A Search-Engine-Topology to Improve Document Retrieval on the Web[*]

Uwe Roth, Andreas Heuer, Ernst-Georg Haffner, Christoph Meinel
Institute of Telematics
Bahnhofstr. 30-32
D-54292 Trier, Germany
{roth | heuer | haffner | meinel}@ti.fhg.de

Abstract: Nowadays, search-engines are the only reasonable solution for searching and finding data on the Internet. Search-engines are confronted, though, with four major problems: The Internet is growing rapidly. Search-engines cannot keep up with indexing new servers. It is becoming harder and harder to keep indexed web-pages up to date. It is difficult (or even impossible) for search-engines to index dynamically generated web-documents. Internet documents which are not based on plain text (.doc, .pdf, .class, .wav, .mov) cannot be indexed.
All of these problems are related to the fact that search-engines are only able to "act" towards web-servers like normal users. They can only obtain information via HTTP.
This paper aims at presenting an alternative approach. We will describe a topology of search-engines. The basic module is a local search-engine on the corresponding web-server and a protocol for creating the topology. Existing search-engines may use the topology in order to obtain better and faster results from web-sites.

Introduction

With the rapid growth and expansion of the Internet it became clear soon that it is not the availability of information on the Internet but the location of information which is causing problems. Search-engines were created to make those distributed information accessible. Today, there are a number of different search-engines, each based on a different approach. Some are being taken care of by editors, some are kept up to date by spiders and robots which search the Internet for new sites. All of them, though, match up in the fact that their approaches cannot cope with the real size of the Internet so that they, in consequence, cover only a fraction of the Internet. The quality of the presented data is often inadequate as well.

On the other hand, it is hard for search-engines to keep their existing score of data up to date. Meta-search-engines that are able to forward requests to different search-engines at the same time may diminish these difficulties but they cannot be considered a real solution. On the other hand, when carrying out complex requests (for example a search for a phrase or for a document containing two adjacent words) search-engines need to copy a text version of the document (which will probably be encoded) onto the server of the search-engine. This means that search-engines aiming at making the whole WWW available for requests will need an (encoded) copy of the entire WWW on the server of the search-engine. Even though hard disc storing space is getting cheaper all the time, such a venture would simply be a wasted effort. Considering this, it is not surprising that studies on search-engines conclude that only 16% of the entire WWW are getting evaluated by a search-engine and that only 42% of the WWW are getting summarized by all search-engines (Lawrence, 1999).

Dynamically generated HTML-pages are on their way to becoming the greater problem: These pages exist only while data is being transmitted to the web-server. The page is being generated after feedback with a database. It is here that robots and spider meet their limitations. An even bigger part of such data gets ignored.

The following example illustrates this fact: At an online-bookstore, the web-page of any given book can only be accessed by entering the name of the author, the title or the ISBN code. In fact, though, only a small fraction of actually available books can be accessed via the usual link-navigation.

Meantime, the Internet is increasingly becoming a kind of dump for documents which are not based on text and which, as a result, cannot be indexed. This is especially annoying when text-documents are offered only in non-

[*] as in proceedings of WebNet 2000 – Conference on the WWW and Internet, San Antonio, Texas, USA, October 30-November 4, 2000, pp 470-475

textual formats (.doc, .pdf). Other documents (images, audio files, video files, or even program code) should be accessible via keywords as well though.

One last, less important, aspect is the lack of any option to influence the quality of the results of the search-query. There is neither an option to increase time for search-queries to get higher quality of the result nor an option to have e-mail posting on search queries.

In this paper, we will present a topology for search-engines as a solution for the problems described above.

One part of the solution is the definition of a new Internet protocol (IPSE = Interchange Protocol for Search Engines). The protocol is not used for determining which search-engines should deal with search-queries internally or to decide which methods to use when conducting a query. Its objective is solely to determine a frame for the communication between search-engines.

This paper pursues an entirely new approach for improving search-engines. We do not try to document the whole of the Internet and to draw out heap after heap from its hoard of information. What we do, on the contrary, is to broadcast locally available information of every server in the world.

It is an advantage that the defined topology supports existing search-engines to collect information and help to make them available faster. Thus, we can create this topology without endangering existing search-engines and without expecting all Internet-servers to support our topology from the start.

The topology described in this paper and the corresponding protocol serve as the basis for prototypes which are being developed at the moment.

The Topology

The Internet does not offer sufficient support for search-engines. The HTTP-protocol was neither invented to help search-engines with locating data nor was it created for making additional document-information available. The only way for search-engines to analyze a web-page is to first load a document, then to store it on a textual basis and after that, to indexed it. An additional task for search-engines is analyzing tags in order to generate meta-information and to locate links to other documents so that a list of supplementary documents to be visited in the future can be generated. By working through the entire list, one hopes to capture the contents of the whole web-site and of the whole WWW as well.

The question remains, though, were the greatest amount of information about one web-page is to found. It seems that the place to look would be the web-server that is displaying the website. At the present stage of research, what is missing is an adequate opportunity of presenting this kind of information to search-engines.

When creating a topology of search-engines, the initial step is a service located on the web-server to be used by search engines when accessing information via the web-site.

Such information is:

- A list of possible URLs
- Word-lists for the URLs
- Plain-text for the URLs
- Meta-information about the URLs

Initially, this service is used only by already existing search-engines for more complete and faster queries on information via a web-site.

In a second step, the service offers the possibility of replying to queries via the web-site directly. Many servers today are already equipped with an option for making query requests via the web-site. Thus, it is reasonable to implement this feature into the search-engine service right from the beginning.

The more important reason, however, is the fact that by making it possible to transfer queries to search-engine-services of different web-servers we can create a search-engine-topology.

This goes to show that query requests to search-engine servers will not always only concern local web-servers but may also have an impact on unrelated web-servers. The search-engine service is only able to reply to request queries that it is dealing with itself or that it is coaching.

The different ways of coaching will be described in the following chapter.

Being able to coach foreign web-sites as well makes it possible to give a scaling of the quality of replies.

For the purpose of taking into account currently available document resources when processing a request, the request can be forwarded to the search-engine-service that is directly connected with the web-server. In a case where it is of no importance whether information about documents are up to date or not, the search engine may operate on the local basis of data that is available at the time (nevertheless, this data might be obsolete).

There is no single, fixed procedure that search-engine-services have to follow in order to access data on another server. It can be done either by using the locally available search-engine-service or via HTTP or by using additional sources. Likewise, there is no fixed procedure for treating request queries. In the future, the quality of replies will continue to be dependent on the implementation of the search-engine.

It should be remarked here that there is a risk for the search-engine-service to be misused for presenting inaccurate information in order to get a higher access frequency on websites. It is for the search-engine service to decide which servers to "trust" and to what degree.

Push-services and e-mail-services are also possible features of a search-engine service. Push-services are used to obtain information at an unspecified period in time. This means that the flow of information will be initiated from a foreign search-engine-service and not by request.

The topological hierarchy of the search-engines will be relatively plane. The first level will contain all local search-engines - that is, search-engines that are directly connected to the web-server. The second level will be comprised of search-engines which aim either at capturing the entire Internet or which offer services specified by topic. The third level will continue to contain meta-search-engines. In our concept, search-engines will need authorization when confronting search-engine-services. This way, a scaling of request features will be possible. Access to the search-engine-service may actually be restricted to such a degree that the service will no longer be of any use for meta-search-engines trying to obtain information.

Coaching

The search-engine-service described above will be displayed by the IPSE-server. The IPSE server is the server that is implementing the Interchange Protocol for Search-Engines. Each IPSE-server is able to coach several HTTP- or FTP-servers.

We have already hinted at their being different types of coaching:

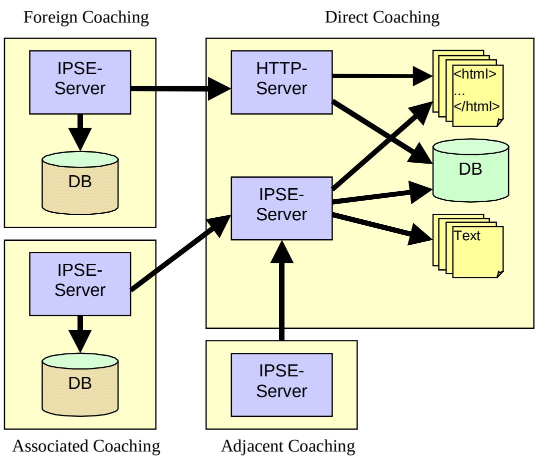

Figure 1. Different kind of coaching

Direct coaching	The IPSE-server can directly access the document root of the associated HTTP- or FTP-server. It is able to access additional file information (e.g. meta-data or plain-text variations of documents that are not available as plain text). The server is informed about the fact that some pages are being generated by CGI scripts or by servlets. It is familiar with scripts that can carry out a dynamic generation of the URLs in question. Information about documents with partial access authorization will be given to authorized users.
Associated coaching	The server cannot directly access the document root of the associated HTTP- or FTP-server. However, the server is informed about other servers that are conducting direct coaching. The server might be able to get authorization for requesting information that has access restrictions. In a cyclic process, all the data about documents which is directly available from the other IPSE-server is being copied.
Adjacent coaching	The server cannot directly access the document root of the corresponding HTTP- or FTP server. However, the server is informed about other servers that are conducting direct coaching. The server might be able to get authorization for requesting information with access restriction. The server has to forward all requests that concern the coached party of this IPSE-server.
Foreign coaching	The server has no direct access to the document root of the corresponding HTTP- or FTP-server. The server is not informed about any servers conducting direct coaching. The server asks for all information that is available via HTTP- or FTP-protocol.

Associated, adjacent and foreign coaching are all part of the type „indirect coaching“. Foreign coaching works according to the same principles that are valid when search engines interact with HTTP-servers.

There is no fixed procedure to follow when setting up lists of servers on the IPSE-server. Likewise, there is no rule determining what kind of coaching is to be used for confronting HTTP- or FTP-servers. Manual configuration seems useful for specialized IPSE-servers which deal with topic-related requests – especially when the requests ask for data with access restrictions. Just like existing search engines, IPSE-servers intending to reply to requests of the general kind will continue to put their robots and spiders to work when carrying out a query. However, they will check first if there is any IPSE-server.

One sub-category of coaching is "limited coaching". Limited coaching can be used for any type of indirect coaching (associated, adjacent or foreign). In this case, it is not the entire store of documents on the foreign IPSE-server that is being coached but only a part of it. Limited coaching is necessary when specialized, topic-related IPSE-servers are being established. The IPSE-server must be configured with manual input in this case.

The protocol

IPSE is a – like FTP and SMTP – a text-based protocol (Postel, 1982/1985). It works according to the same request/reply mechanism: First, a command is being given to the IPSE-server by the transmission of a line of text. The IPSE-server replies with one or with several lines of text. The last line will start with an error code. Next, the commands defined in IPSE will be carried out.

Contrary to other protocols, IPSE has no real, fixed pattern for the internal processing of the commands. A query request may have different results, depending on the way the search mechanism has been implemented.

General commands:

HELP [<command>]	Help: Lists all commands or gives information on a specific command.
QUIT	Quit: Closing a session.

Authorization:

HELO <computer-name>	Hello: The computer name is entered as FQDN (Fully Qualified Domain Name). This is important when one IPSE-server connects to another.
USER <user>	User: Entering the user name of the user who is receiving the information. The name cannot be an e-mail address but has to be a user name that is configured in the IPSE-server. Parts of web servers protected by htaccess can be made accessible for specified users this way. It would also be possible to block entire commands for users without authorization. This depends solely on the configurability of the IPSE-server. "anonymous" is a protected user name, reserved for an anonymous access. Users who do not use the USER-command will automatically be labeled "anonymous".
PASS <password>	Password: For authorization to the system, a password is needed. Like in FTP, the password is being transmitted in plain text. A later stage of the prototype will establish a SIPSE, like SSH, SCP or SHTTP accordingly. This does not in any way influence the protocol or the structure described here.

Information on coaching:

CCHG [<server-name including protocol>]	Coaching: Gives a list indicating the type of coaching or the specified server including the type of coaching.

Information on documents:

URLS <Server-Name>	URLs: Gives a list of all URLS of the available documents. When entering a specific date, only URLs that have

[date]	changed since this period of time will be displayed. For servers with limited coaching, only the subset of the coached URLs will be given.
LNKS <URL>	Links: Gives all the links appearing in one document.
WRDS <URL>	Words: Gives a word-list for the indicated URL and the frequency of the documents in the document. This information does not necessarily have to be derived from the document defined by the URL. The list could be derived from an alternative source if the document was not text-based or if only a part of the HTML-document had been submitted to the search – for example, because navigation text was to be ignored. Stop-lists are not used.
AWRS <Server-Name>	All Words: Like WRDS but for all coached URL's.
PLIN <URL>	Plain: Gives the plain text for an URL. Like for WRDS, this text may originate from an alternative source.
META <URL> [<meta>]	Meta: Gives all meta data of an URL or the specific meta date. Besides meta-tags which are defined in HTML-code, meta information can be derived from an analysis of the text (e.g. the title from the title tag or the transmission of the H1-tag to the keyword meta-data.) Alternative sources for meta data are also possible. Meta-data should contain at least all characteristics defined in the Dublin-core and also mime types.
UABO <Mode>	URL-subscription: The UABO-command allows to have URLs of up-dated or new documents delivered as Push-service. The mode START indicates that the user is subscribing to the service, the mode STOP indicates that the subscription is cancelled.
ACTU <URL>	Actualized-URL: delivers the most recent or new URLs if a subscription for the server exists

Queries:

MAIL <E-Mail-address>	Mail: Defines an e-mail address for receiving the result of a query if it was an offline-request.
MODE <Modus> [<Hops>] [<Max-Time>]	Mode: Defines the type of request. L(ocal) only uses locally available data, E(xternal) uses local data only when conducting direct coaching. Requests to adjacent and associated servers get forwarded. In this case, Hops indicates the maximum number of servers that will be used for the forwarding process. If the command mode „M“ is attached, the reply will be delivered a part of the mail-message. If the mode „P“ is used, the information will be delivered as push-service. In both cases, the result of a request is a request-id. It is calculated by the server and is unique. If the „M“-mode was used, the request id is being delivered as part of the mail message. If the mode „P“ was used, the IPSE-server initiates a connection to the requesting IPSE-server and transfers the results along with the RSLT-command. After a IPSE-servers has forwarded a request, it has to generate a reply itself. To do this, it will indicate how long it can afford to wait for the reply by using the parameter <Max-Time. If this parameter is not indicated, it depends on the configuration of the server how long it will wait for a reply.
QERY <Start-number> <number> <request>	Query: Generation of a request. For simplicity, we will refrain here from giving a detailed definition. Request syntax, nevertheless, is one of the few aspects where we are able to exercise some influence on the internal features of the IPSE-Server. Only in syntax can we test complex request options. It is up to the individual IPSE-server to implement only parts of the possible scope of the implementation. Possible operators are AND, OR, NOT, NEAR, NOT and META. The META-operator allows to request meta data. Examples: META ("MIME-Type") = "text/html" META ("DATE") "1999-12-07" META ("KEYWORDS") IN ("PAPER", "CONFERENCE", "INTERNET") The result of a request is a list of URLs (if the mode is not M or P). Otherwise, a request id will be returned. The actual results – using the request id -are transferred later on by using Push-service. Query requests are sorted according to relevance. By entering a start number and number of results the outcome can be influenced.
RSLT <ID> [<URL>]	Result: This command is used for results that get transmitted via push-service. Each located URL sets off a RSLT with request id. If the URL is missing, the transmission will be interrupted.

Summary

In this paper, we have been describing the need for a specialized service for improving the quality of search-queries. The approach discussed in this paper offers a solution to the problems mentioned in the text. Our approach is based on the simple observation that the most up-dated and far-reaching pieces of information are available on local websites and local webservers. The topology presented in this paper and the protocol defined here offer solutions that help to broadcast these pieces of information to the outside world and to reduce the amount of data stored on other search-engines as well. However, even in the future, the quality of broadcasted information will continue to depend on the quality of website maintenance. If there is no continuous proper maintenance of meta data, it will not be possible in the future to request this type of data. Administrative efforts are necessary if links to dynamically generated websites are to be made accessible. Additional examples come to mind easily. The topology presented here and the protocol that has been defined might serve as the basis for the realization of such projects.

Related Work

The approach presented in this paper does not take into account how different documents are related to each other by context. Our approach does not give credit to the objective of the „Semantic Web“ (Berners-Lee, 1999). The problem of trying to capture the vastness of the Internet must be detained for yet another ‘period of grace’.

At the time, WebDAV (WebDAV, 2000; Slein 1998) is trying to built an extension for HTTP which allows users to collaboratively edit and manage files on remote web servers. Our approach does not contradicts this. WebDAV does offer more information about documents, but was not built as a platform for a search-engine-topology.

One spin-off of the effort to define WebDAV was the definition of the DASL-protocol (DAV Searching and Locating) (Reddy, 1999). DASL concentrates on request features. It does not emphasize on the topology where the protocol is been used.

Several efforts have been made to see the WWW was as a big database. Special languages have been developed to declare questions about web-contents (Mendelzon, 1996; Konopnicki, 1995; Lakshmanan, 1996; Fiebig, 1997). These languages give not an answer to the question how the information could be made available. They only allow flexible questions about information that is already available. A good place for such languages would be the QERY-command in our protocol.

Our approach of a search-engine-topology can be compared with the architecture of mediator-systems to handle semi-structured data (Liu, 1996; Levy, 1996; Abiteboul, 1997; Chawathe, 1994; Hammer, 1997). In such a system our IPSE-Server would have the function of a mediator and of a wrapper of web-servers. These architectures are not especially built for the internet and have a more general approach of semi-structured data. So they do not consider the web-specific problems.

IPSE is only a means for creating a search-engine topology. It has kept problem-focused and simple on purpose. We have abandoned here to implement a feature for transmitting data in the XML-format.

However, changes in the strategy would neither affect our approach nor the topology.

References:

Lawfence, Steve; Lee Giles, C. (1999): *Accessibility of information on the web*; Nature 7/99, S 107, http://www.wwwmetrics.com/

Postel, J.; Reynolds, J. (1985): *RFC 959 - File Transfer Protocol (FTP)*; Online-reference http://sunsite.auc.dk/RFC/rfc/rfc959.html

Postel, J. (1982): *RFC 821: Simple Mail Transfer Protocol*; Online-reference http://sunsite.auc.dk/RFC/rfc/rfc821.html

Berners-Lee, Tim; Conolly, Dan; Swick, Ralph R. (1999): *Web Architecture: Describing and Exchanging Data*; Online-reference http://www.w3.org/1999/04/WebData

WebDAV (2000): *WebDAV Resources*; Online-reference http://webdav.org

Slein, J; Vitali, F; Whitehead, E; Durand, D (1998): *RFC 2291 - Requirements for a Distributed Authoring and Versioning Protocol for the World Wide Web*; Online-reference http://sunsite.auc.dk/RFC/rfc/rfc2291.html

Reddy, Saveen; Lowry, Dale; Reddy, Surenda ;Hendson, Rick; Davis, Jim; Babisch, Alan (1999): *DAV Searching & Locating*; Online-reference http://www.ietf.org/internet-drafts/draft-ietf-dasl-protocol-00.txt

Mendelzon, A. O.; Milo, T. (1996): *Querying the World Wide Web*; In Proc. 4th Int. Conf. on Parallel and Distributed Information Systems (PDIS)

Konopnicki, D.; Shmueli, O. (1995): *W3QS: A Query System for the World Wide Web*; In Proc. 21th Int. Conf. on Very Large Databases (VLDB), pages 54-64

Lakshmanan, L. V. S.; Sadir, F.; Subramanian, I. N. (1996): *A Declarative Language for Querying and Restructuring the Web*; In Proc. 6th Int. Workshop on Research Issues in Data Engineering, RIDE'96

Fiebig, T.; Weiss, J.; Moerkotte, G. (1997): *RAW - A Relational Algebra for the Web*; In Workshop on Management of Semistructured Data

Abiteboul, S.; Vianu, V. (1997): *Queries and Computation on the Web*; In F.N. Afrai, P. Kolaitis, editors: Proc. 6th Int. Conf. on Database Theory (ICDT), pages 262-290

Chawathe, S.; Garcia-Molina, H.; Hammer, J.; Ireland, K.; Papakonstantinou, J.; Ullmann, J.; Widom, J. (1994): *The TSIMMIS project - integration of heterogeneous information sources*; In IPSJ Conference; Tokyo

Hammer, J.; Garcia-Molina, H.; Cho, J.; Aranha, R.; Crespo, A. (1997): *Extracting Semistructured Information from the Web*; In Workshop on Management of Semistructured Data

Levy, A. Y.; Rajaraman, A.; Ordille, J. J. (1996): *Querying Heterogeneous Information Sources Using Source Descriptions*; In Proc. 22nd Int. Conf. on Very Large Data Bases (VLDB); pages 251-262

Liu, L.; Pu, C.; Lee Y. (1996): *An Adaptive Approach to Query Mediation Across Heterogeneous Information Sources*; In Proc. 1st Int. Conf. on Cooperative Information Systems (CoopIS)