Using Annotations in Enterprise Search

Pavel A. Dmitriev
Department of Computer Science
Cornell University
Ithaca, NY 14850
dmitriev@cs.cornell.edu*

Nadav Eiron
Google Inc.
1600 Amphitheatre Pkwy.
Mountain View, CA 94043*

Marcus Fontoura
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA, 94089
marcusf@yahoo-inc.com*

Eugene Shekita
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
shekita@almaden.ibm.com

ABSTRACT

A major difference between corporate intranets and the Internet is that in intranets the barrier for users to create web pages is much higher. This limits the amount and quality of anchor text, one of the major factors used by Internet search engines, making intranet search more difficult. The social phenomenon at play also means that spam is relatively rare. Both on the Internet and in intranets, users are often willing to cooperate with the search engine in improving the search experience. These characteristics naturally lead to considering using user feedback to improve search quality in intranets. In this paper we show how a particular form of feedback, namely user annotations, can be used to improve the quality of intranet search. An annotation is a short description of the contents of a web page, which can be considered a substitute for anchor text. We propose two ways to obtain user annotations, using explicit and implicit feedback, and show how they can be integrated into a search engine. Preliminary experiments on the IBM intranet demonstrate that using annotations improves the search quality.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Anchortext, Community Ranking, Enterprise Search

1. INTRODUCTION

With more and more companies having a significant part of their information shared through a corporate Web space, providing high quality search for corporate intranets becomes increasingly important. It is particularly appealing for large corporations, which often have intranets consisting of millions of Web pages, physically located in multiple cities, or even countries. Recent research shows that employees spend a large percentage of their time searching for information [16]. An improvement in quality of intranet search reduces the time employees spend on looking for information they need to perform their work, directly resulting in increased employee productivity.

As it was pointed out in [15], social forces driving the development of intranets are rather different from the ones on the Internet. One particular difference, that has implications on search, is that company employees cannot freely create their own Web pages in the intranet. Therefore, algorithms based on link structure analysis, such as PageRank [24], do not apply to intranets the same way they apply to the Internet. Another implication is that the amount of anchor text, one of the major factors used by Internet search engines [14, 1], is very limited in intranets.

While the characteristics of intranets mentioned above make intranet search more difficult compared to search on the Internet, there are other characteristics that make it easier. One such characteristic is the absence of spam in intranets. Indeed, there is no reason for employees to try to spam their corporate search engine. Moreover, in many cases intranet users are actually willing to cooperate with the search engine to improve search quality for themselves and their colleagues. These characteristics naturally lead to considering using user feedback to improve search quality in intranets.

In this paper we explore the use of a particular form of feedback, user annotations, to improve the quality of intranet search. An annotation is a short description of the contents of a web page. In some sense, annotations are a substitute for anchor text.

One way to obtain annotations is to let users explicitly enter annotations for the pages they browse. In our system, users can do so through a browser toolbar. When trying to obtain explicit user feedback, it is important to provide users with clear immediate benefits for taking their time to give the feedback. In our case, the annotation the user has entered shows up in the browser toolbar every time the user visits the page, providing a quick reminder of what a page is about. The annotation will also appear on the search engine results page, if the annotated page is returned as a search result.

While the methods described above provide the user with useful benefits for entering annotations, we have found many users reluctant to provide explicit annotations. We therefore propose another method for obtaining annotations, which automatically extracts them from the search engine query log. The basic idea is to use the queries users submit to the search engine as annotations for pages users click on. However, the naïve approach of assigning a

*This work was done while these authors were at IBM Almaden Research Center.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2006, May 23–26, 2006, Edinburgh, Scotland.
ACM 1-59593-323-9/06/0005.

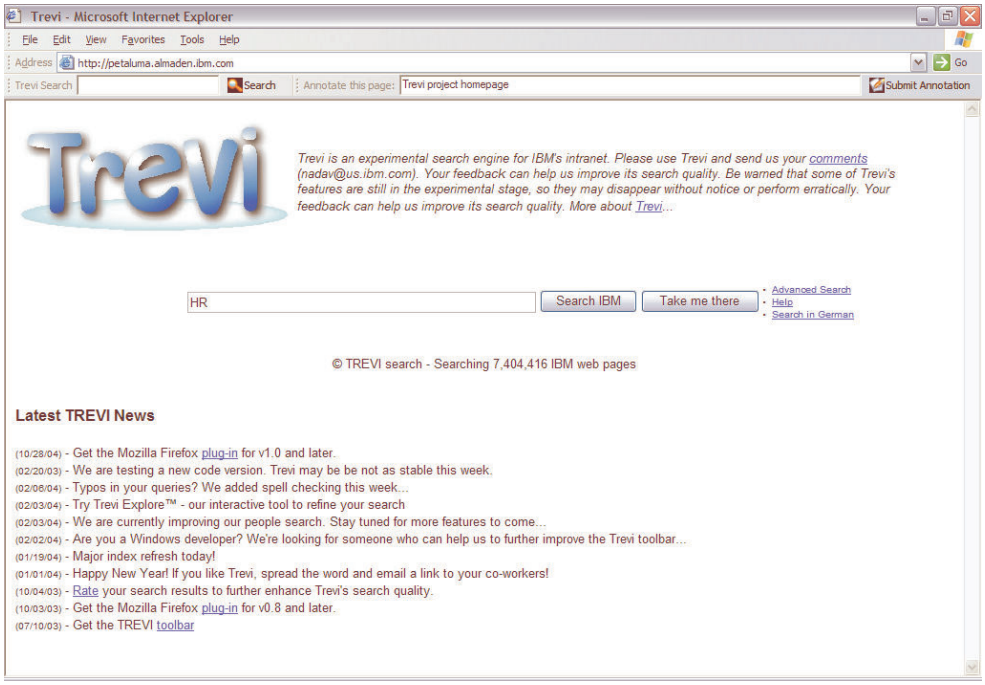

Figure 1: The Trevi Toolbar contains two fields: a search field to search IBM intranet using Trevi, and an annotation field to submit an annotation for the page currently open in the browser.

query as an annotation to every page the user clicks on may assign annotations to irrelevant pages. We experiment with several techniques for deciding which pages to attach an annotation to, making use of the users' click patterns and the ways they reformulate their queries.

The main contributions of this paper include:

- A description of the architecture for collecting annotations and for adding annotations to search indexes.
- Algorithms for generating implicit annotations from query logs.
- Preliminary experimental results on a real dataset from the IBM intranet, consisting of 5.5 million web pages, demonstrating that annotations help to improve search quality.

The rest of the paper is organized as follows. Section 2 briefly reviews basic Web IR concepts and terminology. Section 3 describes in detail our methods for collecting annotations. Section 4 explains how annotations are integrated into the search process. Section 5 presents experimental results. Section 6 discusses related work, and section 7 concludes the paper.

2. BACKGROUND

In a Web IR system retrieval of web pages is often based on the pages' content plus the anchor text associated with them. Anchor text is the text given to links to a particular page in other pages that link to it. Anchor text can be viewed as a short summary of the content of the page authored by a person who created the link. In aggregate, anchor text from all incoming links provides an objective description of the page. Thus, it is not surprising that anchor text has been shown to be extremely helpful in Web IR [1, 14, 15].

Most Web IR systems use inverted indexes as their main data structure for full-text indexing [29]. In this paper, we assume an inverted index structure. The occurrence of a term t within a page p is called a *posting*. The set of postings associated to a term t is stored in a *posting list*. A posting has the form $<$*pageID, payload*$>$, where *pageID* is the ID of the page p and where the payload is used to store arbitrary information about each occurrence of t within p. For example, payload can be used to indicate whether the term came from the title of the page, from the regular text, or from the anchor text associated with the page. Here, we use part of the payload to indicate whether the term t came from content, anchor text, or annotation of the page and to store the offset within the document.

For a given query, a set of candidate answers (pages that match the query words) is selected, and every page is assigned a relevance score. The score for a page usually contains a query-dependent textual component, which is based on the page's similarity to the query, and a query-independent static component, which is based on the *static rank* of the page. In most Web IR systems, the textual component of the score follows an additive scoring model like *tf* $\times$ *idf* for each term, with terms of different types, e.g. title, text, anchor text, weighted differently. Here we adopt a similar model, with annotation terms weighted the same as terms from anchor text. The static component can be based on the connectivity of web pages, as in PageRank [24], or on other factors such as source, length, creation date, etc. In our system the static rank is based on the site count, i.e., the number of different sites containing pages that link to the page under consideration.

3. COLLECTING ANNOTATIONS

This section describes how we collect explicit and implicit annotations from users. Though we describe these procedures in the context of the Trevi search engine for the IBM intranet [17], they

can be implemented with minor modifications on top of any intranet search engine. One assumption our implementation does rely on is the identification of users. On the IBM intranet users are identified by a cookie that contains a unique user identifier. We believe this assumption is valid as similar mechanisms are widely used in other intranets as well.

3.1 Explicit Annotations

The classical approach to collecting explicit user feedback asks the user to indicate relevance of items on the search engine results page, e.g. [28]. A drawback of this approach is that, in many cases, the user needs to actually see the page to be able to provide good feedback, but after they got to the page they are unlikely to go back to the search results page just for the purpose of leaving feedback. In our system users enter annotations through a toolbar attached to the Web browser (Figure 1). Each annotation is entered for the page currently open in the browser. This allows users to submit annotations for any page, not only for the pages they discovered through search. This is a particularly promising advantage of our implementation, as annotating only pages already returned by the search engine creates a "rich get richer" phenomenon, which prevents new high quality pages from becoming highly ranked in the search engine[11]. Finally, since annotations appear in the toolbar every time the user visits the page he or she has annotated, it is easy for the user to modify or delete their annotations.

Currently, annotations in our system are private, in the sense that only the user who entered the annotation can see it displayed in the toolbar or search results. While there are no technical problems preventing us from allowing users to see and modify each other's annotations, we regarded such behavior undesirable and did not implement it.

3.2 Implicit Annotations

To obtain implicit annotations, we use Trevi's query log, which records the queries users submit, and the results they click on. Every log record also contains an associated userID, a cookie automatically assigned to every user logged into the IBM intranet (Figure 2). The basic idea is to treat the query as an annotation for pages relevant to the query. While these annotations are of lower quality than the manually entered ones, a large number of them can be collected without requiring direct user input. We propose several strategies to determine which pages are relevant to the query, i.e., which pages to attach an annotation to, based on clickthrough data associated with the query.

```
LogRecord ::= <Query> | <Click>
Query ::= <Time>\t<QueryString>\t<UserID>
Click ::= <Time>\t<QueryString>\t<URL>\t<UserID>
```

Figure 2: Format of the Trevi log file.

The first strategy is based on the assumption that if the user clicked on a page in the search results, they thought that this page is relevant to the query in some way. For every `Click` record, the strategy produces a (`URL`, `Annotation`) pair, where `Annotation` is the `QueryString`. This strategy is simple to implement, and gives a large number of annotations.

The problem with the first strategy is that, since the user only sees short snippets of page contents on the search results page, it is possible that the page they clicked on ends up not being relevant to the query. In this case, the strategy will attach the annotation to an irrelevant page.

Typically, after clicking on an irrelevant link on a search engine results page, the user goes back to the results page and clicks on another page. Our second strategy accounts for this type of behavior, and is based on the notion of a session. A session is a time-ordered sequence of clicks on search results that the user makes for a given query. We can extract session data from the query log based on `UserID`s. Our second strategy only produces a (`URL`, `Annotation`) pair for a click record which is the last record in a session. `Annotation` is still the `QueryString`.

The two other strategies we propose try to take into account the fact that users often reformulate their original query. The strategies are based on the notion of a query chain [25]. A query chain is a time-ordered sequence of queries, executed over a short period of time. The assumption behind using query chains is that all subsequent queries in the chain are actually refinements of the original query.

The third and the fourth strategies for extracting implicit annotations are similar to the previous two, except that they use query chains instead of individual queries. We extract query chains from the log file based on the time stamps of the log records. The third strategy, similarly to the first one, produces a (`URL`, `Annotation`) pair for every click record, but `Annotation` now is the concatenation of all `QueryString`s from the corresponding query chain. Finally, the fourth strategy produces a (`URL`, `Annotation`) pair for a click record which is the last record in the last session in a query chain, and `Annotation` is, again, the concatenation of `QueryString`s from the corresponding query chain.

Recent work has demonstrated that the naïve strategy of regarding every clicked page as relevant to the query (our first strategy) produces biased results, due to the fact that the users are more likely to click on the higher ranked pages irrespective of their relevance [21]. Our hope is that the session-based strategies will help to eliminate this bias. However, these strategies produce significantly smaller amounts of data comparing to the original strategy. Using query chains helps to increase the amount of data, by increasing the size of the annotation data added to a page.

3.3 The Value of Annotations

Annotations, both explicit and implicit, have the potential to influence retrieval and ranking in many ways. One particular instance in which annotations are helpful is in enriching the language used to describe a concept. Like anchor text, annotations let users use their own words to describe a concept that the annotated page talks about. Users' vocabulary may be radically different in some cases than the one used by authors. This dichotomy in vocabulary is particularly prevalent in intranets, where corporate policy dictate that certain terms be avoided in official documents or web pages. We expect that the similarity between anchor text vocabulary and query vocabulary [14] will also be present in annotations. This is obviously the case for our method of collecting implicit annotations, as those annotations are just old queries.

As an example, the United States Federal Government went through a restructuring lately that changed the name of the agency that is responsible for immigration from "Immigration and Naturalization Service", or INS, to "Unites States Citizenship and Immigration Services", or USCIS[1]. While all formal web pages describing government immigration-related activities no longer mention the words "INS", users might still search by the old, and better known, name of the agency. We can therefore expect that annotations will also use these terms. This is true even of implicit annotations, as many search engines do not force all terms to be present in a search

[1]While this example does not directly applies to most intranets, similar examples are common in many large intranets.

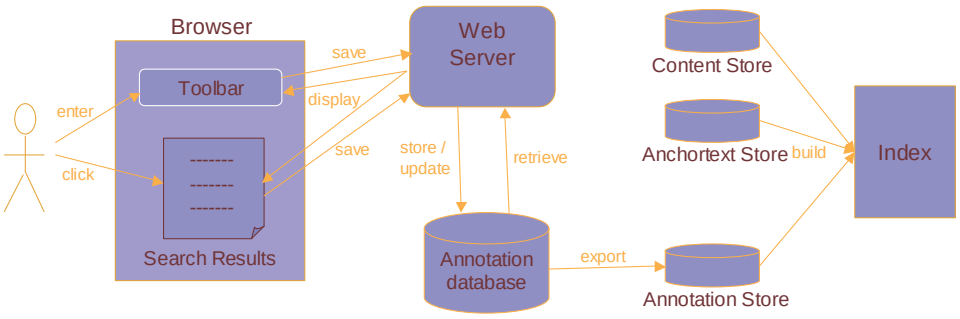

Figure 3: Flow of annotations through the system.

result. Thus, someone searching for "INS H1-B visa policies" may actually find a USCIS page talking about the subject. An implicit annotation of such a page will still be able to add the term "INS" to that page, improving its ranking for users using the older terms in future queries, or in queries where the term "INS" is required to appear.

4. USING ANNOTATIONS IN INTRANET SEARCH

In order for annotations to be integrated into the search process, they must be included in the search engine index. One way to achieve this is to augment the content of pages with annotations collected for these pages [22]. However, this does not take into account the fact that annotations have different semantics. Being concise descriptions of content, annotations should be treated as meta-data, rather than content. From this point of view, they are similar to anchor text [14]. Thus, we decided to use annotations in a similar way to how anchor text is used in our system.

The flow of annotations through the system is illustrated in Figure 3. After submitted by the user, explicit annotations are stored in a database. This database is used to display annotations back to the user in the toolbar and search results. Periodically (currently once a day), annotations are exported into an annotation store – a special format document repository used by our indexing system [17]. The annotation store is combined with the content store and anchor text store to produce a new index. This is done by sequentially scanning these three stores in batch mode and using a disk-based sort merge algorithm for building the index [17]. Once the new index is ready, it is substituted for the old one.

The index build algorithm takes the pages from the stores as input and produces the inverted index, which is basically a collection of posting lists, one list for each token that appears in the corpus. To reduce storage and I/O costs, posting lists are compressed using a simple variable-byte scheme based on computing the deltas between positions.

As the stores are scanned, tokens from each page are streamed into a sort, which is used to create the posting lists. The primary sort key is on token, the secondary sort key is on the pageID, and the tertiary sort key is on offset within a document. By sorting on this compound key, token occurrences are effectively grouped into ordered posting lists.

Since we use an optimized fixed-width radix sort [12, 26], we hash the variable length tokens to produce a 64-bit token hash. Therefore our sort key has the form *(tokenID, pageID, section/offset)*, where tokenID is a 64-bit hash value, pageID is 32 bits, and section/offset within a document is 32 bits. The encoding of the sort key is illustrated in Figure 4. In the posting list data structure the section/offset is stored in the payload for each posting entry.

As shown in Figure 4, we use two bits to denote the section of a document. This is so a given document can be streamed into the sort in different sections. To index anchor text and annotation tokens, the anchor and annotation stores are scanned and streamed into the sort after the content store is scanned. The section bits are used to indicate whether a token is for content, anchor text, or annotation. Consequently, after sorting, the anchor text tokens for a page P follow the content tokens for P, and the annotations tokens follow the anchor text.

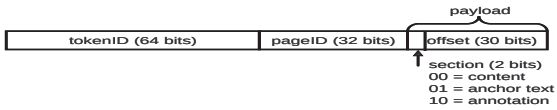

Figure 4: Sort key used to sort content, anchor text, and annotation tokens

Within each page P, tokens are streamed into the sort in the order in which they appear in P, that is, in offset order. Taking advantage of the fact that radix sort is stable, this allows us to use a 96-bit sort key that excludes offset, rather than sorting on the full 128-bit key.

Currently, for retrieval and ranking purposes annotations are treated as if they were anchor text. However, one can imagine a ranking function that would treat annotations and anchor text differently. Experimenting in this direction is one of the areas for our future work.

The process for including implicit annotations is similar, except that the annotation store is built from search engine log files instead of the database.

5. EXPERIMENTAL RESULTS

In this section we present experimental results for search with explicit and implicit annotations. We implemented our approaches on the Trevi search engine for the IBM intranet, currently searching more than 5.5 million pages.

The explicit annotations dataset consists of 67 pages, annotated with a total of 158 annotations by users at IBM Almaden Research Center over a period of two weeks during summer 2005. Out of the 67 annotated pages, 14 were contained in the Trevi index. The implicit annotations dataset consists of annotations extracted from Trevi log files for a period of approximately 3 months. The number

Annotation	Annotated Page
change IBM passwords	Page about changing various passwords in IBM intranet
stockholder account access	Login page for IBM stock holders
download page for Cloudscape and Derby	Page with a link to Derby download
ESPP home	Details on Employee Stock Purchase Plan
EAMT home	Enterprise Asset Management homepage
PMR site	Problem Management Record homepage
coolest page ever	Homepage of an IBM employee
most hard-working intern	an intern's personal information page
good mentor	an employee's personal information page

Table 1: Examples of Annotations

of annotated pages is 12433 for the first and the third strategies, 8563 for the second strategy, and 4126 for the fourth strategy.

Given the small number of explicit annotations we were able to collect, relative to the size of the dataset (5.5 million pages), the results presented below can only be viewed as very preliminary. Nevertheless, we observed several interesting characteristics of annotations, which highlight their usefulness in enterprise search.

5.1 Types of Annotations

Table 1 shows examples of explicit annotations. The most typical type of annotations were concise descriptions of what a page is about, such as the first set of annotations in Table 1. While these annotations do not usually introduce new words into the description of a page, in many cases they are still able to increase the rank of the page for relevant queries. Consider, for example, the annotation "download page for Cloudscape and Derby", attached to a page with the link to Derby download. Cloudscape is a popular IBM Java RDBMS, which was recently renamed Derby and released under an open source license. Cloudscape has been integrated in many IBM products, which resulted in frequent co-occurence of the "download" and "Cloudscape" on the same page. The renaming also led to replacement of Cloudscape with Derby on some of the pages. As a result, neither of the queries "download Cloudscape" or "download Derby" return the correct page with a high rank. However, with the above annotation added to the index, the page is ranked much higher for both queries, because the keywords occur close to each other in the annotation, and Trevi's ranking function takes this into account.

Another common type of annotations were abbreviations (the second set in Table 1). At IBM, like at any other big company, everything is given a formal sounding, long name. Thus, employees often come up with abbreviations for such names. These abbreviations, widely used in spoken language, are not always mentioned in the content of the web pages describing the corresponding entities. We observed that entering an abbreviation as an annotation for a page describing a program or a service was a very common type of annotations. Such annotations are extremely useful for search, since they augment the page with a keyword that people frequently use to refer to the page, but which is not mentioned in its content.

The third type of annotations reflect an opinion of a person regarding the content of the web page. The last set of annotations in Table 1 gives examples of such annotations. While a few annotations of this type that we have in our dataset do not convey any serious meaning, such annotations have a potential to collect opinions of people regarding services or other people, which employees do not have an opportunity to express otherwise. One can imagine, for example, that a query like "best physical training class at Almaden" will indeed return as the first hit a page describing the most popular physical training program offered to IBM Almaden employees, because many people have annotated this page with the keyword "best".

5.2 Impact on Search Quality

To evaluate the impact of annotations on search quality we generated 158 test queries by taking explicit annotations to be the queries, and the annotated pages to be the correct answers. We used performance of the search engine without annotations as a baseline for our experiments. Table 2 shows the performance of explicit and implicit annotations in terms of the percentage of queries for which the correct answer was returned in the top 10 results.

Baseline	EA	IA 1	IA 2	IA 3	IA 4
8.9%	13.9%	8.9%	8.9%	9.5%	9.5%

Table 2: Summary of the results measured by the percentage of queries for which the correct answer was returned in the top 10. EA = Explicit Annotations, IA = Implicit Annotations.

Adding explicit annotations to the index results in statistically significant at 95% level improvement over the baseline. This is expected, given the nice properties of annotations mentioned above. However, even with explicit annotations the results are rather low. One reason for that is that many of the annotations were attached to dynamically generated pages, which are not indexed by the search engine. As mentioned above, only 14 pages out of 67 annotated pages were actually in the index.

Implicit annotations, on the other hand, did not show any significant improvement over the baseline. There was also little difference among the different implicit annotation strategies. A closer investigation showed that there was little overlap between the pages that received implicit annotations, and the ones that were annotated explicitly. We suspect that, since the users knew that the primary goal of annotations is to improve search quality, they mostly tried to enter explicit annotations for pages they could not find using Trevi. We conclude that a different experimentation approach is needed to evaluate the true value of implicit annotations, and the differences among the four annotation strategies.

6. RELATED WORK

There are three categories of work related to this paper: enterprise search, page annotations on the Web, and improving search quality using user feedback. We discuss each of these categories in subsequent sections.

6.1 Enterprise Search

While there are many companies that offer enterprise search solutions [2, 3, 5, 7], there is surprisingly little work in this area in the research community.

In a recent paper [18], David Hawking enumerates the challenges in enterprise search. Since enterprise search engines deal with a a variety of information sources, such as databases, content management systems, e-mail, etc., one of the key challenges is finding effective ways for ranking results from such heterogeneous collections. While it is definitely possible to modify an off-the-shelf enterprise search engine to use the ranking function suitable for a particular company, finding such ranking function is a challenging and time consuming task, specific to the company the search engine is being deployed in. Our method of using user feedback is one way to tune the ranking to a particular company, which does not require modifying the search engine and does not depend on the specifics of the company's intranet.

The problem of enterprise search is also addressed in [15]. The authors use rank aggregation to evaluate the effects of different factors on ranking of search results. One particular finding they report is that anchor text was by far the most important ranking factor. Since the semantics of our annotations is very similar to that of anchor text, it is not surprising that they lead to significant improvement in search quality.

While commercial products are likely to use some forms of explicit or implicit user feedback, to our knowledge this paper is the first research effort to study impact of different forms of feedback on the quality of enterprise search.

6.2 Annotations on the Web

The idea of capturing and sharing notes people make on documents in an automatic fashion can be traced back to Memex system [10]. It is thus not surprising that annotations on the Web have long been used to provide users with a way to store a description or an opinion of a web page for the user's future reference, to facilitate information filtering and categorization, and to support collaboration among other tasks [6, 13, 27]. Annotating capabilities were even included in early versions of the Mosaic browser [4]. In [23] a browser toolbar was used to collect and display annotations, which is similar to our approach to working with explicit annotations.

Our system is different from the above systems in that we integrate annotations into the search process. Recently, support for using annotations in search was implemented in the Yahoo! MyWeb 2.0 system [8]. In this system annotations are used in a web community environment, where users can choose to share annotations, as well as other data with their friends, allowing them to search through that data. While the idea of [8] is similar to ours, the way annotations are integrated into the search process seems to be different. Since [8] is a commercial system, lack of knowledge of their algorithms does not allow us to perform a thorough comparison of their system to ours.

Another difference of our system is that, in addition to using explicit annotations, we also derive annotations implicitly from users' behavior. It remains an area for future research to see whether implicitly derived annotations can be used instead of explicit annotations in the tasks mentioned above.

6.3 User Feedback in Web Search

User feedback used in web search can be classified into two categories according to how the feedback is collected. Explicit feedback approach requires direct participation from the user to indicate the relevance of search results, or provide comments on the results [9, 20]. Implicit feedback approaches try to infer feedback data from users' behavior, liberating them from the burden of providing feedback explicitly [19, 22].

While the explicit feedback approach produces higher quality data, it is difficult to collect large amounts of it. Most of the research, therefore, concentrated on using implicit feedback, typically clickthrough data obtained from search engine log files [19, 22]. The idea is to interpret user clicks on pages form the search results as relevance judgments. Two kinds of judgments that can be inferred from clickthrough data are absolute and relative judgments.

Absolute judgments interpret a click as an indication of relevance of the page to the query [9, 22]. Relative judgments, in its simplest form, consider pairs of pages in search results, and assume that if the user clicked on one page in the pair, and did not click on the other, then the former page is more relevant to the query than the latter one [19]. In [25] the notion of query chains was introduced, and used to infer relative relevance judgments. In our work, we adopt their definition of query chains, but apply it to infer absolute feedback.

Recent work on evaluating various strategies of interpreting clickthrough data found that interpreting clicks as absolute relevance judgments (assuming every page the user clicked on is relevant to the query) produces results biased towards higher ranked pages, and is influenced by the overall quality of the search engine [21]. At the same time, [22] demonstrated an improvement in search results using such judgments. In our preliminary experiments we observed a small improvement in search quality from using implicit absolute feedback. We did not observe a significant difference among different strategies for the feedback. More experiments are needed to see whether session and query-chain based strategies help to eliminate the bias and produce better results.

7. CONCLUSION

In this paper we showed how user annotations can be used to improve the search quality in intranets. In addition to collecting annotations directly from the users through a browser toolbar, we proposed several strategies for obtaining implicit annotations from search engine query logs. We showed how annotations can be integrated into a search engine index, and used during the search. Preliminary experiments on the IBM intranet demonstrated that annotations can help to improve the search quality.

There are several directions in which this work can be extended. First, our method for extracting query chains is based purely on time between queries. Other factors, such as similarity between queries and the results could potentially lead to more accurate query chain extraction. Second, while using annotations for ranking in the same way as anchor text produces good results, more research is needed to see whether further improvement is possible if annotations are used for ranking in a different way. Third, it is possible that our implicit annotations strategies attach a query as an annotation to an irrelevant page. An approach to detect such cases may rely on the use of existing explicit and implicit annotations for this page. Forth, with time the amount of annotation data for a page may actually exceed the amount of page content and anchor text. Thus, there is a need to balance the amounts of content, anchor text, explicit annotation, and implicit annotation data we want to use for ranking. Finally, with the page's content changing over time, some annotations may become obsolete. One way to deal with this problem is to introduce the concept of age into the annotation framework, with older annotations having less influence on the ranking.

8. ACKNOWLEDGEMENTS

The authors would like to thank reviewers of this article for their insightful comments and suggestions.

9. REFERENCES

[1] Anchor text optimization. www.seo-gold.com/tutorial/anchor-text-optimization.html.

[2] Google enterprise solutions. http://www.google.com/enterprise/.

[3] IBM OmniFind solution for enterprise search. http://www-306.ibm.com/software/data/integration/db2ii/editions_womnifind.html.

[4] NCSA mosaic: Annotations overview. http://archive.ncsa.uiuc.edu/SDG/Software/XMosaic/Annotations/overview.html.

[5] Panoptic enterprise search engine. http://www.panopticsearch.com.

[6] StumbleUpon. http://www.stumbleupon.com.

[7] Verity enterprise search solution. http://www.verity.com/products/search/enterprise_web_search/index.html.

[8] Yahoo! MyWeb 2.0 BETA. http://myweb2.search.yahoo.com/.

[9] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. pages 107–117, 1998.

[10] Vannevar Bush. As we may think. In *The Atlantic Monthly*, July 1945.

[11] Junghoo Cho and Sourashis Roy. Impact of search engines on page popularity. In *Proc. 13th World Wide Web Conference*, pages 20–29, May 2004.

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 2003.

[13] Laurent Denoue and Laurence Vignollet. New ways of using web annotations. In *Proc. 9th World Wide Web Conference*, Amsterdam, 2000.

[14] Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *Proc. 26th ACM Conference on Research and Development in Information Retrieval*, pages 459–460, 2003.

[15] Ronald Fagin, Ravi Kumar, Kevin S. McCurley, Jasmine Novak, D. Sivakumar, John A. Tomlin, and David P. Williamson. Searching the workplace web. In *Proc. 12th World Wide Web Conference*, Budapest, Hungary, 2003.

[16] Susan Feldman and Chris Sherman. The high cost of not finding information. In *IDC Technical Report 29127*, 2003.

[17] Marcus Fontoura, Eugene J. Shekita, Jason Y. Zien, Sridhar Rajagopalan, and Andreas Neumann. High performance index build algorithms for intranet search engines. In *VLDB*, pages 1158–1169, 2004.

[18] David Hawking. Challenges in enterprise search. In *Fifteenth Australian Database Conference*, Dunedin, NZ, 2004.

[19] Thorsten Joachims. Optimizing search engines using clickthrough data. Alberta, Canada, 2002.

[20] Thorsten Joachims, Dayne Freitag, and Tom Mitchell. Webwatcher: A tour guide for the world wide web. In *Proc. International Joint Conference on Artificial Intelligence*, Nagoya, Japan, 1997.

[21] Thorsten Joachims, Laura Granka, Bing Pang, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proc. 28th ACM Conference on Research and Development in Information Retrieval*, Salvador, Brazil, 2005.

[22] Charles Kemp and Kotagiri Ramamohanarao. Long-time learning for web search engines. In *Proc. 6th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Helsinki, Finland, 2002.

[23] Hannes Marais and Krishna Bharat. Supporting cooperative and personal surfing with a desktop assistant. In *10th annual ACM symposium on User Interface Software and Technology*, Banff, Alberta, Canada, 1997.

[24] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998. Paper SIDL-WP-1999-0120 (version of 11/11/1999).

[25] Filip Radlinski and Thorsten Joachims. Query chains: Learning to rank from implicit feedback. In *Proc. 11th ACM Conference on Knowledge Discovery and Data Mining*, Chicago, IL, USA, 2005.

[26] Robert Sedgewick. *Algorithms in C++*. Addison-Wesley Publishing Company, Boston, MA, 1998.

[27] Venu Vasudevan and Mark Palmer. On web annotations: Promises and pitfalls of current web infrastructure. In *32nd Hawaii International Conference on Systems Sciences*, Maui, Hawaii, 1999.

[28] Vishwa Vinay, Ken Wood, Natasa Milic-Frayling, and Ingemar J. Cox. Comparing relevance feedback algorithms for web search. In *Proc. 14th World Wide Web Conference*, Chiba, Japan, 2005.

[29] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes*. Morgan Kaufmann, 1999.