`Annotate!` A Tool for Collaborative Information Retrieval

Mark Ginsburg
Doctoral Student, Stern School of Business, New York University
44 W 4 St., 9-181 MEC
New York, NY 10012 USA
mark@edgar.stern.nyu.edu

Abstract

Difficulties with web-based full text information retrieval (IR) systems include spurious matches, manually intensive document sifting, and the absence of communication or coordination between users.

To address these difficulties, we introduce the `Annotate!` *system which enables document annotations, and captures global usage history.* `Annotate!` *provides improved data and metadata clues to guide the user in a search session.*

Two data sets, declared in XML, are at the core of `Annotate!`*:* ***discussion*** *data, a composite of documents and user annotations and* ***session*** *data which captures user timings at the various interface layers. We discuss a prototype implementation, and show that the collaborative infrastructure enabled by* `Annotate!` *can be predicted to improve the diffusion of ideas in the search community.*

1. Introduction

Despite rapid progress in hardware computing platforms in recent years and equally rapid advances in the popularity of the Web as a distributed hypermedia publishing system, information retrieval remains a knotty problem both in the Internet and in organizational intranets.

Popular web-based full text search (WFTS) engines such as Excite, Lycos, and Alta Vista use a simple hypermedia interface on the front-end Query Interface, but on the back end the Retrieval Interface is simply an array of hyperlinks pointing back to source documents. From the user's perspective, the chief failing of an ad-hoc Web IR system is the lack of data and metadata clues in the search, retrieval, and document browsing interfaces. The three interfaces are (a) the Query Interface (entry point of a search engine), (b) the Retrieval Interface (a typical result from a full text search engine is a ranked array of hyperlinks, to base documents), and (c) the Document Interface (the result from clicking on a link in the Retrieval Set is typically a single document browsing session).

Without sufficient signposts, the ad-hoc user is vulnerable to the traditional full text problems of homonymy and synonymy; there is no metadata (data about the documents) available at the Document layer; author and timestamp information must be explicitly represented in the document body. The limited set of clues coupled with the semantically weak *GOTO* mechanism [16] coupling the Retrieval and Document layers makes for continued inefficiency in users' interactions with a web search system.

The user must navigate the Retrieval Set, one document at a time, with very limited clues (e.g. a summary paragraph, the document's title, and the engine's confidence score) to indicate if that particular document might be important to solving the original problem. Compounding the problem, the document may be of poor quality and hence not worth the time and trouble to browse its contents.

Given that the average user is constrained both by time and patience in the course of a search session [5], the Retrieval Interface poses a formidable challenge to the user — he or she must browse, laboriously, one document at a time from the Retrieval Set with only limited data and metadata clues as signposts that might point to a document actually relevant for the problem at hand. And, since web-based search engines typically do not circumvent the statelessness of the Web's Hypertext Transfer Protocol (HTTP) we have no memory is kept between search sessions for a given user, and experiences for better or worse with specific queries cannot be communicated between users.

We present a software tool, `Annotate!`, to address these challenges by adding a collaborative dimension to Web full text search. Our collaborative implementation accomplishes two goals. Firstly, we transform single author documents into multiple participant discussion fora, thus doing away with the sometimes artificial distinction between author and reader in traditional electronic publishing [16]. Secondly, `Annotate!` serves as an example of how we add state to an intrinsically stateless Web appli-

cation to enable an organization to learn from usage history. As Garvin states, "learning organizations are skilled at ...learning from their own experience and past history ...and transferring the knowledge quickly and efficiently throughout the organization" [6].

Section 2 discusses the design principles of `Annotate!` which are essentially high-level heuristics for engineering collaboration.

Section 3 presents the two key data and metadata data stores, *discussion* data and the *session* data, that `Annotate!` utilizes. Discussion data grows from a document annotation facility which allow readers to act as secondary authors; document appraisals and other semantic markup are represented by interface changes. It also enables social filtering, which is another class of interface change. Session data accrues with system usage: queries, results, and user trajectories through the results.

Section 4 demonstrates an implementation of `Annotate!` using eXtensible Markup language (XML) Document Type Definitions to declare the Discussion and Session templates.

Section 5 reviews the specific technical and general organizational benefits we expect to accrue from ongoing use of `Annotate!`.

Finally, Section 6 presents concluding remarks and mentions an ongoing field study which uses the `Annotate!` system.

2. Engineering Collaborative Information Retrieval

From an ad-hoc application in which every user is isolated from himself or herself in the time dimension (no session memory) and from the other users in proximate or distant work groups in the time and distance dimension (no group query memory, no group identifiers) our goal is to move to a state-oriented application that provides useful clues as the user moves through search, retrieval, and document browsing.

We follow a simple six-step heuristics framework from the starting point of a a non-collaborative application in a hypertext system such as the Web which has an identifiable (known entry point, known intermediate layers, and known exit conditions) user session.

Given these conditions are satisfied, we can write the following high-level heuristics to engineer collaboration:

1. First identify the interface layers involved in the user session. Map these to a start, in-process (intermediate), or end signal.
2. identify data and metadata *clues* in these layers — *clues* are interface signposts to help the user get to the most helpful next interface layer.
3. get a sense of timings of a typical user session passing through these layers.
4. identify discrepancies in (2) and order them by (3) as a crude measure of importance
5. semantically declare data and metadata composite data structures to redress (4) and identify the 'collector' interfaces that can create new instances of this data.
6. form an output map of components of data declared in (5) to one or more layers identified in (1). This defines which elements in a given dataset will participate in which 'target' interface alteration.

Specific implementations use two further guidelines:

1. choose appropriate statistical or agent technology to define trigger rules by which data instances will alter interfaces by the mapping scheme defined in (6).
2. Follow accepted user-interface principles in the interface alteration choices. For example, usability studies offer evidence that users react well to human face icons [9]; we adapt face icons in the implementation in Section 4.

3. Adding Collaboration to Web-based Full Text Search

We can apply the collaborative framework to organizational web full text search.

Following the heuristics set forth in Section 2, the first step is to examine the hypertext flow of the application. In the case of WFTS, we have three basic layers with a simple navigational path:

$$Query \Rightarrow Retrieval \Leftrightarrow Document$$

The lack of clues is apparent if we consider a typical user session. The user provides input at two stages. In the Query Layer, he or she inputs keyword(s). In the Retrieval Layer, hypertext links are followed to source documents. This commences the manually intensive phase of browsing source documents, returning to the retrieval set, and continuing. The important Retrieval Layer only contains the document title and a search engine confidence score, often insufficient to guide the user's descent into the time-consuming Document Layer.

To redress the lack of clues in the user navigation, we can attempt to improve the IR system with *discussion* instances which are combinations of *core document* and *multiple annotations*. An implementation of this data structure is presented in Section 4.

Continuing with our heuristics, we need to map discussion components to target interfaces. We map appraisals to a document recommender form on the Query interface, and we map annotation metadata (identifiers, reasons and again, the appraisals) as icons in the Retrieval interface.

4. Implementing Collaborative Information Retrieval

`Annotate!` is layered on top of the Excite core search engine with a series of small server-side Perl modules. Excite is a convenient choice because the distribution (which is no-cost, licensed software available for many operating systems platforms) includes the source code for the interface libraries. These libraries control the look and feel of the application on the front-end (the user entering keyword(s) to search) and the results returned from the search engine. We take advantage of the source code availability to make our modifications. Note that `Annotate!` is independent, though, of the core search engine and can be used in conjunction with experimental search algorithms [5] or enhanced user-interface frameworks [14].

4.1. The Discussion Document Type Definition

We define the Discussion Document Type Definition in XML[1].

In the DTD, which uses regular expression syntax, we define the **discussion** dataset as exactly one **document** with *zero or more* annotations.

Here is an excerpt of the Discussion DTD:

```
1<!element discussion
2  (document, annotation*) >
3  <!element document
4    (mddocument*, ddocument) >
5    <!element mddocument
6     (d-creator?, d-timestamp?,...  )
7     <!element d-creator   (#PCDATA)>
8     <!element d-timestamp (#PCDATA)>
9     <!element ...                 >
10    <!element ddocument (#PCDATA)>
11 <!element annotation
12   (mdannotation, dannotation) >
13   <!element mdannotation
14     (ident, annocontext, ... ) >
15     <!element ident
16     (name?, busunit, ... ) >
17      <!element name  (#PCDATA)  >
18      <!element busunit (#PCDATA)>
[...]
22      <!element annocontext(#PCDATA)>
23      <!attlist annocontext reason
24  (agree| agree with reservations |
25     see for more information     |
26     a more general lesson
27     can be drawn|out of date     |
28     errors exist | material
       superseded by | general comment)
29     ''general comment'' >
30   <!attlist annocontext agree-rtg
31   (1|2|3|4|5|6|7 ) '' no comment ''
32   <!attlist annocontext quality-rtg
33   (1|2|3|4|5|6|7 ) '' no comment ''
34   <!element annotime (#PCDATA) >
35   <!element expirestime (#PCDATA)>
36   <!element dannotation (#PCDATA)>
37   >
```

The core document *may* have metadata (line 4) but each annotation piece *must* have metadata (line 12).

The annotation metadata has an identifier section in which we can capture detail about the individual or higher-level information such as the business unit which we can derive in some organizations from the annotator's IP address.

We also define an annotation context field with two numeric attributes: *agree-rtg* (line 30) and *quality-rtg* (line 32). These attributes, if the user chooses to fill them in, can be aggregated and used as the basis for a social filter system.

Starting with a core document that is fundamentally unstructured (poor semantics) we logically bind annotations with strong and unambiguous semantics to improve retrieval on the hybrid data instance.

4.2. Why XML?

Whereas HTML represents a centralized effort to impose a lowest common denominator for document rendering (a *presentational* markup language), XML offers us key benefits as a *semantic* markup language: it is extensible, validatable by external modules, and its tags are self-documenting [8]. Also, the DTD can be posted and imported from the network helping revise a prototypic annotation structure quickly at a particular organization [4].

4.3. The Session Data Store

The session data store captures timings at the various interface layers as follows: in the Query layer, we record the server time when the user submits a query. The key of the timing data structure is a concatenation of the client's IP number and the query submission time. Once we establish this key on the server, we can append further timings. When

[1]The complete Discussion and Session DTDs are available at `http://edgar.stern.nyu.edu/xml/`

the Retrieval List is created, we use Javascript to record the start time (the 'load') of the Retrieval window and the end time (the 'exit' from this window, as a user follows a document link). The end time of the Retrieval List represents the start time reading a document. When the user exits the Document Layer and returns to the list, the new Retrieval List start time serves as the effective end time of the prior document read (inflated only by the network delay between page loads). The only data missed by this technique is the time spent on the last document of the user's session.

Here is an example record with linebreaks added for readability:

```
112.23.32.29:3 |
/my/path/to/foo.html  7 8 ||
/my/new/path/to/bar.html  11 13 ||
/my/other/path/to/plugh.html  19 22
```

This would result in a browsing time of $11 \Leftrightarrow 8 = 3$ seconds for *foo.html* and $19 \Leftrightarrow 13 = 6$ seconds for *bar.html*. Note this technique does not reveal the time spent browsing the last document in the chain; in this case, *plugh.html*.

The session data also contains useful data on the users' *query, retrieval set, document selection* session records. Unimplemented, but an interesting approach, is to take the Query and Retrieval Set as a *situation* and the user's document navigation as an *action*. Then, we can use the statistical technique Memory Based Reasoning (MBR) [15] to help guide future users whose *situations* are close to prior ones. It would be an interesting empirical test to implement MBR as an additional signpost at the Retrieval layer[2].

4.4. Integrating the Discussion Data Store with Search Interface Layers

Having written the Discussion template DTD, we can now examine the integration between the discussion data store and the various interface layers.

Figure 1 presents the high-level view of the relationship between the Discussion instances and the various interface layers. Thus the Document layer acts as a Collector Interface for user annotations which grows the Discussion instances. In turn, simple interface rules modify the Retrieval layer.

As we see in Figure 1, the document annotation facility in the Document layer may be used to alter both the Retrieval and Query Interfaces. Following other recent work [2] [12], we use document appraisals as a social filter system.

In our simpler implementation we set up an interface agent (software changing the interface on behalf of the user without the necessity of manual launch [11]) between the Discussion data store and the Retrieval Interface with a rule of displaying the two most recent annotations in the Retrieval Interface.

Figure 2 shows an example of the updated Retrieval Interface after some annotation of a base repository. The column headings are as follows: G is the general reason for annotating; the light bulb icon represents "a more general idea can be drawn"; the eye icon represents "see also this link ..." and so on. F and Q are user satisfaction measures with the factual accuracy and quality of the document, respectively, on a Likert 1—7 scale. These are mapped to a spectrum of facial expressions. We also present summary statistics: $\bar{F}$ is the mean factual accuracy rating and $\bar{Q}$ is the mean quality rating. The standard Excite confidence score is preserved with the modification of different colors reflecting levels of annotation activity: from red (very active) to green (medium) to blue (inactive).

If the user elects to create an annotation for a selected document, he or she can appraise the document on the dimensions of quality and factual accuracy, and give the reason for annotating[3].

5. Benefits of Collaborative Information Retrieval

The first major advantage of our collaborative information system is an improved and dynamic growing set of data and metadata clues available to the user in the various interface layers.

Using the Retrieval metadata signposts should assist the user in minimizing wasted browsing time at the Document layer but this must be tested empirically. The Retrieval icons which arise from annotations are powerful in many ways — they do not require user effort since they are bound to the interface changes via automated alteration rules, they reflect immediacy (the most recent annotations), assist navigation into the Document layer, and in the aggregate enable a social filter at the Query layer.

A potential second class of benefits relates to the search process itself. If a user finds a document of marginal value according to the available interface clues, but decides to pursue it anyway, it is possible that a general lesson may spring out of the document. As Vannevar Bush said in 1945 [3], the "human mind ... operates by association. With one item in its grasp, it snaps instantly to the next that is suggested by the association of thoughts." Our implementation

[2] Similar work has been done with the Eudora e-mail client, with the e-mail header and message being the *situation* and the user's response (deletion, forwarding, etc.) being the *action* [10].

[3] `Annotate!` can be extended to link annotations to core search actions. For example, if the user commits a "see" reference to a document, a spider could canvass *see also* ... links, pick up those contents if reachable on the network, and index them as well. Thus usage of `Annotate!` would not only increase interface signposts, it would also grow the underlying document collection

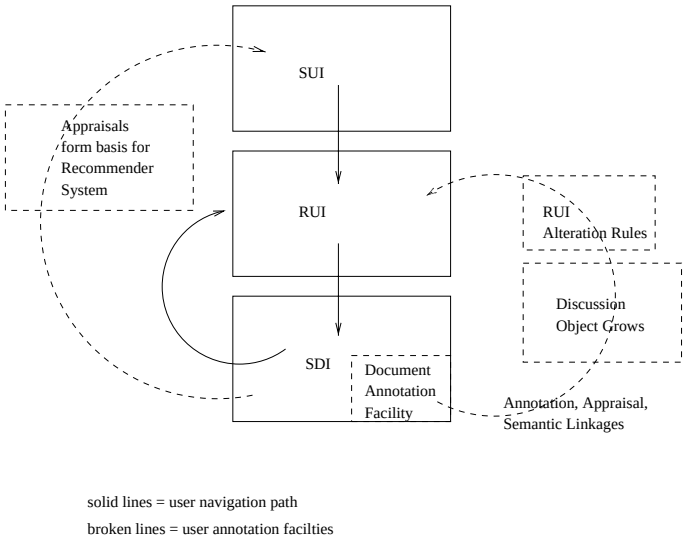

Figure 1. The Discussion Data Store influences, or is influenced by, all of the `Annotate!` Interface Layers

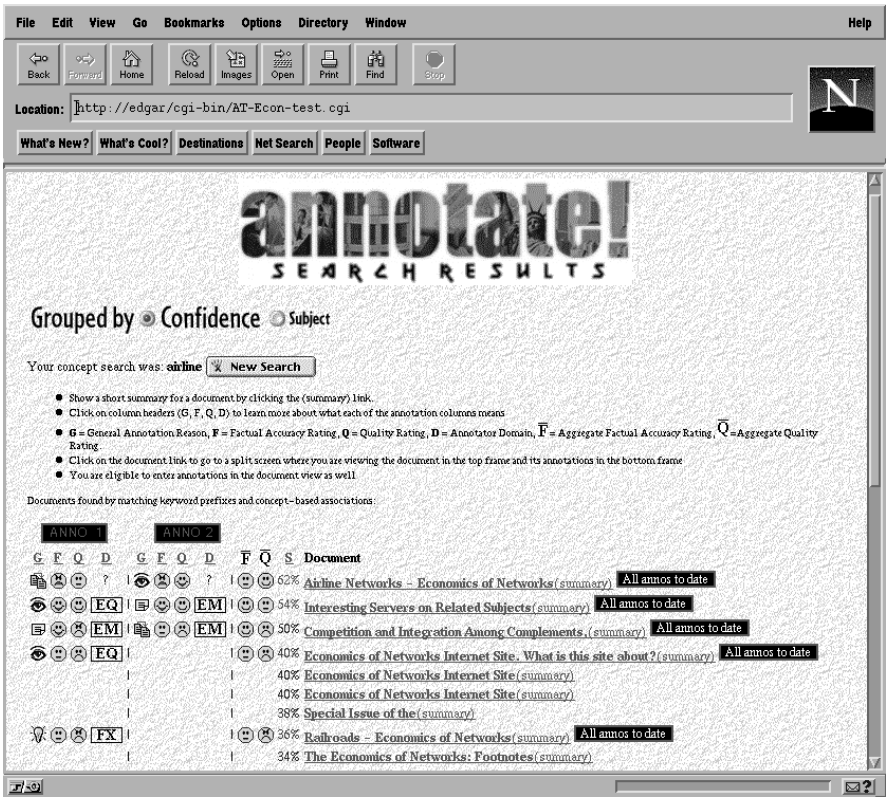

Figure 2. Screen shot of the `Annotate!` System: Retrieval Layer

captures such associations and allows them to be shared between users.

Permitting the seekers to become authors permits them to record their associations for global use. Over time the system grows in power; it continues to function at a base level statistically on unordered keyword queries, but interface clues and additional indexed annotation components add to the power of the overall system. By achieving state and enabling peer-to-peer authorship, we expect that the dissolution of the conventional Author-Reader barrier [16] should lead to a more widespread distribution and discussion of document repositories. Overcoming HTTP statelessness should help to build an organizational memory store (Ackerman) [1], and thus facilitate organizational learning [6]. In a federalist organization with semi-autonomous business units [13], `Annotate!` should improve diffusion of ideas across the units. `Annotate!` should also improve quality assessment since it is explicitly modeled in our discussion template DTD, and improve the management of expertise since more efficient retrieval on a work product better justifies the time spent in its creation.

Finally, `Annotate!` should improve knowledge management[7] — if more readers are retrieving, browsing, and commenting on the documents (more 'lively' repositories) it is more likely that the greater information throughput (even in business units far removed potentially from the document source) will increase the growth of the firm's knowledge base on a query by query basis.

6. Conclusions

6.1. General Remarks

This work represents an attempt for an organization to manage its intellectual resources more effectively. "An organization must retain knowledge of its past efforts" [1] and this suggests a memory store should be made available whenever possible, particularly in the malleable and important Information Retrieval function.

6.2. Ongoing Work

We hope to show, in a field test currently being conducted with this tool in a large federalist organization, that our collaborative dimension will improve quantitative and qualitative IR measures. The control group uses a modifed version of Excite that supports distributed Sparc/Unix and Lotus Domino search from a single entry point, and the experimental group uses `Annotate!` with an identical entry look and feel.

We expect the benefits should accrue the more the system is used. If successful, it will demonstrate the worth of having the everyday activity of searching and retrieving information from text archives contribute to an aggregate organizational memory store. Such a store enables the organization to learn from this common task.

Future plans include the exploration of social issues in anonymous versus non-anonymous annotation variants, and the field testing of the `Annotate!` system in firms with varying document repository compositions, sizes, and initial distribution.

References

[1] M. Ackerman. *Answer Garden: A Tool for Growing Organizational Memory*. PhD thesis, Massachusetts Institute of Technology, 1994.

[2] C. Avery and R. Zeckhauser. Recommender systems for evaluating computer messages. *Communications of the ACM*, 40(3):88–89, March 1997.

[3] V. Bush. As we may think. *Atlantic Monthly*, 176(1):101–108, July 1945.

[4] D. Connolly, R. Khare, and A. Rifkin. The evolution of web documents: The ascent of XML. *World Wide Web Journal*, 2(4):119–128, 1997.

[5] J. W. Cooper and R. J. Byrd. OBIWAN — a visual interface for prompted query refinement. In *Digital Documents*, volume 2, pages 277–285. 31st Hawaii International Conference on System Sciences, IEEE, January 1998.

[6] D. A. Garvin. Building a learning organization. *Harvard Business Review*, pages 78–91, July-August 1993.

[7] M. Ginsburg and A. Kambil. Annotate! a web-based knowledge management support system for document collections. NYU Stern School of Business Working Paper No. IS-98-13, 1998.

[8] R. Khare and A. Rifkin. The origin of (document) species. In *Proceedings, 7th World Wide Web Conference*, Brisbane, Australia, April 1998. World Wide Web Consortium, W3C.

[9] T. Koda and P. Maes. Agents with faces: The effects of personification of agents. In *Proceedings of HCI'96*, London, 1996.

[10] Y. Lashkari, M. Metral, and P. Maes. Collaborative interface agents. In *Proceedings of AAAI'94*, Seattle, Washington, August 1994. AAAI.

[11] P. Maes. Pattie Maes on software agents: Humanizing the global computer. *IEEE Internet Computing*, pages 10–19, July-August 1997.

[12] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, March 1997.

[13] J. W. Ross and J. F. Rockart. Enabling new organizational forms: A changing perspective on infrastructure. *Proceedings of the International Conference on Information Systems (ICIS)*, December 1996.

[14] B. Shneiderman, D. Byrd, and W. B. Croft. Sorting out searching: A user-interface framework on text searches. *Communications of the ACM*, 41(4):95–98, April 1998.

[15] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[16] C. Watters, M. Conley, and C. Alexander. The digital agora: Using technology for learning in the social sciences. *Communications of the ACM*, 41(1):50–57, January 1998.