CORA Cork Open Research Archive
Cartlann Taighde Oscailte Chorcaí

Title	Open meta-search with OpenSearch: a case study
Authors	O'Riordan, Adrian P.
Publication date	2007-12-01
Original Citation	O'Riordan AP (2007) Open Meta-search with OpenSearch: A Case Study. Technical Publication [Online]. Available at http://www.cs.ucc.ie/~adrian/Metasearch-OpenSearch-AORiordan.pdf
Type of publication	Report
Link to publisher's version	http://www.cs.ucc.ie/~adrian/Metasearch-OpenSearch-AORiordan.pdf
Rights	© 2007, the Author.
Download date	2026-11-05 17:02:37
Item downloaded from	https://hdl.handle.net/10468/982

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Open Meta-search with OpenSearch: A Case Study

Adrian P. O'Riordan
Computer Science Department,
Sir Robert Kane Building, University College Cork, Cork, Ireland
a.oriordan@cs.ucc.ie

Abstract: The goal of this project was to demonstrate the possibilities of open source search engine and aggregation technology in a Web environment by building a meta-search engine which employs free open search engines and open protocols. In contrast many meta-search engines on the Internet use proprietary search systems. The search engines employed in this case study are all based on the OpenSearch protocol. OpenSearch-compliant systems support XML technologies such as RSS and Atom for aggregation and distribution. The meta-search engine itself combines the ranked lists of the chosen search sources based on user-supplied weightings. This is implemented in Lucene, a free open source information retrieval library.

Keywords: OpenSearch, Web Search, Meta-search, Aggregation, RSS, Ranking, Search Interface
Categories: H.3.1, H.3.4, H.3.5, H.4.0, H.5.2

1 Introduction

A meta-search engine is a search engine that utilizes multiple search engines by sending a user request to a number of different engines aiming to improve recall in the process. A related technology, aggregation combines different information sources to generate a composite view of that information. These systems whether search engines or comparators, are often proprietary systems but recently there has been interest in building, using and combining free open source systems. Up to the present individual search engines have remained far more popular that meta-search technology. Aggregation technology, such as price comparison Websites, has had more commercial success.

It has been known for nearly a decade that meta-search engines can improve recall or search coverage [Ng and Kantor, 98] and custom Web portals can provide easier access to specialised information. While meta-search engines may allow more Web content to be searched using a single query than a stand-alone search engine, there is the issue of combining results and the precision of the result set can be poor.

A reason for choosing open search technology is that the terms of service (TOS) of the most popular search engines constrain the use of such systems; they can have restrictive APIs, and forbid screen scraping or "piggybacking". In addition the APIs often place limits on the number of returned URLs. For example, Google Search's SOAP API, withdrawn as of 2006 [Google SOAP, 06], had limits on both the number

Technical Report, Computer Science Department, University College Cork, Cork, Ireland. December 2007.
Available from the author at www.cs.ucc.ie/~adrian/Metasearch-OpenSearch-AORiordan.pdf

of returned results and the number of queries that could be issued per day. The new Google REST API has an even more stringent TOS which prohibits un-licensed use completely [Google REST, 08]. Also paid inclusion (sponsored links) is an issue with commercial search engines that isn't an issue with free open systems. Misinformation or persuasion is not a new problem in search engine design [Marchioni, 97].

In this paper we present our experiences of designing and building an open meta-search system based on OpenSearch. The structure of the paper is as follows. Section 2 describes relevant background material relating to meta-search and aggregation. Section 3 details the prototype system developed. Section 4 gives conclusions and suggestions for future work.

2 Background: Meta-search and Aggregation

This section discusses the state of the art in meta-search and aggregation technology.

2.1 Meta-search in the Field

Of the hundreds of search engines built only a handful predominate in terms of popularity. Just three search engines have approximately 84% of the U.S. market share between them: *Google Search*, *Yahoo! Search*, and *Live Search* (Microsoft). This figure was calculated from "U.S. Expanded Search Queries by Search Engine" data for December 2007 [com Source, 07]. Search provider takeovers and mergers have only consolidated market dominance, for instance *Ask.com* has acquired and integrated former search services and technologies such as *teoma*, the *Excite* search engine, *iWon* (search and lottery), *MyWay* (portal) and *bloglines* (feed aggregator). The market leaders have large frequently updated indices and offer a range of services consolidating their market leading positions.

It is hard to differentiate the hundreds of other search services on the Web. Many search services now offer extra functionality to set them apart such as the use of clustering, subject specific or domain specific vertical search, multi-search, feed/blog search, and image/audio/video search. For example *Clusty* and *kartOO* both use a visual metaphor to cluster the result set. Multi-search systems can send a query to multiple search engines but do not attempt to combine the results instead displaying them in separate lists, panes, or frames. An example multi-search system is *Multi-Search-Engine.com*.

MetaCrawler, developed in the mid-1990s, was one of the first meta-search engines to appear on the Web [Selberg and Etzioni, 95]. A more recent example, *Dogpile* [Jansen *et al.*, 07], searches using all of the following search services: *Google Search*, *Yahoo! Search*, *LookSmart*, *Ask.com*, and *Windows Live Search*. Meta-search engines have a relatively small market share despite heralded benefits such as increased coverage. Other meta-search services available currently (as of late 2007) include *jux2*, *InfoGrid*, *zuula*, *fazzle*, and *Ixquick* but note that these types of Websites appear, change names and disappear relatively quickly.

The ranked results from each constituent engine can be presented separately or more often integrated into a single results list. Some newer search technologies also cluster the combined results. Typically the user has neither the option of specifying which search engines to employ nor does the system try to exclusively use the search engines most likely to handle the query best. Different search engines return quite different sets of results for the same query. A study involving 10,316 user-entered search queries across three major search engines, *Google Search*, *Yahoo! Search*, and *Ask.com* found that only three percent of all results returned were across all these Web search engines and the percentage of total results unique to one search engine was established to be 85 percent [Bharat and Broder, 98]. The small level of overlap reflects major differences between the engines in terms of indexing and ranking. Empirical results, though only indicative, provides evidence that search engines differ considerably in the returned results sets and rankings. This alone is motivation for work on combining searchers.

The term Federated Search is used to describe related work in the area of library and information science which typically employ the Z39.50 protocol [Schatz *et al.*, 99]. SRU (Search/Retrieve via URL) is a newer standard search protocol for representing queries maintained by the Library of Congress [McCallum, 06]. LeVan compares and contrasts SRU with OpenSearch which use utilize and describe later [LeVan, 06].

The potential advantages of meta-search can be summarized as: (1) A single interface to multiple resources; (2) The searcher may not know what collection to target so he/she targets many; (3) No single collection may have all the information one requires; (4) Reduction in the time spent searching; (5) Can add summary or comparison information; (6) Reduce or eliminate advertising.

2.2 Search versus Meta-search

Meta-search engines create what is sometimes called a virtual database – processing a query on-the-fly by spawning multiple queries and sending to multiple sources. Researchers at Google [Madhavan *et al.*, 06] have challenged the efficacy of meta-search and federated search as opposed to using a single large index on (mirrored) clusters of computers. One argument put forward by Madhavan *et al.* is as follows: "with the numbers of queries on a major search engine, it is absolutely critical that we send only relevant queries to the deep web sites; otherwise, the high volume of traffic can potentially crash the sites" [*ibid.*]. This is an issue can be dealt with at the query processing stage and then only suitable underlying search engines chosen. Another of their arguments is as follows: "virtual approach makes the search engine reliant on the performance of the deep web sources" [*ibid.*]. It is important that search engines are polite and that a meta-search facility discard unresponsive sources.

A strong case can be made for meta-search in other regards. Web content is dynamic so small tailored engines should be able to crawl and update their indices more regularly. Another benefit of meta-search relates to structured queries and domain models where for example form submissions are needed to access the Deep Web; it is

very hard for a single index to represent all the possible schemas in use. In summary we believe searchers will need access to specialized Web collections and specialized search engines will be able to access and/or process such content effectively.

2.3 Aggregation

An emerging technology (from approximately 2005 onwards) is feed aggregation. This is an aggregate-and-wait approach in contrast to the seek-and-find approach of most meta-search engines. Aggregation is currently primarily employed in summarizing, comparing, or recommending news feeds or product information. Examples include news aggregation (*Google News*), shopping aggregators (*Shopping.com*), and service aggregators (*Realtor.com*). The current technological solution is to use RSS (Really Simple Syndication) [Harvard, 03] or ATOM [IETF, 05] feeds. Search aggregators employ standardised XML technology for extracting data sources and feed formats such as RSS and Atom. Content providers can control the availability of data via Web syndication. Aggregators subscribe to particular feeds or "channels". Many syndication and micro-content publishing methods such as RDF (Resource Description Framework) [W3C, 04] and Topic Maps [ISO/IEC, 03] have been developed, but RSS (current specification RSS 2.0), a simple method for publishing frequently updated information such as news, podcasts and blogs, is the most widely used at present.

Meta-search and aggregation face technical challenges with regard to ranking and the elimination of spam/noise. Researchers in Information Retrieval and Data Retrieval have examined these issues in combining result sets from multiple databases and document collections or from multiple indices over the same data [Fagin *et al.*, 01]. This knowledge can be applied to search and aggregation.

3 A Meta-search Prototype

We developed a prototype system that uses the OpenSearch protocol to enable search aggregation in a standardized format [OpenSearch]. In our prototype a user can issue a text query (word or phrase) with a number of modifiers: Boolean logic (AND, OR and NOT operators), nesting of same, and required fields. One use case is where the meta-search engine only employs the engines that can explicitly handle the type of query formulation in the query issued. For example, some engines allow nesting in Boolean expressions whereas others don't. Hence, a query such as Retrieval AND (Text OR Information) will be sent to the first set but not the latter as part of the overall search. Figure 1 shows the Chooser module selecting the search engines to send the query and the Integrate module combining the results.

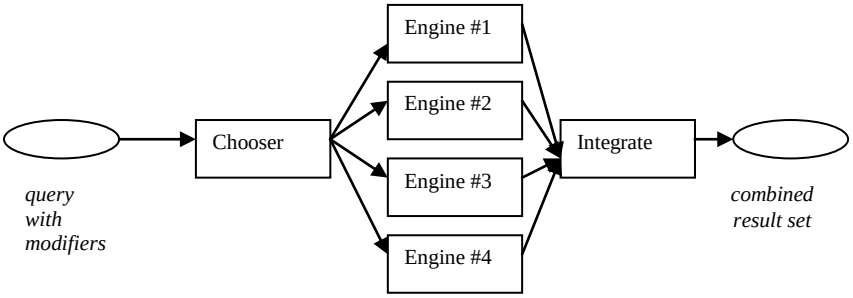


Figure 1: Schematic of Customizable Meta-search

The main advantage of search aggregation over proprietary meta-search is flexibility. Meta-search in this model can be viewed as an application of aggregation. Different types of data such as news feeds and product catalogues can be added as data sources. OpenSearch is a similar technological solution to feed aggregation except aimed at search and not tied to a single format but instead supportive of various versions of RSS and Atom.

The OpenSearch protocol was developed by *Amazon, Inc.* search subsidiary *A9* [OpenSearch]. Information and downloads are available from www.opensearch.org. OpenSearch is a technology freely available under a Creative Commons license that enables search aggregation in a standardized format. As of 2007 only a handful of general purpose search engines support OpenSearch, for instance the *A9* search service itself, *YaCy* [YaCy], and *mozDex* [mozDex]. There are some other topic-specific and desktop search applications that support OpenSearch but they aren't used in this prototype. OpenSearch 1.1, the latest version, allows results to be returned in both RSS 2.0 or Atom 1.0 format.

Various extensions to OpenSearch have been proposed or are in development. For example a proposed extension to handle SRU will allows SRU (Search and Retrieval via URL) queries to be used within OpenSearch contexts. Other proposed extensions include support for mobility, e-commerce, and geo-location.

3.1 Web Interface

Figure 2 below shows the main screen of the prototype system. A user can choose to include or exclude each listed engine (currently *A9*, *YaCy*, *mozDex*, and *alpha*) by checking or un-checking the corresponding checkbox to the right. Users can prefer or bias particular engines by setting weights. The three available weights are "High", "Normal" and "Low", chosen so as to be intuitive to users. A numeric valued weight may not make sense to a user, as they would not know how the weight is applied in the background. Currently a selection of High doubles the weighting of those corresponding results for that search engine, whereas Low halves the weighting of results. The weights are applied when the results are merged. The system analyses the query and if a user enters a Boolean query the query will be routed to only search engines that support the Boolean features used in the query. Note that among existing meta-search engines some such as *WebCrawler* support nested Boolean queries

whereas others such as *InfoGrid* do not. Additional customization features are site-specific search, in-title search, and varying the number of results displayed.



Figure 2: User Interface of OpenSearch Prototype

A major consideration for meta-search engines is the response time. Users do not want long waits for results to be returned. In our current implementation each search is run as a separate thread. Results are therefore passed to the Integrator module at approximately the same time from each search. In the first incarnation of our meta-search engine, threads were not utilised and the response time was poor. Non-responsive sources are dropped after a short period of time.

3.2 Implementation with Lucene and OpenSearch

A number of software tools can be used to read and write OpenSearch results ([MediaWiki, 06], [MovableType, 05]). Our implementation was build using the free *nutch* software library [Khare *et al.*, 05]. This builds on *Lucene*, an information retrieval library in Java, adding Web-specifics, such parsers for HTML and support for a few other document formats. We did not require the *nutch* crawler in this prototype since the search sources carry out their own searching (and indexing) and

provide individual results. The meta-search engine has responsibility for choosing these sources, feeding them the queries, and combining the results.

Lucene, an Apache top-level project, is a free open source information retrieval library. *Nutch*, which uses *Lucene*, is designed to scale to the whole Web and support high traffic volumes. Parsing and querying are all customisable via plugins. A Chooser component discovers and fetches sources from a Web database, WebDB, a custom store of known OpenSearch-compliant search engines. It is easy to add or remove individual engines. There is a directory of public OpenSearch Description Documents available at www.opensearch.org.

OpenSearch consists of XML descriptor files that identify and describe OpenSearch aggregators, OpenSearch RSS or OpenSearch Response formats for providing open search results. OpenSearch Description Documents are used to describe the interface to a particular search engine. The XML elements include OpenSearchDescription, ShortName, Description, Url, Contact, Tags, Query, Attribution, SyndicationRight, InputEncoding, and OutputEncoding.

Note that a range of other search and syndication technologies can read *OpenSearch* data including the *ROME* (Sun/java.net) and *Abdera* (Apache) libraries for Web syndication, and data servers such as Google's *GData* API and OpenLink's *Virtuoso*. Linking to these is outside the scope of this paper.

3.3 Ranking in Prototype

We considered ways of combining or aggregating ranks. A simple method is the "take the best rank" algorithm on any duplicate results. We wanted a fairer algorithm so we examined a second approach, Borda's positional method [Saari, 95], which is the basis of our implementation. The Borda method is a system where voters rank candidates in order of preference. Each search engine is regarded as a voter, and assigns n votes to its top result, n – 1 to the second result, n – 2 to the third result and so on. The meta-search engine then gathers up all the votes for the retrieved web pages from all the search engines and the ranking is determined by summing up all the votes. The algorithm we used is a weighted version of the Borda positional method. In this algorithm search engines (voters) are not treated equally but their votes are considered along with a preference set by the user. So the vote for the i^{th} result of search engine j is calculated as $Vote_{i,j} = w_j * rank_i$ where w_j is the weight assigned to the search engine by the user and $rank_i$ is the rank assigned to the result in Borda's positional method. This is a type of consensus ranking where you combine individual rankings from different "judges" to arrive at a consensus that is a "fair" combination of each ranking. The ranking algorithm is implemented as a plug-in that affected only the Integator component.

One issue with search results is how to characterize them. Some search engines attempt to remove duplicate results if one result can be determined to be a copy or mirror of the other. Another consideration is that results from the same site can be

grouped, or only the top-level page returned. What is displayed can also vary, from page title, URL, both, and with additional summary information.

For this project we dealt with the URLs of the returned results as follows. For two URLs to be considered the same the URLs returned from the different search engines had to match exactly. The following possibilities were not considered in the project:

- Two URLs with different yet similar paths that lead to the same webpage. For example a Google search on UCC Law returns *www.ucc.ie/law* but Yahoo returns *www.ucc.ie/en/lawsite*. Although the URLs both refer to the same page, they are considered distinct in our analysis.
- Two URLs that have completely different paths but still return the same page. For example University College Cork Blackboard has two access points *http://barra.ucc.ie* and *www.ucc.ie/Blackboard*.

This is an area where improvements could be made to remove more duplicates. The ranking data was collected as follows. For each query we collected the URLs of the first 100 results returned by each source search engine. The data was stored in a MySQL database table. The database tables are shown in Figure 3 below. These results were then filtered using SQL to select a final result set.

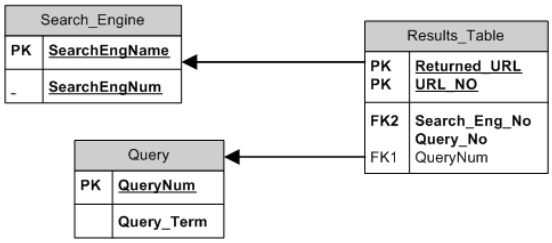


Figure 3: Database Tables for Rank Data

4 Conclusions

Open search adds flexibility in the design of powerful meta-search solutions. We developed a prototype search system as a test-bed for these ideas. This was based on the OpenSearch initiative and provides user customization features for selecting and weighting various sources. We also draw parallels between meta-search and aggregation technologies of which OpenSearch is a search-oriented example. We implemented a rank aggregation algorithm based on Borda's count method. We conclude that OpenSearch is a viable option for meta-search design especially as use of formats such as RSS and Atom becomes more widespread. The intersection of search and aggregation can lead to interesting technological solutions.

A deeper understanding of the similarity and variance of search engine ranking could feed into the design and development of more effective meta-search. Possible work

could look at the discrepancy in the results sets for different search engines in more detail via statistical comparison. One approach would be to systematically try queries and "deltas" of same to see how rank orderings change. A focus on different combination sets is another avenue of further research. In this scenario different search engines are combined systematically (single engine, pairs of engines, triples, all engines) and the results analysed to record changes to the rank orders. This type of experiment could answer questions such as: is there a use in combining the results of more than n (three, say) engines or is the effect negligible?

On the issue of open source search technology the cost and restrictions of proprietary systems will push companies and organizations to explore open search systems and experience reports provide evidence of technological possibilities.

Acknowledgements

I wound like to acknowledge the work of Colm Keane (Integrator component) and Cormac Debarra (Web interface) who worked on parts of the prototype system for their final year BSc Computer Science undergraduate projects at UCC.

References

[Bharat and Broder, 98] Bharat, K., Broder, A.: A Technique for Measuring the Relative Size and Overlap of Public Web Search Engines, In Proc. 7th WWW Conf., p. 379-388, Brisbane, Aus., April 1998.

[com Source, 07] com Source data: U.S. Expanded Search Queries by Search Engine, December 2007, com Source data, On Search Engine Watch Website, 2007. http://www.searchenginewatch.com

[Fagin *et al.*, 01] Fagin, R., Lotem, A., Naor, M.: Optimal Aggregation Algorithms for Middleware, In Proc. 20th ACM SIGACT-SIGMOD-SIGART Symp.: Principles of Database Systems, 2001.

[Google SOAP, 06] Google SOAP: Search Search SOAP API (now discontinued as of 2006). http://code.google.comapis/soapsearch/

[Google REST, 08] Google REST: Google Search API, 2008. http://code.google.com/apis/ajaxsearch/

[Harvard, 03] Harvard RSS 2.0 Specification: Hosted at Berkman Center for Internet & Society, Harvard Law School, 2003. http://cyber.law.harvard.edu/rss/rss.html

[IETF, 05] IETF Standard: The Atom Syndication Format, 2005. http://tools.ietf.org/html/rfc4287

[ISO/IEC, 03] ISO/IEC Standard 13250:2003: Topic Maps: Information Technology — Document Description and Markup Languages. http://www1.y12.doe.gov/capabilities/sgml/sc34/document/0322_files/iso13250-2nd-ed-v2.pdf

[Jansen *et al.*,07] Jansen, B., Spink, A., Koshman, S.: Web Searcher Interaction With the Dogpile.com Metasearch Engine, Journal of the American Society for Information Science and Technology, 58(5): 744-755, 2007.

[Khare *et al.*, 05] Khare, R., Cutting, D., Sitaker, K., Rifkin, A.: Nutch: A Flexible and Scalable Open-Source Web Search Engine, CommerceNet Labs Technical Report #04-0, May 10, 2005. http://labs.commerce.net/wiki/images/0/06/CN-TR-04-04.pdf

[LeVan, 06] LeVan, R.: OpenSearch and SRU: A Continuum of Searching, Information Technology and Libraries, Sept. 2006.

[Madhavan *et al.*, 06] Madhavan, J., Halevy, A., Cohen, S., Dong, X.L., Jeffery, S.R., Ko, D., Yu, C.: Structured Data Meets the Web: A Few Observations, Data Engineering Bulletin, 2006.

[Marchiori, 97] Marchiori, M.: The Quest for Correct Information on the Web: Hyper Search Engines, In 6th Int. WWW Conf., Santa Clara, C.A, p. 265-274, 1997.

[McCallum, 06] McCallum, S.: A Look at New Information Retrieval Protocols: SRU, OpenSearch/A9, CQL, and XQuery, World Library and Information Congress, 72th IFLA General Conf. (IFLA 2006), Seoul, 2006.

[MediaWiki, 06] MediaWiki OpenSearch Extension, version 0.2, 2006. http://www.mediawiki.org/wiki/Extension:OpenSearch

[MovableType, 05] Movable Type OpenSearch Description, 2005. http://hublog.hubmed.org/archives/001212.html

[mozDex] mozDex Search: http://www.mozdez.com

[Ng and Kantor, 98] Ng K.B, Kantor, P.B.: An Investigation of the Preconditions for Effective Data Fusion in IR: A Pilot Study, In Proc. 61th Annual Meeting of the American Society for Information Science, 1998.

[OpenSearch] OpenSearch: http://www.opensearch.org

[Saari, 95] Saari, D.G.: Basic Geometry of Voting, Springer-Verlag, 1995.

[Schatz *et al.*, 99] Schatz, B.R., Mischo, W., Cole, T., Bishop, A.P., Harum, S., Johnson, E.H., Neumann, L, Chen, H., Ng, T.D.: Federated Search of Scientific Literature, IEEE Computer 32(2), 51-60, 1999.

[Selberg and Etzioni, 95] Selberg, E., O. Etzioni, O.: Multi-Service Search and Comparison Using the MetaCrawler, In Proc. 4th Int. WWW Conf., 1995.

[W3C, 04] W3C Specification: Resource Description Framework, 2004. http://www.w3.org/RDF/

[YaCy] YaCy Search: http://yacy.net