# A Plugin Architecture Enabling Federated Search for Digital Libraries

Sergey Chernov, Christian Kohlschütter, and Wolfgang Nejdl

L3S Research Center, University of Hannover, Expo Plaza 1, 30539 Hannover, Germany,
{chernov,kohlschuetter,nejdl}@l3s.de

**Abstract.** Today, users expect a variety of digital libraries to be searchable from a single Web page. The German Vascoda project provides this service for dozens of information sources. Its ultimate goal is to provide search quality close to the ranking of a central database containing documents from all participating libraries. Currently, however, the Vascoda portal is based on a non-cooperative metasearch approach, where results from sources are merged randomly and ranking quality is sub-optimal. In this paper, we describe a Lucene-based plugin which replaces this method by a truly federated search across different search engines, where the exchange of document statistics improves document ranking. Preliminary evaluation results show ranking results equal to a centralized setup.

## 1 Introduction

Information integration over distributed sources is an urgent problem to be solved for providing access to a variety of Digital Libraries through a common search interface and portal. The German Vascoda project [15] tries to accomplish this ambitious task, integrating distributed scientific information resources from all over Germany. Its Web portal is the major German Website for providing unified access to interdisciplinary scientific and scholarly information. It comprises the Internet services from more than 40 academic libraries and institutions, and enables unified access to electronic full-text documents, document delivery services and pay-per-view options. The architecture is designed for expansion and new libraries can easily join Vascoda, thus positioning Vascoda as the main information source for the scholarly and research community in Germany as well as in Europe.

Currently, the Vascoda Information Portal provides a non-cooperative metasearch environment only[1]. Metasearch has inherent limitations for merging documents from heterogeneous sources into one consistent document ranking. As different search engines execute queries on different collections, document relevance is computed using diverse statistics, making results incomparable. It is thus virtually impossible to merge results into a single, consistent ranking without additional statistics about these collections.

Unfortunately, simply indexing all documents in a single search engine installation is not a useful option. While such a central approach would be technically feasible, taking content and collection restrictions for all participating libraries into account is

[1] From now on, by *metasearch* we refer to non-cooperative distributed search.

not an easy task, and sometimes plain impossible as some license contracts do not allow sharing data beyond the original library.

We therefore investigated a *federated* search infrastructure, where all partners can run their own (existing) search systems in a decentralized manner, but are still able to provide homogeneous search and ranking over all available collections. To achieve this, we provide a Lucene-based plugin for decentralized search engines, which is responsible for providing search across all collections and for exchanging document statistics between the connected sites.

In our project, we have explored this setup between two libraries with different search infrastructures, the German National Library of Science and Technology (TIB), and the Bielefeld University Library. TIB runs a search engine based on the open-source search engine library Apache Lucene [6, 8], Bielefeld uses an installation of FAST Data Search[2]). While both members want to integrate their systems in a federated search architecture within Vascoda, they need to keep their existing systems - completely replacing their search infrastructure is not an option. Besides designing and implementing a federated architecture for this setup, we ran experiments on the HU-Berlin EDOC, ArXiv and Citeseer document collections, split across the two participants, with promising results.

The paper is structured as follows: Section 2 introduces general issues of distributed information retrieval, relevant components for the federated search and the Lucene and FAST search engines. In Section 3 we discuss our Lucene-based plugin in more detail, and provide evaluation details in Section 4. Finally, we summarize and outline future work in Section 5.

## 2 Relevant Background

### 2.1 Distributed Information Retrieval

Good surveys on distributed information retrieval problems and solutions can be found in [2, 4, 5, 13, 16]. Here we only briefly review the underlying data model and main problems in distributed search.

**Vector Space Model**. For our federated search infrastructure we used the Vector Space Model [17]. It is effective, simple and well-known in the DL community. In this model, a document $D$ is represented by the vector $\boldsymbol{d} = (w_1, w_2, \ldots, w_m)$ where $m$ is the number of distinct terms and $w_i$ is the weight indicating the "importance" of term $t_i$. The weight is combined from the Term Frequency ($TF_i$) (the number of the term's occurrences in the document[3]) and Inverse Document Frequency ($IDF_i$) (the logarithm of the ratio $N/DF_i$, where $DF_i$ is the number of documents containing the term $t_i$, and $N$ is the overall number of documents in the collection). A common standard for term weighting is the $TF$x$IDF$ product and its variants.

**Metasearch vs. Federated Search**. In general, we can divide distributed search environments into two categories: uncooperative, isolated environments (metasearch) and cooperative, integrated environments (federated search). Metasearch participants have

[2] `http://www.base-search.net/`

[3] In information retrieval terminology, the term "frequency" is used as a synonym for "count".

no other access to the individual databases than a ranked list of document descriptions in the response to a query, while federated search participants have access to collection-wide statistics like $DF$ and $N$. As a consequence, the result rankings produced by metasearch are less homogeneous than using federated search.

**Search Broker.** The unifying unit in distributed information retrieval is a search broker, which provides access to several search engines. Figure 1 visualizes this unit in the distributed setup between TIB and Bielefeld University. Distributed search has much in common with search on a single source, but entails additional tasks (listed in [5][4]):

**Resource Description.** A full-text database provides information about its contents through a set of statistics, which is called *resource description*. Initially, the broker obtains and stores resource descriptions for every search engine. Every description is divided into *content summary*, *metadata summary* and *search engine configuration* [10]. The content summary is a limited set of statistics about each database/search engine. The metadata summary characterizes the metadata schema of resources and fields available for search. The search engine configuration finally represents the current state of customizable, search engine-wide parameters such as text pre-processing rules.

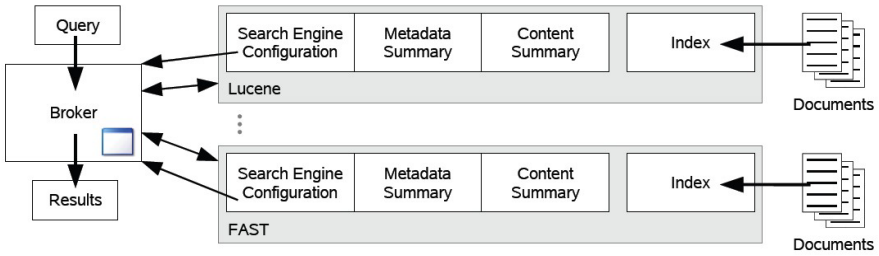


**Fig. 1.** Federated Search with Lucene and FAST Search Engines

**Database Selection.** The collected resource descriptions are used for database selection and query routing tasks. We want to select only those databases that are relevant to our query, according to their resource descriptions. The most natural way to do this is to ask the user to manually pick the set of interesting information sources. A better option is to automatically calculate a query-dependent "usefulness measure" of each database, based on the vector space model or language modeling approach [18].

**Query Mapping.** An important and difficult task is the handling of different document schemata and query languages. Some databases cover basic reference data only, while other contain enriched information with subject or classification information, abstract or additional description parts. The resource descriptions are helpful for a proper query mapping.

**Result Merging.** When a query is performed over several selected databases, one single ranking should be created integrating these results. Due to differences in search

[4] We enlarged this list by the "Query mapping task"

engine implementations and DL schemata, this is not an easy task. Document similarity/importance scores are not directly comparable among different collections and require a global normalization using collection-wide statistics. This requires a cooperative search environment, distributed search over non-cooperative DLs results in loss of search quality [1].

**Difficulties in Distributed Ranking.** Even in a cooperative environment, where all search engines provide all necessary statistics, it is hard to guarantee exactly the same ranking as that of the centralized database. The following factors may reduce search quality (we enlarged the list from [9]):

- Relevant documents are missed after the database selection step;
- Different stemmers and stopword lists influence both $TF$ and $DF$ values;
- Overlap affects globally computed $DF$ values;
- Query syntax may be incompatible;
- Unknown features of proprietary algorithms cannot be disabled;
- Document schemata (e.g., field names) on resources do not match.

### 2.2 Distributed Search in Digital Libraries

**Related Projects**. Several projects have attempted to improve distributed search in digital libraries. A recent effort is the DAFFODIL project [7], a system to support information search in distributed digital libraries. The project employs an advanced user interface for supporting search stratagems. While it also addresses the quality of ranking, it focuses on extended search functionality rather than the problems of distributed indexing and result merging. Another approach, pursued in the DILIGENT project [3], is to combine digital libraries using Grid technology. The project goal is to create a prototype Grid infrastructure for digital libraries. The project does not focus on the objectives of federated search. Commercial distributed search systems like Scirus[5] or Google Scholar[6] are interesting, but their implementation details are not publicly available.

**Protocols for DLs Interoperability**.

In federated search, brokers and search engines cooperate in order to deliver a common, consistent ranking, just as if search were performed by a single search engine. Factors like stemming algorithms, stop word lists, local collection statistics, ranking formulae etc. have to be unified across the federation. Combining information from many digital libraries thus requires a common standard interface for query and result exchange. Among the protocols like *Z39.50* [14], *OAI-PMH* [11, 12] and *SDARTS* [10], SDARTS is the most suitable one. It provides most of the Z39.50 functionality, but is much easier to support. The protocol keeps the common requirements for implementers to a minimum, while still supporting sophisticated features of advanced search engines.

**FAST and Lucene Search Engines**

The *FAST Data Search* engine is a commercial system by Fast Search & Transfer ASA (FAST). Its core consists of a full-text engine based on the Vector Space Model, extending it with additional features. The user can configure just a subset of all parameters. FAST plays an important role in the Vascoda project, since several members

[5] http://www.scirus.com/srsapp/

[6] http://scholar.google.com/

already use it for their local search service. *Lucene* [8] is also used as the basis for several search engines of VASCODA members. By itself, it is not a complete search engine, but rather a programming library which allows the easy creation of search engines with the desired properties and functionality. Lucene has been implemented in several programming languages, in this project we build upon its original (and mainline) implementation in Java.

## 3 Plugin Architecture

### 3.1 Integrating FAST Data Search and Lucene Search

While being very similar at some points, the two engines are incompatible on several levels, including index structures, API and ranking. Since FAST is a commercial, closed-source product, neither are all technical internals of FAST available nor are they exposed through the FAST API.

It is however possible to use intermediate results of the FAST pipeline for further computation, such that the FAST pipeline handles all crawling and text extraction tasks, while indexing and query evaluation for distributed search is handled by Lucene (local search is of course still handled by FAST in the FAST environment). Specifically, we take pre-processed plain-text documents, including metadata from both FAST and Lucene pipelines, and (re-)index them in a homogeneous manner, using a common lemmatization/stemming schema. This wrapper / *plugin* approach encapsulates the original search engine system and provides a common interface to the federation. Search is then performed over these plugins instead of the original vendor's system. Since Lucene is open-source and already provides distributed search facilities, we decided to implemented a Lucene-based plugin.

Given the modular nature of the FAST pipeline, (re-)indexing in the FAST context is easy. FAST Data Search provides a textual, XML-structured representation of indexed documents in its so-called FIXML format. Documents are described as field-structured plain text, which makes the transition to the Lucene indexing module quite easy. This plugin concept is extensible to other search engine products as well, as long as their processing pipelines are modular enough to provide the required intermediate results.

Figure 2 shows the resulting plugin-enriched federated search infrastructure. Before query time, all necessary statistics information (number of documents $N$ and document frequencies $DF_t$) is being collected on a regular basis and cached by the searcher (for each term $t$, all the plugins' $DF_t$ values are accumulated to one global $DF_t$). The search itself then proceeds as follows: the searcher parses the user's query into a Lucene-specific representation and transmits that to all (or a to a selected subset of the) available plugins. At query time, the plugins receive global $DF_t$ and $N$ statistics information from the searcher in order to rank their results accordingly. When the rank score has been determined, they call the searcher to *consume* the *hit*, i.e. Lucene-specific document ID plus rank score. This ID can then later be used to access document information, e.g. for the summaries on the results page.

Participating institutions may provide custom plugins for their own search engines. They can also connect their own user interfaces, with different layouts and functionality, to the federation.

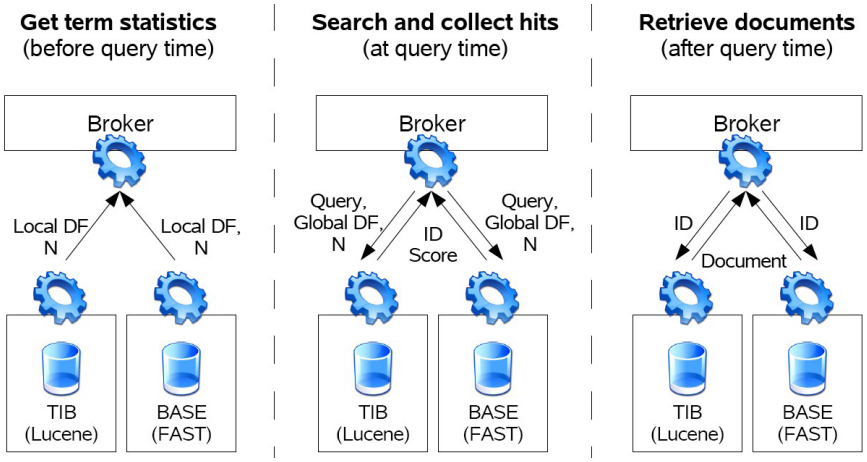


**Fig. 2.** Federated Search Infrastructure using the Lucene-based Plugin

### 3.2 Index and Search Capabilities

Our approach provides the advantages of centralized search (homogeneous index structure and ranking), while offering distributed search using standard Lucene features. In order to support this, the original document collection has only to be re-indexed by our Lucene-based plugin. This can be done automatically and incrementally by Lucene, so only added, deleted or changed documents are affected.

It is clear that the federation can only provide a common subset of all possible search features. While this set is actually quite large, our approach keeps the DLs' existing search engines untouched, so search outside the federation remains available.

The following capabilities are explicitly supported by our implementation:

- **Homogeneous Ranking**, based on the TF/IDF weighted vector-space model. Since the mapping from words to terms is homogeneous, the TF/IDF statistics can safely be used.
- **De-Duplication.** Since true duplicates yield the same rank score, they can easily be filtered out when creating the results page.
- **Database selection.** Query term-based database selection is easily performed by checking all plugin DF statistics before submitting the query.

### 3.3 Costs for Joining the Federation

An obvious question is what a digital library has to do when it wants to join the federation and how much it has to invest. In general, it only has to provide a fast Internet connection (since the majority of queries may be routed to all participants) and additional computational power. Clearly, these resources would be necessary for participating in any kind of federation. A new participant also has to provide storage space for the federated search index files: in our experiments this meant about a 30% increase for the additional index.

Compared to a homogeneous search engine architecture, the asset costs are manageable since they do not increase with the number of participants but with the number of different search systems. The costs of developing a plugin for a new search engine type can be shared among all members using it.

## 4 Experimental Evaluation

We have evaluated our prototype on several test collections and setups, as described in the next paragraphs.

### 4.1 Federation Setup

**Data**. Regarding collections, we used a document corpus based on the HU-Berlin EDOC document set and the ArXiv collection (metadata and full-text), together with the Citeseer OAI collection (metadata only). EDOC[7] is the institutional repository of Humboldt University, it encompasses about 2,500 annotated full-text documents of theses, dissertations, scientific publications and public readings. CiteSeer[8] is a system at Penn State University, USA, with the focus on literature in computer and information sciences. At the moment, the system contains more than 700,000 documents. The ArXiv preprint server[9] is located at Cornell University Libraries and provides access to more than 350,000 electronic documents in Physics, Mathematics, Computer Science and Quantitative Biology.

**Servers**. For the experiments, we have set up a small federation of two search engines, one in Bielefeld and one in Hannover. The servers communicate via a regular Internet connection. On each server, we have installed our plugin prototype and the corresponding Web front-end. The front-end only communicates to the local plugin, which then accesses local, disk-based Lucene document collections, or external, network-connected plugins, depending on the exact evaluation task. The server in Hannover was a Dual Intel Xeon 2.8 GHz machine, the server in Bielefeld was powered by a Dual AMD Opteron 250. On the Bielefeld server, a Lucene-plugin based version of the BASE (Bielefeld Academic Search Engine) Web user interface has been deployed, on the Hannover server, a fictitious "HASE" (Hannover Academic Search Engine) front-end also runs based on a Lucene-based search engine. Both GUIs, HASE and BASE are only connected to their local search plugins. The plugins then connect to the local collections (ArXiv and CiteSeer in Bielefeld and EDOC in Hannover) and to the plugin at the other site. This setup is depicted in Figure 3.

### 4.2 Hypotheses

Due to the underlying algorithms, we expected no difference in the result set (both item presence and position in the result list) between a centralized, combined index covering all collections and the distributed scenario proposed using Lucene-based Federated

[7] http://edoc.hu-berlin.de
[8] http://citeseer.ist.psu.edu/
[9] http://arxiv.org/

Search. Also, we expected almost no difference in search performance, since the collection statistics information is not exchanged at query time, but only at startup time and whenever updates in any of the included collections have occurred. Since the result display only lists a fixed number of results per page (usually 10 short descriptions, all of about the same length), the time necessary for retrieving the results lists from the contributing plugins is constant, i.e. not dependent on the number of found entries, but just on the servers' overall performance (CPU + network I/O).

### 4.3 Results

**Re-indexing**. Our current prototype re-indexed the metadata-only ArXiv collection (350,000 entries) in about 2 hours; the EDOC full-text document set (2,500 documents) was imported in 19 minutes. We expect that re-indexing will be even faster with a production–level implementation. The resulting index structure was about 20-40% of the size of the uncompressed FIXML data (depending on collection input) and also, interestingly, only about a third of the size of the original FAST index data. This is possibly due to extra data structures for the FAST engine, which are not necessary for our Lucene-based Federated Search. Based on these numbers, no significant overhead is induced by the necessary re-indexing step compared to a centralized index based on FAST Data Search.

**Search**. We performed several searches (one-word-, multi-word- and phrase queries) on our federation as well as on a local index, both using Lucene. We measured average query time and compared the rankings for one word, multi-word and phrase queries. Query times of a distributed setup were almost equal to the local setup, about 0.05-0.15 seconds per query, with an overhead of about 0.4 seconds for the distributed setup. The resulting ranking is equal to the one of a centralized setup.

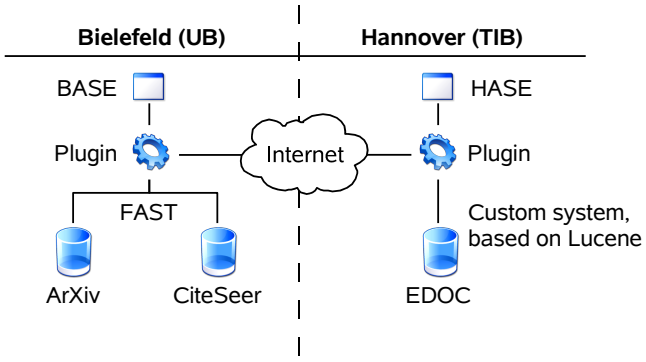


**Fig. 3.** Test Scenario for Federated Search

## 5 Conclusions

This paper describes ongoing work on the federated search infrastructure for digital libraries in the context of the Vascoda project, and discussed a Lucene-plugin based solutions for federated search, which provides high search quality even for decentralized collections and engines, easy integration of new members and preservation of existing systems and workflows.

A prototype based on the Lucene-plugin described in this paper integrates two search engine installations, one based on FAST Data Search at Bielefeld University Library and another Lucene-based search system at the German National Library of Science and Technology TIB. The federated system provides a final result ranking equal to search on a centralized database, while overall computational and maintenance costs remain reasonably low. The next major step will be to implement a stable version of the plugin for all collections of the current participants, and to make it publicly accessible.

## References


1. Wolf-Tilo Balke, Wolfgang Nejdl, Wolf Siberski, and Uwe Thaden. DL meets P2P – Distributed Document Retrieval based on Classification and Content. In *ECDL '05: Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries*, pages 379–390, 2005.
2. Jamie Callan. *Advances in information retrieval, W.B. Croft, editor*, chapter Distributed Information Retrieval, pages 127–150. 2000.
3. Donatella Castelli. DILIGENT: A Digital Library Infrastructure on Grid Enabled Technology. http://www.ercim.org/publication/ercim_news/enw59/castelli.html. *ERCIM News*, 59, 2004.
4. Nicholas Eric Craswell. *Methods for Distributed Information Retrieval. http://eprints.anu.edu.au/archive/00000503/.* PhD thesis, ANU, January 01 2001.
5. W. Bruce Croft. Combining Approaches to IR. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, 2000.
6. Doug Cutting et al. Lucene. http://lucene.apache.org.
7. Norbert Fuhr, Claus-Peter Klas, Andre Schaefer, and Peter Mutschke. Daffodil: An Integrated Desktop for Supporting High-Level Search Activities in Federated Digital Libraries. In *ECDL '02: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, pages 597–612, 2002.
8. Otis Gospodnetic and Erik Hatcher. *Lucene in Action*. Manning, 2005.
9. Luis Gravano, Kevin Chen-Chuan Chang, Hector Garcia-Molina, and Andreas Paepcke. STARTS: Stanford Proposal for Internet Meta-Searching. In *SIGMOD '97: Proceedings of the 1997 ACM International Conference on Management of Data*, pages 207–218, 1997.
10. Noah Green, Panagiotis G. Ipeirotis, and Luis Gravano. SDLIP + STARTS = SDARTS a Protocol and Toolkit for Metasearching. In *JCDL '01: Proceedings of the The First ACM and IEEE Joint Conference on Digital Libraries*, pages 207–214, 2001.
11. Carl Lagoze, Herbert Van de Sompel, Michael Nelson, and Simeon Warner. The Open Archives Initiative Protocol for Metadata Harvesting Protocol Version 2.0 of 2002-06-14. http://www.openarchives.org/oai/openarchivesprotocol.html.
12. X. Liu, K. Maly, M. Zubair, Q. Hong, M.L. Nelson, F. Knudson, and Holtkamp. Federated Searching Interface Techniques for Heterogeneous OAI Repositories. *Journal of Digital Information*, 4(2), 2002.

13. Weiyi Meng, Clement T. Yu, and King-Lup Liu. Building Efficient and Effective Metasearch Engines. http://doi.acm.org/10.1145/505282.505284. *ACM Comput. Surv.*, 34(1):48–89, 2002.
14. National Information Standards Organization. Z39.50: Application Service Definition and Protocol Specification, 2003.
15. Heike Neuroth and Tamara Pianos. VASCODA: A German Scientific Portal for Cross-Searching Distributed Digital Resource Collections. In *ECDL '03: Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries*, pages 257–262, 2003.
16. Tamar Sadeh. Google Scholar Versus Metasearch Systems. *High Energy Physics Libraries Webzine*, 12, 2006.
17. Gerard Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11):613–620, 1975.
18. Luo Si, Rong Jin, James P. Callan, and Paul Ogilvie. A Language Modeling Framework for Resource Selection and Results Merging. http://doi.acm.org/10.1145/584792.584856. In *CIKM '02: Proceedings of the ACM 11th Conference on Information and Knowledge Management*, pages 391–397, 2002.