. Google Base is a tool for searching multiple databases through a unified interface. It includes typical databases (e.g., Products, Houses, Jobs), and users may incorporate new databases or records for their searches.

Traditional search systems have many limitations. They are space consuming and hard to access, as all search constraints must be formulated in separate fields. Query formulation is often cumbersome because users must move from one search field to another, and switch between mouse and keyboard. Tab key used by experienced users does not solve the problem either, as it only allows moving between fields in a predefined order. Despite these problems, form-based interfaces are still the most popular way of creating structured information search queries. Used for years, this traditional solution is probably too deeply rooted to be replaced.

The character of traditional search interfaces prompts us to think about search as a simple task. A user formulates a query, in which all search constraints are defined. The query is compared to a dataset by the retrieval engine. Finally, the search results are browsed by the user. Practically, in many cases these actions are only the start of a complicated search procedure. Users are often not satisfied with the initial search results and decide to reformulate the query (i.e., add, remove, or modify some search constraints). The reformulated query is again processed by the retrieval engine. This loop is repeated until the user is satisfied with the results.

The tool we developed is designed to suit the iterative nature of search. The presented interface follows the idea of Berrypicking search model presented by Bates [3]. He claims that in real life users begin the search process only with a broad idea of their aim. Information they receive gives them a chance to specify their needs. Search process is divided into stages. At each stage, browsing the results allows the users to modify the query and better represent the information need.

To prevent possible ambiguities we present the definitions of two terms used frequently in this paper. By *query formulation* we understand the process of creating search constraints and entering them into an information retrieval system in a form defined by search interface. By *browsing* results we understand the process of reading or skimming through the information collection returned by a search query. This definition of browsing does not imply navigation between documents connected by hyperlinks.

A. Iterative search in traditional interfaces

To present how iterative search is supported by traditional interfaces we use the examples of on-line travel agencies, namely Expedia[2] and Yahoo! Travel[3]. Both interfaces ask the user to fill-in a search form with the most important information (i.e., destination, travel dates, number of travelers). Designers of both interfaces took into account the iterative

[1] http://base.google.com

[2] http://expedia.com

[3] http://travel.yahoo.com

1-4244-1543-8/08/$25.00 ©2008 IEEE

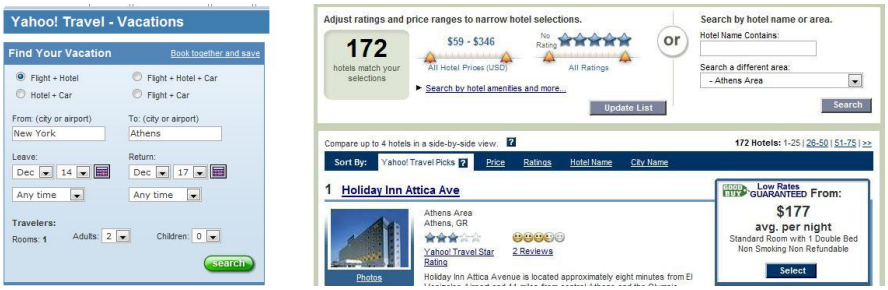


Fig. 1. Yahoo! Travel search interface. The basic search form available on the main page(left), and the advanced interface placed above the results list(right).

character of search, the search results list is accompanied by a new, more advanced search form. Users are often not able to come up with and formulate all important search constraints at the time that they start the search process – some specific constraints (i.e., hotel features or price range) can be defined after the users browse the list of results, and see some of the possible choices (Fig. 1). The same pattern is found in eBay[4], Amazon[5], Dell[6], and many other websites, in some cases the additional search-form is replaced by a predefined list of filters. This approach has two main disadvantages. First, the search process is divided into two separated phases – query formulation, and result browsing. Each time users want to modify search constraints the browsing phase must be stopped to move to the top of the page and reformulate the query. Second, it is the designer's decision which attributes should be available by a search interface. For example Yahoo! interface allows setting the price and hotel quality range, while Expedia interface lists hotel features that users can choose to filter the results. Lack of uniform design requires users to spend time learning each interface they use. This second problem is eased by Google Base. Its interface has a uniform layout for all included databases. In addition, most of database fields presented in the result snippets are searchable via the form. The drawback of this approach is a complex interface with a large number of search fields.

B. Motivation

The main objective of the Natural Search Pointers project is to create a simple search interface that overcomes problems of traditional solution in supporting iterative search. Considering the dominance of the traditional approach, proposing another solution that replaces form-based query formulation seems futile. We instead chose to extend the traditional interface. In fact, Natural Search Pointers interface is even more general, and can be used on top of any keyword-based search interface, as it involves only the list of results.

The main disadvantage of the traditional interface it that it separates query formulation and results browsing. While browsing the results, users often specify their needs based on the presented information. To reformulate the query, they need to memorize all important facts, go back to the page top where the search interface is placed and enter constraints into the form. It would be much easier to highlight these important facts instead. Numerous studies confirmed that highlighting is a well developed and frequently used technique both in paper and electronic environment [4]–[6]. *Natural Search Pointers* interface utilizes this technique to reformulate queries, during the results browsing phase. Users can simply point out information they want to use as a search constraint. There is no need to memorize any information, since a highlighted element is instantly included in the query. Pointing by highlighting is simple and does not require the user to alternate between mouse and keyboard – users can stay focused on the task.

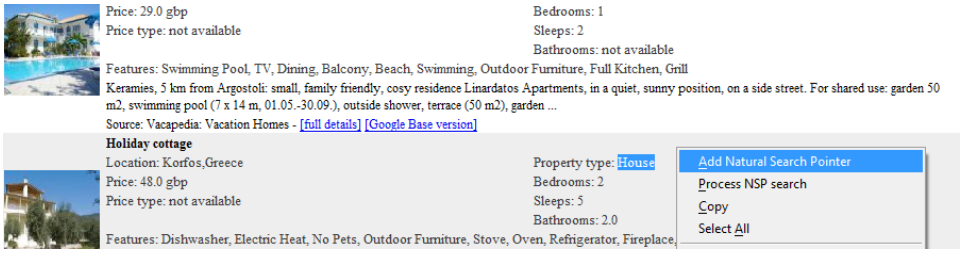
[4]http://www.ebay.com
[5]http://amazon.com
[6]http://dell.com

II. Related work

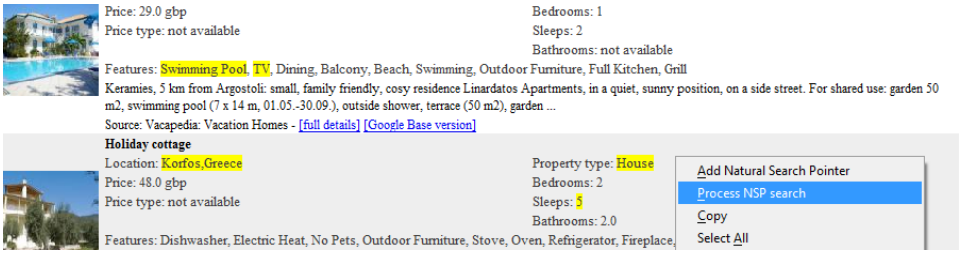
There is a wide variety of research projects on advanced search interfaces. Among them, the most interesting area of research, from the perspective of our work, is combining query formulation and results browsing. Synergy between these tasks is the way to create a valuable information retrieval system [7].

ScentTrails [8] allows users to smoothly alternate between searching and browsing the Web. This interface tries to connect complementary advantages of both tasks: Users formulate keyword queries that modify the collection of websites presented; and links to possibly relevant entries are highlighted. Another interesting approach is presented in WebGlimpse [9]. While browsing a repository, users are given a possibility to formulate search queries that address only documents similar to the one currently read. The system automatically includes browsed context into query constraints. Integrating search and browsing is even more important for databases storing graphics, as search constraints addressing pictures are harder to formulate. Flamenco [10] extracts information by simple keyword queries or meta information filters. These filters (e.g., location, date, objects) are presented as hyperlinks. It helps users to browse through the repository by adding and removing filters. Although presented approaches try to connect the two search phases, complete synergy is questionable because they still are not performed simultaneously.

The problem addressed in this paper is similar to one of PHLAT system features [11]. This personal information manager gives users a possibility to replace or modify a query by retrieved document's attributes (e.g., title). This attribute



(a) User highlights important information and adds it to the query while browsing the results.



(b) Any information presented in the results list can be included to the query, triggered using pop-up menu.

Fig. 2. Natural Search Pointers interface.

may be used as a new query or a filter that narrows previous results. Since PHLAT operates on text keyword queries, it is not clear how such a query may address a specific database structure part. An additional problem is the possibility to choose smaller chunks of information (e.g., single keywords) instead of using the complete attribute. Selecting search terms using a highlighting tool has been shown by Jordan [12]. The system creates definitions of highlighted keywords or phrases based on Wikipedia. The method for formulating queries is very similar to our NSP approach; however, that interface is simpler, it works on unstructured text, and its emphasis is solving ambiguity problems while retrieving the definitions. The advantage of our approach is constructing complex queries for structured information search.

The growing popularity of XML induces researchers to look for new solutions for structured information search. Sengupta and Dillon [13] proposed novel query language (Query By Templates) which gives users a possibility to spatially arrange the query to address appropriate database attributes. Although this approach does not support iterative search, it allows to represent user uncertainty in the queries.

III. Natural Search Pointers Implementation

A. Interface

Natural Search Pointers interface can be an extension of any keyword based search system, including form-based interfaces. It has no influence on the base interface, which remains fully functional.The only difference between the basic and extended interface is that the latter makes results list sensitive to users interaction. The Natural Search Pointers interface is embedded in the results list presented to users. Interaction with the interface starts in the first browsing phase. The main limitation of NSP interface is supporting query formulation based only on the the information presented in a result list. However, the aim of this interface is not to replace, but to overcome the limitations of form-based approach.

Users are allowed to highlight important information presented in the results snippets that should be included in the query. Highlighted text is added to a query by a pop-up menu, as shown in Fig. 2(a). The system automatically recognizes the database attribute associated with the chosen term (e.g., the system knows that highlighted “5” is the number of beds in an apartment). Information about the affiliation of terms is stored by XML tags embedded in the website. After entering the constraints, the search process can be triggered from any place on the website, as illustrated in Fig. 2(b). Users smoothly move from one results list to another. All search pointers are automatically added to the search form. Users can use them in the next search or exclude them from the query. At any moment additional search constraints can be added using the traditional interface. NSP approach reveals the potential of iterative search. There is no distinction between browsing and query formulation phase. Query reformulation is almost effortless, The query can be modified and processed more frequently than in the traditional solution.

To implement the Natural Search Pointers interface we created a fully functional traditional search system, to which the NSP extension was added. The choice of the interface layout is an important issue, as it may influence the way users perceive the complete system. We adopted the style and character of Google Base interface. This is the only standard layout for structured information search, as there is no consensus in commercial systems design. We wrote our own version of the interface because the search engine gets information about structure part association from XML tags. This is an important disadvantage of the current prototype; however, modifying any search website to be NSP compatible can be

done by wrapping all elements in XML tags. Such extension is straightforward, and observing the growing popularity of XML, we may assume that independently websites presenting search results are likely to be NSP compatible.

B. Query processing

The core part of the Natural Search Pointers processing system is a Firefox plug-in that retrieves information about highlighted terms and the XML tags that surround them. The tags convey information about database part association, which defines terms' meaning. Search pointer is a combination of highlighted term (e.g., a group of keywords, or a number) and the database structure part (attribute) it belongs to (e.g., hotel features or price). When a search pointer is highlighted and entered by the pop-up menu, it is transmitted by the Firefox plug-in to the retrieval engine. Pop-up menu is also used to trigger the search process. This approach makes the interaction with the system simple, because users only use the mouse for highlighting terms.

We used Google Base as a source of information. Its API allows the system to formulate queries that address any attributes of Google Base database. It simplified the NSP retrieval engine. Its only tasks are to use search pointers to formulate a query that matches Google Base schema, and later send the list of results to the interface.

IV. Evaluation – User Study Design

Extending the traditional interface introduces some important factors, and complicates the comparison process. Users know the traditional interface, and may refuse to use the features of Natural Search Pointers interface, because they are aware about a way for completing the task only by the traditional solution means. On the other hand, forcing users to work with the NSP interface would bias the results about the practical usability of the extended system. The most important problem in the presented user study was to define tasks that do not promote any of the interfaces. Two questions we wanted to answer through the study were:

- *Do users recognize and utilize new interface features?*
- *Does the NSP interface makes search faster and easier?*

The evaluation was processed on an adaptation of Google Base interface for vacation rental search. This topic was expected to be equally interesting for all participants. We could also expect that all participants have similar level of general knowledge about the topic. The examples of Yahoo! Travel and Expedia show that interface design can have big impact on evaluation results. To mitigate this effect we used a generic Google Base model. The user study was divided into two phases. Users performed two tasks that examine different aspects of evaluated search interfaces. Before the recorded session, participants completed a practice session.

A. Tasks

Simulation of real search interaction is hard, because in real life search needs are highly dependent on user preferences, background information, and context. We evaluated the system on two tasks with different ways of information need formulation. In the first task, the complete information needed to find a specific offer was presented before the test. In the second task the information was given to a participant gradually.

1) "Find exact offer" task: The participant was given a paragraph describing vacation rental offer. The description contained factual information about the offer (e.g., city name) and four additional constraints which allowed the participant to focus the query and reduce the list of results to one page. The latter was the objective of the test, as one page of results can be easily browsed manually.

This task allowed the participant to freely use any of the presented information. The participant made all decisions about query formulation. Such a setting objectively examined all participant's actions; however, it was not a good simulation of real life search. It is very uncommon to search for explicitly presented information. In addition, this task was likely to promote the traditional interface. Ideally, all information could simply be entered into the appropriate form fields, and search could be processed in one iteration. The main aim for this task was to check the lower bound of the extended system.

2) "Interaction based search" task: In this task, the interviewer's aim was to give the participant information about search objectives gradually. To make the situation realistic the interviewer proposed a problem, which should have been solved together with the participant. During the search process the participant was guided step-by-step by the interviewer.

The participant was limited by the proposed search constraints. To keep the test similar for all users all tasks consisted of three iterations; during the iteration additional information was presented roughly after a certain number of browsed offers. The objectivity of this approach is questionable, because in dialog the participant could ask additional questions and take the responsibility for the task by proposing additional search constraints. However, this task was the only possibility to simulate real search situation that follows the Berrypicking idea discussed in Section I. This task was supposed to point out any advantages of Natural Search Pointers interface, as it emulated the iterative approach to search.

B. Study settings

"Find exact offer" and "Interaction based search" tasks were meant to examine the system from different perspectives. The tasks could be performed within the same session in predefined order. We could neglect the experience factor because participants were allowed to learn the system features beforehand.

There were two unrelated factors that must have been controlled: interface learning and search query difficulty. When a user performs tasks on two interfaces, which are somehow similar, the results for the second one are biased because the user has acquired experience on the previous version of the interface. A participant remembers actions and outcomes of the performed search task. We could expect that finding the same information the second time would take much less time. To avoid it, a participant was given different set of search queries for two interfaces. Furthermore, the difficulty of search

tasks must have been equalized. A randomization method [14] was used to remove the impact of irrelevant factors on the results. We used a "within subject" (repeated measures) design using Latin square [15] to control for personal variation while testing the interface version factor. Sixteen participants were randomly assigned to four groups. The groups were diversified based on the order of interface versions and sets of queries. Each participant performed three search queries for each combination of task and interface.

C. Indicators

The most objective and explicit indicator of system usability is the time a user takes to perform the task using the interface. We compared the average amount of time spent on each task using both interfaces.Another useful indicator of system usability are users' opinions. Each participant was asked to fill out a self-report after completing the study. The self-report was adapted from questions presented by Douglas et al. [16], and Hornbæk and Hertzum [17].

V. Evaluation – Results

Simplicity of tasks and general familiarity with search topic relaxed the requirement of participants computer experience. In fact, we were more interested in users with no extensive computer science knowledge. All participants were students (7 computer science students, and 9 students of other disciplines, of which 7 were women and 9 men.

The average time participants spent on browsing and query formulation phases is shown in Fig. 3(a). The length of the first iteration is presented separately as in all cases it was a general keyword-based query (e.g., city name) which was used to focus the search area. The search length results for the first task – "Find exact offer" show a large advantage of the traditional interface. The search length for the interface with Natural Search Pointers was mainly extended by the browsing phase. Although the participants were informed that they should perform search in the most convenient way, and the usage of Natural Search Pointers in not obligatory, 11 participants used NSP to enter the queries. They were browsing the list of offers looking for relevant information that could be used as a search pointer, where in this task, the simplest approach was to copy the given information into the search form. Five users noticed this fact and stopped to use NSP during the first task. The potential of Natural Search Pointers was revealed in the second task – "Interaction based search", where new search constraints were formulated during the search phase. Although some users with no computer science background found using the pop-up menu hard, all participants who used Natural Search Pointers spent less time on query formulation phase. The average accumulated duration of query formulation phase was 14.1 seconds compared to 21.9 seconds for the form-based interface, which is a statistically significant difference according to dependent t-test ($p < 0.05$). Only one user decided not to use NSP in this task.

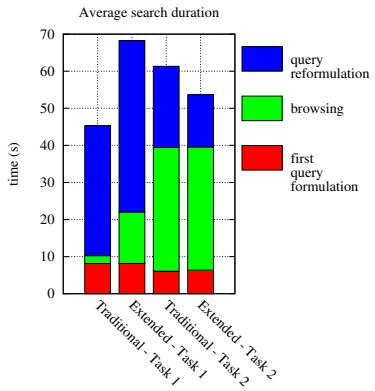
Two search tasks showed the users the limitations and potential of NSP interface. We believe that experience of these tasks allowed the participants to fairly evaluate two versions of the interface in a self-report. The search questions were divided into two groups. The first group addressed specific issues of the search interface use. The second group gave the participants the opportunity to express their feelings about the interface. All responses were collected using a 5-point scale. For the first group of questions the largest difference is noticeable in the mental effort needed to use the interface. Possibly, the users found the need of looking for an appropriate search field and manually entering of search constraints discouraging. Even considering the inconveniences in the first task, the interface with NSP extension was rated higher from the perspective of search speed. The responses for the second group of questions show that the participants appreciated the flexibility of the interface with Natural Search Pointers. They also found working on it more satisfying. Surprisingly, both interfaces were judged as very easy to use. It is likely that the participants judged the interfaces from the perspective of the given tasks, which were easy to perform.

Generally, the user study gave satisfactory answers to the two main research questions. A diversified group of users allows us to assume good generality of the results. The users easily recognized the functionality of NSP extension and used it frequently. The NSP interface allowed users to perform iterative search tasks faster and easier. The results of the first task show that NSP is not a good solution for search of specific and well-defined information. This conclusion does not reduce the usability of the NSP interface, as it is meant to extend, not replace the traditional approach. In practice, we expect the users to quickly find appropriate contexts of the NSP interface. We could observe the example of such behavior in the study, where 5 participants stopped to use NSP interface in the first task. They all used it again in the second task.

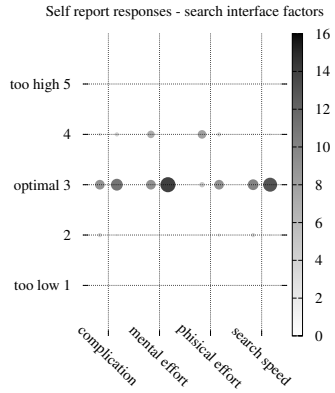
VI. Conclusions and future work

Search as an iterative process is not supported well by the traditional form-based interface. Users are not able to switch smoothly between the two phases of the search process: query formulation and results browsing. We designed an extension of the form-based approach, which allows reformulating search queries during the results browsing phase. At the same time, the extension preserves the full usability of the traditional solution. Natural Search Pointers interface allows users to highlight interesting information they found in the results list and use them as search constraints. This feature was implemented as a Firefox plug-in. We created a fully functional search interface for search in structured information based on Google Base API. The evaluation suggests that NSP should not be used as a stand-alone interface, because searching for specific information can be completed faster using form-based interfaces. NSP potential can be utilized in iterative search. Such tasks can be performed faster, with higher comfort and satisfaction than the usual methods.

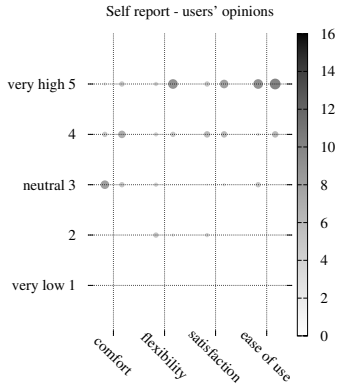
The participants of the user study suggested that the list of results can be even more interactive. The next version of the interface will indicate the element that can be used as a



(a) The search duration for "Find exact offer" (task 1), "Interaction based search" (task 2). Both tasks were performed on two versions of the interface: form-based and form-based extended by NSP. Time needed to perform search is presented as a sum of the first iteration duration, and accumulated duration of browsing and query formulation phases in next iterations.



(b) Each vertical line represents one search interface factor. The participants indicated the distance from a "perfect" interface. The responses for the form-based interface are shown on the left side of the line, the right side is for the interface extended by NSP. The size and intensity of point reflect the number of answers.



(c) Each vertical line represents a general opinion about an interface. The participants were asked to indicate their feeling about a given factor from very low to very high. The layout is identical to the one described for Fig. 3(b).

Fig. 3. Two types of evaluation results. The average duration of search process calculated from logged time stamps, and users' opinion expressed in self-report.

pointer by a small icon appearing while a mouse cursor is over it. The icon can be used to enter the pointer to a query. AJAX techniques will be used to refine the list of search results without the need of refreshing the page, keeping the current offer in the same position. It will make the switch between browsing and query formulation phase completely seamless.

An advantage of Natural Search Pointers interface, not discussed in this paper, is high quality and complexity of user feedback information. The interface may be helpful for users with unspecified, general needs (e.g., users that look around for offers to see what are the possible choices). Asking the users to point out any interesting information may be used to create a rich user profile. The system is able to gather information about pointed terms and the information context in which they were entered (e.g., browsing a list of vacation rental offers, a user highlights information related to specific offers). Data mining techniques, namely classification and association rule mining, can be used to propose offers that suit users' needs best. Adding this recommendation feature to the NSP interface is the topic of our future work on the project.

ACKNOWLEDGMENT

We would like to thank Chris Jordan from Dalhousie University for his help with Firefox plug-in implementation and insightful comments on the NSP interface.

REFERENCES

[1] I. Androutsopoulos, G. Ritchie, and P. Thanisch, "Natural Language Interfaces to Databases–an introduction," *Journal of Language Engineering*, vol. 1, no. 1, pp. 29–81, 1995.

[2] J. Madhavan, A. Halevy, S. Cohen, X. L. Dong, S. R. Jeffery, D. Ko, and C. Yu, "Structured Data Meets the Web: A Few Observations," *IEEE '06: Bulletin of the Technical Committee on Data Engineering*, vol. 31, no. 4, pp. 19–26, December 2006.

[3] M. J. Bates, "The design of browsing and berrypicking techniques for the online search interface," *Online Review*, vol. 13, no. 5, pp. 407–424, 1989.

[4] C. C. Marshall, "Annotation: from paper books to the digital library," in *DL '97: Proc. the second ACM international conference on Digital libraries*. New York, NY, USA: ACM, 1997, pp. 131–140.

[5] H. Obendorf, "Simplifying annotation support for real-world-settings: A comparative study of active reading," in *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*. New York, NY, USA: ACM, 2003, pp. 120–121.

[6] B. N. Schilit, G. Golovchinsky, and M. N. Price, "Beyond paper: supporting active reading with free form digital ink annotations," in *CHI '98: Proc. the SIGCHI conference on Human factors in computing systems*, New York, NY, 1998, pp. 249–256.

[7] J. D. Mackinlay and P. T. Zellweger, "Browsing vs. search: Can we find a synergy?" in *CHI '95: Conference companion on Human factors in computing systems*. New York, NY: ACM, 1995, pp. 179–180.

[8] C. Olston and E. H. Chi, "ScentTrails: Integrating browsing and searching on the web," *ACM Transactions on Computer-Human Interaction*, vol. 10, no. 3, pp. 177–197, 2003.

[9] U. Manber, M. Smith, and B. Gopal, "WebGlimpse: Combining browsing and searching," in *Proceedings. 1997 Usenix Technical Conference*, 1997.

[10] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst, "Faceted metadata for image search and browsing," in *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, 2003, pp. 401–408.

[11] E. Cutrell, D. Robbins, S. Dumais, and R. Sarin, "Fast, flexible filtering with PHLAT," in *CHI '06: Proc. SIGCHI on Human Factors in computing systems*. New York, NY: ACM, 2006, pp. 261–270.

[12] C. Jordan, "Using Wikipedia as a knowledge base for electronic documents," in *Proceedings of the JCDL07 Doctoral Consortium*, 2007.

[13] A. Sengupta and A. Dillon, "Query by templates : Using the shape of information to search next-generation databases," *IEEE transactions on professional communication*, vol. 49, no. 2, pp. 128–144, 2006.

[14] J. E. McGrath, *Methodology matters: doing research in the behavioral and social sciences*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 152–169.

[15] J. Schinka and W. F. Velicer, *Handbook of Psychology, Volume 2, Research Methods in Psychology*. John Wiley & sons, 2002.

[16] S. A. Douglas, A. E. Kirkpatrick, and I. S. MacKenzie, "Testing pointing device performance and user assessment with the iso 9241, part 9 standard," in *CHI '99: Proc. SIGCHI conference on Human factors in computing systems*. New York, NY: ACM, 1999, pp. 215–222.

[17] K. Hornbæk and M. Hertzum, "Untangling the usability of fisheye menus," *ACM Trans. Comput.-Hum. Interact.*, vol. 14, no. 2, p. 6, 2007.