Yarrow: A Real-Time Client Side Meta-Search Learner

Zhixiang Chen and **Xiannong Meng**
Department of Computer Science, University of Texas – Pan American
1201 West University Drive, Edinburg, TX 78539–2999, USA
chen@cs.panam.edu, meng@cs.panam.edu

Abstract

In this paper we report our research on building Yarrow - an intelligent web meta-search engine. The predominant feature of Yarrow is that in contrast to the lack of adaptive learning features in existing meta-search engines, Yarrow is equipped with a practically efficient on-line learning algorithm so that it is capable of helping the user to search for the desired documents with as little feedback as possible. Currently, Yarrow can query eight of the most popular search engines and is able to perform document parsing and indexing, and learning in real-time on client side. Its architecture and performance are also discussed.

1. Introduction

As the world wide web evolves and grows so rapidly, web search, an interface between the human users and the vast information gold mine of the web, is becoming a necessary part of people's daily life. Designing and implementing practically effective web search tools is a challenging task. It calls for innovative methods and strategies from many fields including machine learning. One promising direction is the building of intelligent meta-search engines on top of the existing general-purpose search engines that have been proved in practice to be very effective and useful in helping the users to search for their desired information. People behind those existing popular search engines may not like meta-search process because of their own commercial interests. However, meta-search process is an unavoidable trend, because it represents a higher level in the information *food chain*. There are some interesting discussions in (Selberg & Etzioni 1995; 1997) about the impact of meta-search engines on general-purpose search engines.

There have been a number of popular and successful meta-search engines such as Dogpile [2], Inference Find [4], and MetaCrawler [10]. The common properties of those meta-search engines are as follows. Each [h] of the meta-search engines has a single unified interface for the user to enter her query. When the user starts a search process by entering a query, it sends the query to one or to a collection of the general-purpose search engines in parallel to retrieve the top matched relevant documents. Some of the meta-search engines (for example, MetaCrawler [10]) collate, filter, and sort the retrieved documents into a single list. Some (for example, Inference Find [4]) cluster the retrieved documents from the general-purpose search engines and have customizable features that allow the user to customize a list of parameters to enhance the filtering and ranking performance. But as far as the authors understand, none of those existing meta-search engines support adaptive learning from the user's feedback.

Meta-search engines may be classified into two categories: ***shallow meta-search engines*** and ***deep meta-search engines.*** A shallow meta-search engine simply echoes the search result of one or several general-purpose search engines. There may be some collating, filtering, or sorting processes, but such efforts are very limited. A deep meta-search engine will use the search results of the general-purpose search engines as its starting search space, from which it will adaptively learn from the user's feedback to boost and enhance the search performance and the relevance accuracy of the general-purpose search engines. It may use clustering, filtering, and other methods to help its adaptive learning process. From the engineering point of view, a meta-search engine is usually light-weighted and does not require a large database, nor a large amount of memory. It should and is able to emphasize the intelligent processing of the search results returned by general-purpose search engines. Recent research on web communities (Kleinberg 1999; Gibson, Kleinberg, & Raghavan 1998; Chakrabarti *et al.* 1998) has used a short list of hits returned by a search engine as a starting set for further expansion. There have been great efforts on applying machine learning on web search related applications, for example, scientific article locating and user profiling (Bollacker, Lawrence, & Giles 1998; 1999; Lawrence, Bollacker, & Giles 1999), and focused crawling (Rennie & McCallum 1999).

Yarrow is our first step toward building an intelligent deep meta-search engine. Currently, Yarrow can query eight of the most popular general-purpose search engines and is able to perform document parsing and indexing, and learning in real-time on client side. The predominant feature of Yarrow is that in contrast to the lack of adaptive learning features in existing meta-search engines, Yarrow is equipped with an on-line learning algorithm TW2 so that it is capable of helping the user to search for the desired documents with as little feedback as possible. We designed in (Chen 2000b) the learning algorithm TW2, a tailored version of Winnow2 (Littlestone 1988) in the case of web search. When used to learn a disjunction of at most k relevant attributes, TW2 has surprisingly small mistake bounds that are independent of the dimensionality of the indexing attributes. TW2 has been successfully used as part of the learning components in our other projects (Chen *et al.* 2000b; 2000a).

2. Web Search vs. On-line Learning

As we have investigated in (Chen 2000b; Chen, Meng, & Fowler 1999; Chen *et al.* 2000b), intelligent web search can be modeled as an adaptive learning process such as on-line learning (Angluin 1987; Littlestone 1988), where the search engine acts as a learner and the user as a teacher. The user sends a query to the engine, and the engine uses the query to search the index database and returns a list of urls that are ranked according to a ranking function. Then the user provides the engine relevance feedback, and the engine uses the feedback to improve its next search and returns a refined list of urls. The learning (or search) process ends when the engine finds the desired documents for the user. Conceptually a query entered by the user can be understood as the logical expression of the collection of the documents wanted by the user. A list of urls returned by the engine can be interpreted as an approximation to the collection of the desired documents.

One must realize that in the case of web search the user in general has no patience to try more than a couple of dozens of interactions for a search process, nor to wait for a dozen of minutes for the engine to return the search results. One must also realize that the dimensionality of the web document indexing attributes is extremely high (usually, a huge vocabulary of keywords is used as indexing attributes). In such reality of web search, few well-established machine learning algorithms are applicable to web search.

The Rocchio's similarity-based relevance feedback, the most popular query formation method in information retrieval (J.J. Rocchio 1971; Salton 1989; Baeza-Yates & Riberiro-Neto 1999), is in essence an adaptive learning process from examples. We proved in (Chen & Zhu 2000) that for any of the four typical similarity measurements (inner product, cosine coefficient, dice coefficient, and Jaccard coefficient) (Salton 1989), the learning algorithm from relevance feedback has a linear lower bound in the dimensionality of Boolean vector space. In the discretized vector space $\{0, \ldots, M-1\}^n$, the lower bound of the relevance feeback algorithm for any of the above four similarity measurements is $\Omega(Mn)$ (Chen 2000a). Our lower bounds hold for arbitrary zero-one initial query vectors, and for arbitrary threshold and updating coefficients used at each step of the learning process. Our lower bounds imply that in contrast to various successful applications, at least in theory the Rocchio's similarity-based relevance feedback algorithm is not a good choice for web search.

We use the vector space model (Salton, Wong, & Yang 1975; Salton 1989; Baeza-Yates & Riberiro-Neto 1999) to represent documents. We introduce TW2, a tailored version of Winnow2, to exploit the particular nature of web search. In contrast to the fact that Winnow2 sets all initial weights to 1, TW2 sets all initial weights to 0 and has a different promotion process accordingly. The rationale behind setting all the initial weights to 0 is not as simple as it looks. The motivation is to focus attention on the propagation of the influence of the relevant documents, and use irrelevant documents to adjust the focused search space. Moreover, this approach is realistic because existing effective document ranking mechanisms can be coupled with the learning process as discussed in next section.

Algorithm TW2 (The tailored Winnow2). *TW2 maintains non-negative real-valued weights $w_1, \ldots, w_n$ for attributes $att_1, \ldots, att_n$, respectively. It also maintains a real threshold θ. Initially, all weights have value 0. Let $\alpha > 1$ be the promotion and demotion factor. TW2 classifies documents whose vectors $x = (x_1, \ldots, x_n)$ satisfy $\sum_{i=1}^{n} w_i x_i > \theta$ as relevant, and all others as irrelevant. If the user provides a document that contradicts to the classification of TW2, then we say that TW2 makes a mistake. Let $w_{i,b}$ and $w_{i,a}$ denote the weight w_i before the current update and after, respectively. When the user responds with a document which contradicts to the current classification, TW2 updates the weights in the following two ways:*

- **Promotion***: For a document judged by the user as relevant with vector $x = (x_1, \ldots, x_n)$, for $i = 1, \ldots, n$, set*

$$w_{i,a} = \begin{cases} w_{i,b}, & if\ x_i = 0, \\ \alpha, & if\ x_i = 1\ and\ w_{i,b} = 0, \\ \alpha w_{i,b}, & if\ x_i = 1\ and\ w_{i,b} \neq 0. \end{cases}$$

- **Demotion***: For a document judged by the user as irrelevant with vector $x = (x_1, \ldots, x_n)$, for $i = 1, \ldots, n$, set $w_{i,a} = \frac{w_{i,b}}{\alpha}$.*

Let A denote the total number of distinct indexing attributes of all the relevant documents that are judged by the user during the learning process. Four mistake bounds are obtained for TW2 and their formal proofs

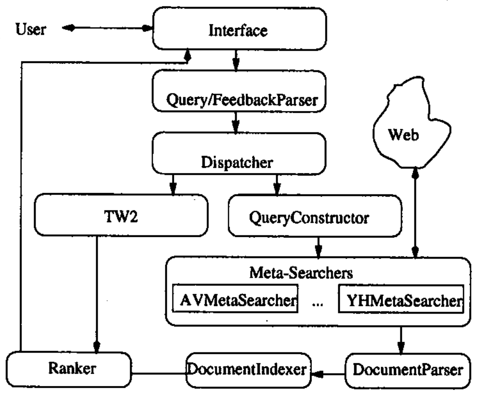


Figure 1: Architecture of WebSail

can be found in (Chen 2000b; Chen *et al.* 2000b). For example, to learn a collection of documents represented by a disjunction of at most k relevant attributes over the n-dimensional boolean space, TW2 makes at most $\frac{\alpha^2 A}{(\alpha-1)\theta} + (\alpha+1)k\ln_\alpha\theta - \alpha$ mistakes.

3. The Yarrow

3.1. The Architecture of Yarrow and How It Works

Yarrow is a multi-threaded program coded in C++. Its architecture is shown in Figure 1. It runs on an Ultra 1 Sun Workstation and does not require a large database or a large amount of memory. For each search process it creates a thread and destroys the thread when the search process ends. Because of its light-weighted size it can be easily converted or ported to run in different environments or platforms. The QueryConstructor, MetaSearchers, DocumentParser, and DocumentIndexer are designed in such a way that they can be scaled to meta-search other popular general-purpose search engines, and can be expanded with new features added to it.

The predominant feature of Yarrow, compared with existing meta-search engines, is the fact that it learns from the user's feedback in real-time on the client side. The learning algorithm TW2 used in Yarrow has a proved surprisingly small mistake bound (Chen 2000b; Chen *et al.* 2000b). Yarrow may be well used as a plug-in component for web browsers on the client side.

Yarrow has an interface as shown in Figure 2. Using this interface, the user can enter her query, specify the number of urls she wants to be returned, and select one of the eight popular general-purpose search engines to perform the initial search. Having entered her query information, she then clicks the search button to start Yarrow. Once started, Yarrow invokes its Query/Feedback Parser to parse the query information or the feedback information out from the interface. Then, the Dispatcher of Yarrow decides whether the current task is an initial search process or a learning process. If it is an initial search process, then the Dispatcher calls the QueryConstructor to formulate the query to fit the specific format of the target search engine, and send the formulated query and the number of urls wanted to the related MetaSearcher to get a list of the most relevant documents from the general-purpose search engine. After this, Yarrow calls its DocumentParser and DocumentIndexer to parse the received documents, to collate them and to index them with at most 64 indexing attributes. The set of the indexing attributes are automatically extracted from the retrieved documents. The attribute-vector representations for all the received documents are also constructed at this point. Yarrow finally presents the top R and the bottom R of the collated list of the retrieved documents to the user for her to judge the relevance of the documents. Usually, Yarrow sets R to 10. The format of presenting the top R and the bottom R documents is shown in Figure 3. In this case, each document url is proceeded by two radio buttons for the user to indicate whether the document is relevant or not[1]. The urls are clickable for viewing the actual documents so that the user can make her judgment more accurately. After the user clicks a few radio buttons for selection of relevant and irrelevant documents, she can click the FeedBack button to submit the feedback to Yarrow, or click the ShowAll button to view all the document urls, or enter a new query to start a new search process.

If the current task is a learning process from the user's relevance feedback, The Dispatcher sends the relevance feedback information to the learning algo-

[1]The search process shown in Figures 3 and 4 was performed on March 1, 2000. The query word is *"UTPA"* and the desired web documents are those related to "the University of Texas - Pan American". The selected general-purpose search engine is Northern Light. After 2 interactions and 5 relevant and irrelevant documents judged by the user as feedback, all the UTPA related web documents among the initial 50 matched documents were moved to the top 10 positions.

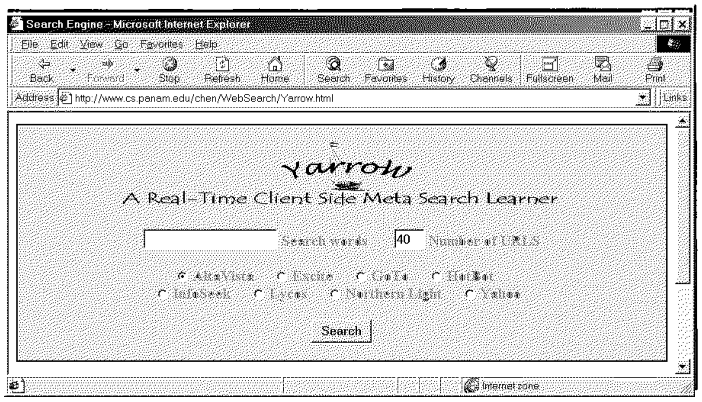

Search Engine - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://www.cs.panam.edu/chen/WebSearch/Yarrow.html
Links
Yarrow
A Real-Time Client Side Meta Search Learner
Search words
40
Number of URLS
AltaVista Excite GoTo HotBot
InfoSeek Lycos Northern Light Yahoo
Search
Internet zone

Figure 2: Interface of Yarrow

rithm TW2. TW2 uses the relevant and irrelevant documents judged by the user as positive and negative examples to promote and demote the weight vector, respectively. TW2 also does individual document promotion or demotion for those judged documents. Once TW2 finishes its updating process, Yarrow then calls its Ranker to re-rank all the documents and later presents the top R and the bottom R to the user for the next step of learning.

3.2. Document Ranking and Equivalence Query Simulation

Since in reality the user cannot be modeled as a teacher as on-line learning does (Angluin 1987; Littlestone 1988), the learning algorithm TW2 must be used with help of document ranking and equivalence query simulation.

Let $w = (w_1, \ldots, w_n)$ be the weight vector of TW2. Let g be a ranking function independent of TW2. We define the ranking function f for TW2 as follows: For any web document d with vector $x_d = (x_1, \ldots, x_n)$,

$$f(d) = \gamma_d[g(d) + \beta_d] + \sum_{i=1}^{n} w_i x_i.$$

g remains constant for each document d during the learning process of TW2. Various strategies can be used to define g, for example, PageRank (Brin & Page 1998). The two additional tuning parameters are used to do individual document promotions or demotions of the documents that have been judged by the user as feedback during the learning process of TW2. The motivation for individual document promotions or demotions is that when the status of a document is clear as judged by the user, it should be placed closer to the top than the rank supported by the weighted sum of TW2 if it is relevant, or placed closer to the bottom otherwise. Initially, let $\beta_d \geq 0$ and $\gamma_d = 1$. γ_d and β_d can be updated in the similar fashion as w_i is updated by TW2.

Yarrow uses the ranking function f to rank the documents classified by TW2 and returns the top R relevant documents and the bottom R irrelevant to the user. The top R and the bottom R documents together represent an approximation to the hypothesis made by TW2. The user can examine these two short lists. If she feels satisfactory with the result then she can say "yes" to end the search process. If she finds that some documents in the lists are misclassified, then she can indicate those to TW2 as relevance feedback. Because normally the user is only interested in the top 10 to 50 ranked documents, R can be tuned between 10 and 50. An alternative way for selecting the bottom R documents is to randomly select R documents among those classified as irrelevant by the learning algorithm.

3.3. The Performance

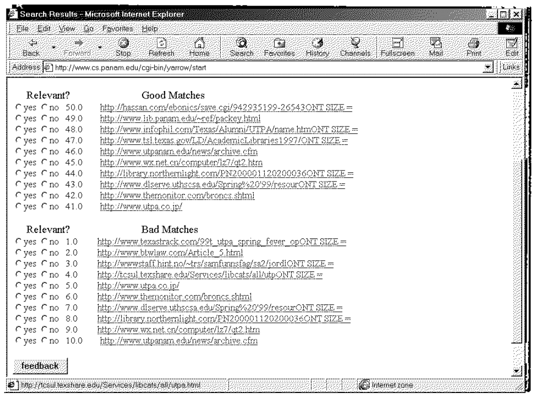
The actual performance of Yarrow is very impressive and promising. We have made Yarrow public. Interested readers can access it via the url given at the end of the paper and can check its performance by themselves. Currently, we have been conducting formal performance evaluations and will report the statistics in the full version of this paper soon. What we can say at this point is that for each search process, very satisfactory result can be achieved with about 5 interactions and about 17 documents judged as relevance feedback. For document indexing, in average a collection of about 700 attributes are automatically extracted and each retrieved document is indexed with at most 64 attributes from the collection.

4. Yarrow and the Popular Meta-search Engines

We now compare Yarrow with the following three of the most popular and successful meta-search engines, Dogpile [2], Inference Find [4], and MetaCrawler [10][2].

Dogpile [2]. It queries the following general-purpose search engines or services: About.com, AltaVista, Deja News, Direct Hit, Dogpile Open Direc-

[2]We collected the features of the three meta-search engines on February 29, 2000.


Search Results - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print Edit
Address
Links
feedback
Internet zone

http://www.cs.panam.edu/cgi-bin/yarrow/start

Relevant?		Good Matches
yes no	50.0	http://hassan.com/ebonics/save.cgi/942935199-26543ONT SIZE =
yes no	49.0	http://www.lib.panam.edu/~ref/packey.html
yes no	48.0	http://www.infophil.com/Texas/Alumni/UTPA/name.htmONT SIZE =
yes no	47.0	http://www.tsl.texas.gov/LD/AcademicLibraries1997/ONT SIZE =
yes no	46.0	http://www.utpanam.edu/news/archive.cfm
yes no	45.0	http://www.wx.net.cn/computer/lz7/qt2.htm
yes no	44.0	http://library.northernlight.com/PN20000112020036ONT SIZE =
yes no	43.0	http://www.dlserve.uthscsa.edu/Spring%20'99/resourONT SIZE =
yes no	42.0	http://www.themonitor.com/broncs.shtml
yes no	41.0	http://www.utpa.co.jp/

Relevant?		Bad Matches
yes no	1.0	http://www.texastrack.com/99t_utpa_spring_fever_opONT SIZE =
yes no	2.0	http://www.btwlaw.com/Article_5.html
yes no	3.0	http://wwwstaff.hint.no/~trs/samfunnsfag/sa2/jordlONT SIZE =
yes no	4.0	http://tcsul.texshare.edu/Services/libcats/all/utpONT SIZE =
yes no	5.0	http://www.utpa.co.jp/
yes no	6.0	http://www.themonitor.com/broncs.shtml
yes no	7.0	http://www.dlserve.uthscsa.edu/Spring%20'99/resourONT SIZE =
yes no	8.0	http://library.northernlight.com/PN20000112020036ONT SIZE =
yes no	9.0	http://www.wx.net.cn/computer/lz7/qt2.htm
yes no	10.0	http://www.utpanam.edu/news/archive.cfm

http://tcsul.texshare.edu/Services/libcats/all/utpa.html

Figure 3: Initial Query Result for "UTPA"

tory, Dogpile Web Catalog, Google, GoTo.com, Infoseek, Infoseek News Wires, Looksmart, Lycos, Real Names, Thurderstone, Yahoo!, and Yahoo News Headlines. It performs parallel queries to those search services, but also allows the user to customize her search to one specific engine. It does not sort, collate, nor cluster the lists of hits returned by each search service. It has no adaptive learning features from the user's feedback.

Inference Find [4]. It queries the following general-purpose search services: AltaVista, Excite, Infoseek, Lycos, WebCrawler, and Yahoo!. It performs parallel queries to those search services, but does not allow the user to customize her search. It sorts and clusters the lists of hits returned by the search services. It has no adaptive learning features from the user's feedback.

MetaCrawler [10]. It queries the following general-purpose search services: About.com, AltaVista, Excite, Infoseek, Looksmart, Lycos, Thunderstone, WebCrawler, and Yahoo! It performs parallel queries to those search services, and also allow the user to customize her search to one specific engine. It collates and sorts the lists of hits returned by the search services. It normalizes the confidence scores used by each of the service, sums the scores and rank them from 1 to 1000. It has no adaptive learning features from the user's feedback.

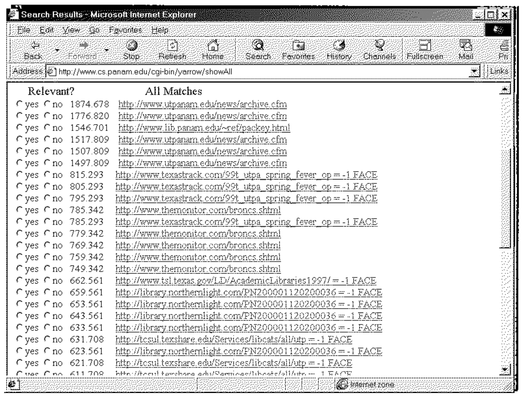
Yarrow [14]. It queries the following general-purpose search services: AltaVista, Excite, GoTo, HotBot, Infoseek, Lycos, Northern Light, and Yahoo!. It does not support parallel queries at this point, but allows the user to specify her favorite search service. Its most important feature is its real-time adaptive learning ability from the user's feedback.

5. Concluding Remarks

The authors believe that deep meta-search should be the ultimate goal of meta-search, that is, a meta-search engine or service should use the retrieved lists of hits from one or a collection of general-purpose search engines or services as its starting search space, from which it should adaptively learns from the user's feedback to boost and enhance the search performance and accuracy of the general-purpose search services. From the engineering point of view, deep meta-search is possible, because a meta-search engine is usually light-weighted and does not require a large database nor a large amount of memory. As the first step to achieve our goal of deep meta-search, we implemented Yarrow during the winter break of the 1999-2000 academic year. Yarrow is powered by an efficient learning algorithm and is also equipped with functions of document parsing and indexing. It adaptively learns from the user's feedback to search for the desired documents. Yarrow is still in its initial stage and needs to be improved and enhanced in many aspects. For example, we need to improve its thread management. The current version is easy to crash when too many threads competing for the limited memory resources.

URL References:

[1] AltaVista: www.altavista.com.
[2] Dogpile: www.dogpile.com.
[3] Excite: www.excite.com.
[4] Inference Find: www.infind.com.
[5] Infoseek: www.infoseek.com.
[6] Google: www.google.com.
[7] GoTo: www.goto.com.
[8] HotBot: www.hotbot.com.
[9] Lycos: www.lycos.com.
[10] MetaCrawler: www.metacrawler.com.
[11] Northern Light: www.northernlight.com.
[12] WebSail:
www.cs.panam.edu/chen/WebSearch/WebSail.html.
[13] Yahoo!: www.yahoo.com.
[14] Yarrow:
www.cs.panam.edu/chen/WebSearch/Yarrow.html.
[15] Features:
www.cs.panam.edu/chen/WebSearch/Features.html.


Search Results - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Pri
Address http://www.cs.panam.edu/cgi-bin/yarrow/showAll
Links
Internet zone

```
   Relevant?              All Matches
 ○ yes ○ no  1874.678  http://www.utpanam.edu/news/archive.cfm
 ○ yes ○ no  1776.820  http://www.utpanam.edu/news/archive.cfm
 ○ yes ○ no  1546.701  http://www.lib.panam.edu/~ref/packey.html
 ○ yes ○ no  1517.809  http://www.utpanam.edu/news/archive.cfm
 ○ yes ○ no  1507.809  http://www.utpanam.edu/news/archive.cfm
 ○ yes ○ no  1497.809  http://www.utpanam.edu/news/archive.cfm
 ○ yes ○ no  815.293   http://www.texastrack.com/99t_utpa_spring_fever_op = -1 FACE
 ○ yes ○ no  805.293   http://www.texastrack.com/99t_utpa_spring_fever_op = -1 FACE
 ○ yes ○ no  795.293   http://www.texastrack.com/99t_utpa_spring_fever_op = -1 FACE
 ○ yes ○ no  785.342   http://www.themonitor.com/broncs.shtml
 ○ yes ○ no  785.293   http://www.texastrack.com/99t_utpa_spring_fever_op = -1 FACE
 ○ yes ○ no  779.342   http://www.themonitor.com/broncs.shtml
 ○ yes ○ no  769.342   http://www.themonitor.com/broncs.shtml
 ○ yes ○ no  759.342   http://www.themonitor.com/broncs.shtml
 ○ yes ○ no  749.342   http://www.themonitor.com/broncs.shtml
 ○ yes ○ no  662.561   http://www.tsl.texas.gov/LD/AcademicLibraries1997/ = -1 FACE
 ○ yes ○ no  659.561   http://library.northernlight.com/PN20000112020036 = -1 FACE
 ○ yes ○ no  653.561   http://library.northernlight.com/PN20000112020036 = -1 FACE
 ○ yes ○ no  643.561   http://library.northernlight.com/PN20000112020036 = -1 FACE
 ○ yes ○ no  633.561   http://library.northernlight.com/PN20000112020036 = -1 FACE
 ○ yes ○ no  631.708   http://tcsul.texshare.edu/Services/libcats/all/utp = -1 FACE
 ○ yes ○ no  623.561   http://library.northernlight.com/PN20000112020036 = -1 FACE
 ○ yes ○ no  621.708   http://tcsul.texshare.edu/Services/libcats/all/utp = -1 FACE
 ○ yes ○ no  611.708   http://tcsul.texshare.edu/Services/libcats/all/utp = -1 FACE
```

Figure 4: Refined Result for "UTPA" after 2 Interactions and 5 Examples

References

Angluin, D. 1987. Queries and concept learning. *Machine Learning* 2:319–432.

Baeza-Yates, R., and Riberiro-Neto, B. 1999. *Modern Information Retrieval.* Addison-Wesley.

Bollacker, K.; Lawrence, S.; and Giles, C. L. 1998. Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the Second International Conference on Autonomous Agents*, 116–113. New York: ACM Press.

Bollacker, K.; Lawrence, S.; and Giles, C. L. 1999. A system for automatic personalized tracking of scientific literature on the web. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, 105–113. New York: ACM Press.

Brin, S., and Page, L. 1998. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh World Wide Web Conference.*

Chakrabarti, S.; Dom, B.; Raghavan, P.; Rajagopalan, S.; Gibson, D.; and Kleinberg, J. 1998. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh World Wide Web Conference*, 65–74.

Chen, Z., and Zhu, B. 2000. Linear lower bounds for learning from relevance feedback. submitted for publication, February 2000.

Chen, Z.; Meng, X.; Fowler, R.; and Zhu, B. 2000a. FEATURES: real-time adaptive feature learning and document learning for web search. submitted for publication, May 2000.

Chen, Z.; Meng, X.; Zhu, B.; and Fowler, R. 2000b. Websail: from on-line learning to web search. submitted for publication, February 2000.

Chen, Z.; Meng, X.; and Fowler, R. H. 1999. Searching the web with queries. *Knowledge and Information Systems* 1:369–375.

Chen, Z. 2000a. Lower bounds for similarity-based relevance feedback over discretized vector spaces. submitted for publication, March 2000.

Chen, Z. 2000b. On the query complexity of web search. manuscript, February 2000.

Gibson, D.; Kleinberg, J.; and Raghavan, P. 1998. Inferring web communities from link topology. In *Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia.*

J.J. Rocchio, J. 1971. Relevance feedback in information retrieval. In Salton, G., ed., *The Smart Retrieval System - Experiments in Automatic Document Processing*, 313–323. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Kleinberg, J. 1999. Authoritative sources in a hyperlinked environment. *Journal of ACM* 46(5):604–632.

Lawrence, S.; Bollacker, K.; and Giles, C. L. 1999. Indexing and retrieval of scientific literature. In *Proceedings of the Eighth ACM International Conference on Information and Knowledge Management.*

Littlestone, N. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning* 2:285–318.

Rennie, J., and McCallum, A. 1999. Using reinforcement learning to spider the web efficiently. In *Proceedings of the Sixteenth International Conference on Machine Learning.*

Salton, G.; Wong, A.; and Yang, C. 1975. A vector space model for automatic indexing. *Comm. of ACM* 18(11):613–620.

Salton, G. 1989. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer.* Addison-Wesley.

Selberg, E., and Etzioni, O. 1995. Multi-service search and comparison using the metacrawler. In *Proceedings of the Fourth World Wide Web Conference.*

Selberg, E., and Etzioni, O. 1997. The metacrawler architecture for resource aggregation on the web. *IEEE Expert* 12(1):8–14.