Published in the 1st USENIX Conference on Web Application Development 2010

SeerSuite: Developing a scalable and reliable application framework for building digital libraries by crawling the web

Pradeep B. Teregowda
Pennsylvania State University

Isaac G. Councill
Google

Juan Pablo Fernández R.
Pennsylvania State University

Madian Khabsa
Pennsylvania State University

Shuyi Zheng
Pennsylvania State University

C. Lee Giles
Pennsylvania State University

Abstract

SeerSuite is a framework for scientific and academic digital libraries and search engines built by crawling scientific and academic documents from the web with a focus on providing reliable, robust services. In addition to full text indexing, SeerSuite supports autonomous citation indexing and automatically links references in research articles to facilitate navigation, analysis and evaluation. SeerSuite enables access to extensive document, citation, and author metadata by automatically extracting, storing and indexing metadata. SeerSuite also supports MyCiteSeer, a personal portal that allows users to monitor documents, store user queries, build document portfolios, and interact with the document metadata. We describe the design of SeerSuite and the deployment and usage of CiteSeer^x as an instance of SeerSuite.

1 Introduction

Efficient and reliable access to the vast scientific and scholarly publications on the web requires advanced citation index retrieval systems [18] such as CiteSeer [19, 36], CiteSeer^x [20], Google Scholar [6], ACM Portal [1], etc. The SeerSuite application framework provides an unique advanced and automatic citation index system that is usable and comprehensive, and provides efficient access to scientific publications. To realize these goals our design focuses on reliable, scalable and robust services.

A previous implementation, CiteSeer (maintained as of this date), was designed to support such services. However, CiteSeer was a research prototype and, as such, suffered serious limitations. SeerSuite was designed to provide a framework that would replace CiteSeer, to provide most of its functionality, but designed to be extensible. SeerSuite improves on several aspects of the original CiteSeer with features such as reliability, robustness and scalability. It does this by adopting a multi-tier architecture with a service orientation and a loose coupling of modules.

CiteSeer^x, an instance of SeerSuite is one of the top ranked resources on the web and indexes nearly one and half million documents. The collection spans computer and information science (CIS) and related areas such as mathematics, physics and statistics. CiteSeer^x acquires its documents primarily by automatically crawling authors web sites for academic and research documents. CiteSeer^x daily receives approximately two million hits and has more than two hundred thousand documents downloaded from its cache. The MyCiteSeer personal portal has over ten thousand registered users.

While the SeerSuite application framework has most of the functionality of CiteSeer, SeerSuite represents a complete redesign of CiteSeer. SeerSuite takes advantage of and includes state of the art machine learning methods and uses open source applications and modules. The structure and architecture of SeerSuite is designed to enhance the ease of maintenance and to reduce the cost of operations. SeerSuite is designed to run on off the shelf hardware infrastructure.

With CiteSeer^x, SeerSuite focuses primarily on CIS. However, there are requests for similar focused services in other fields such as chemistry [31] and archaeology [37]. SeerSuite can be adopted to providing services similar to those provided by CiteSeer^x in these areas. SeerSuite is a part of the project Chem$_X$Seer [32], a digital library search engine and collaboration service for chemistry. Though designed as a framework for CiteSeer^x-like digital libraries and search engines, the modular and extensible framework allows for applications that use SeerSuite components as stand alone services or as a part of other systems.

2 Background and Related Work

Domain specific repositories and digital library systems have been very popular over the last decades with several examples such as arXiv [21] for physics and RePEc [12] for economics. The Greenstone digital library [7] was developed with a similar goal. These repository systems, unlike CiteSeer^x, depend on manual submission of document metadata, a tedious and resource expensive process. As such CiteSeer^x which crawls authors web page for documents is in many ways a unique system, closest in design to Google Scholar.

The popularity of services provided by the original CiteSeer and limitations in its design and architecture were the motivation behind the design of SeerSuite [15, 28]. The design of SeerSuite was an incremental process involving users and researchers. User input was received in the form of feedback and feature requests. Exchange of ideas between researchers and users across the world through collaborations, reviews and discussions have made a significant contribution to the design. In addition to ensuring reliable scalable services, portability of the overall system and components was identified as an essential feature that would encourage adoption of SeerSuite elsewhere. During the process of designing the architecture of SeerSuite, other academic repository content management system (CMS) architectures such as Fedora [26] and DSpace [5] were studied. The Blackboard architecture pattern [33] had a strong influence on the design of the metadata extraction system. The main obstacles in adopting existing repository and CMS systems were the levels of customization and the effort required to meet SeerSuite design requirements. More specifically, implementation of the citation graph structure with a focus on automatic metadata extraction and workflow requirements for maintaining and updating citation graph structures made these approaches cumbersome to use.

In addition to repository, search engine and digital library architectures, advances in metadata extraction methods [22, 23, 16, 25, 30] and the availability of open source systems have influenced SeerSuite design. We begin our discussion of SeerSuite by describing the architecture.

3 Architecture

In the context of SeerSuite *reliability* refers to the ability of the framework instances to provide around the clock service with minimal downtime, *scalability* to the ability to support increasing number of user requests and documents in the collection, and *robustness* to the ability of the instance to continue providing services while some of the underlying services are unavailable or resource constrained.

An outline of SeerSuite architecture is shown in figure 1. By adopting a loosely coupled approach for modules and subsystem, we ensure that instances can be scaled and can provide robust service. We describe the overall architecture in the web application, data storage, metadata extraction and ingestion, crawling and maintenance sections.

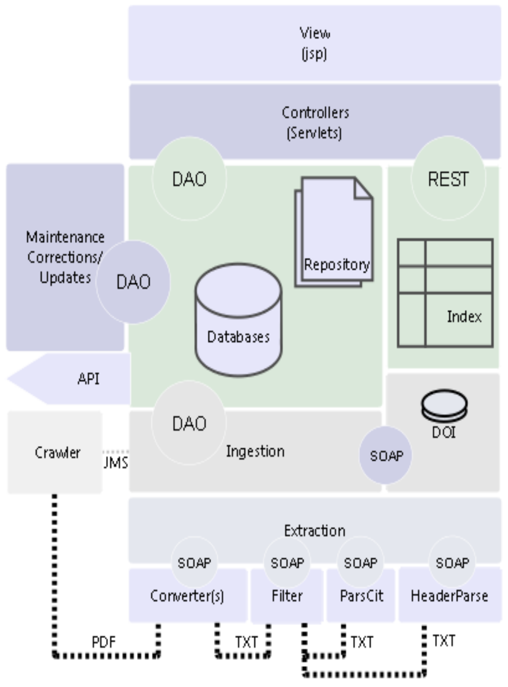


Figure 1: SeerSuite Architecture

3.1 Web application

The web application makes use of the model view controller architecture implemented with the Spring framework. The application presentation uses a mix of java server pages and javascript to generate the user interface. Design of the user interface allows the look and feel to be modified by switching Cascading Style Sheets (CSS). The use of JavaServer Pages Standard Tag Library (JSTL) supports template based construction. The web pages allow further interaction through the use of javascript frameworks. While the development environment is mostly Linux, components are developed with portability as a focus. Adoption of the spring framework allows development to concentrate on application design. The framework supports the application by handling interactions between the user and database in a transparent manner.

Servlets use Data Access Objects (DAO) with the support of the framework to interact with databases and the repository. The index, database and repository enable the data objects crawled from the web to be stored and accessed through the web, using efficient programmatic interfaces. The web application is deployed through a web application archive file.

3.2 Data storage

The databases store metadata and relationships in the form of tables providing transaction support, where transactions include adding, updating or deleting objects. The database design partitions the data storage across three main databases. This allows for growth in any one component to be handled by further partitioning the database horizontally.

The main database contains objects and is focused on transactions and version tracking of objects central to SeerSuite and digital libraries implementations. Objects stored in the main database include document metadata, author, citation, keywords, tags, and hub (URL). The tables in the main database are linked together by the document they appear in and are identified by the document object id.

One of the most unique aspects of SeerSuite is the citation graph. The nodes of this graph correspond to documents or citations and the edges to the relationships among them. This graph is stored in its own database. The citation relationship is stored in a graph table in the form of 'cited by' and 'cites' fields. In addition the database stores a canonical representation of citation or document metadata which is basically the most frequent and correct of the citations, as determined by a Bayesian network. These records serve as a grouping point for metadata collected about a particular document or citation. This database thus provides support for autonomous citation indexing and related features. The citation relationships are established by generating and matching keys for the document metadata records during ingestion or updates. The use of triggers allows data in the graph database to be updated when transactions such as insertions, deletions and corrections occur in the main database. In addition to triggers, application logic and maintenance functions maintain the link between the graph and the main database. MyCiteSeer personalization portal stores user information, queries and document portfolios in a separate database. The link between the user and the main database is through references in the tags in the MyCiteSeer database and the main database version tracking tables. A conventional RDBMS with support for triggers is used to host these databases.

The repository system provides SeerSuite with the ability, to provide cached copies of the documents crawled. In addition to the cached copy, the repository stores xml files containing extracted document metadata. These files serve as backup copies of the metadata stored in the databases. This is similar to the Fedora Object XML document representation [26]. The repository is organized into a directory tree. The top of the tree consists of a root folder containing sub directories mapped according to the segments in the document identifier. The final level contains metadata, text and cached files. The document identifier structure ensures that the there are a nominal number of sub folders under any folder in the tree structure.

An index provides a fast efficient method of storing and accessing text and metadata through the use of an inverted file. SeerSuite uses the Apache Solr an Index application [13] supported by Apache Tomcat to provide full text and metadata indexing operations. The metadata items are obtained from the database and the full text from the repository. The interaction of the application in the form of controllers is through the REST (Representational state transfer) [17] interface. This allows any indexing application supporting REST API's to be adopted by SeerSuite. In addition, this enables introduction of newer feature sets in the index or new versions of Solr without disruptions.

3.3 Metadata Extraction and Ingestion

Metadata extraction methods are built using Perl and C++. To enable interaction with the extraction services, a service oriented architecture is utilized using a Business Process Execution Language (BPEL) or, for convenience, scripts. Each component of the extraction system individually contributes to the final document, in this case an xml file, which is then ingested. The feedback to individual systems is manual adjusting of parameters or replacing components. The system can be rewired to include or exclude any extraction modules or applications.

The user can either batch process the incoming data or process each item individually. Addition of metadata into the system is controlled by the ingestion system, which interacts with main database using DAOs. The ingestion system ensures that essential metadata is correctly and uniquely labeled, with the help of an object identifier and checksum based de-duplication. In addition, the ingestion system makes use of listeners to share notification data such as alerts that inform users and programs about objects of interest.

The ingestion process can itself be distributed across machines, taking advantage of the Document Object Identifier (DOI), database and shared repository services. The DOI is issued by on of our web services with its own database and tracks document identifiers and the time of their issue.

3.4 Crawler

The suite also includes a Heritrix [8] based crawler for harvesting documents from the web. The interface between the ingestion system and the crawler is based on the Java Messaging Service over ActiveMQ [2]. Due to the modularity of the design, other crawlers can be used. The ingestion system sends URL submissions messages containing a job identifier and url through the ingestion channel, and the crawler responds with 'new content' messages pointing to the acquired resource and metadata to the ingestion system. The crawler can use other optional channels to provide status messages to listeners. The crawler uses the Heritrix job submission system which consumes messages in the submissions channel and processes these submissions.

3.5 Maintenance

Maintenance services support and enable associated functionality for the ingestion and presentation systems. Maintenance systems are responsible for updating the index, identifying metadata by inference, generating citation and document statistics, charts generation, and external data linking in the system. For convenience, common maintenance processes are available through a command line interface.

An evidence based inference system [16] utilizes a Bayesian inference framework to determine canonical metadata for documents in the collection. The system builds an ideal citation record for a document inferring from the information provided by citations and the document metadata.

Citation graphs which accompany the document view are generated from the graph database, by examining the citation relationships and the distribution across years. In addition to enabling access to extensive document metadata, SeerSuite allows documents in the collection to be linked to copies or bibliographic records in other collections. SeerSuite provides components to map and link to other services such as DBLP [4] and CiteULike [3].

The configuration of an SeerSuite instance or application is controlled through properties and context files, through which information about the file system for the repository, database, index and system parameters can be specified. The web application and the maintenance and indexing functions use similar configuration files. In addition the SeerSuite distribution provides configuration files or examples of configurations for applications used such as the Solr index and Tomcat.

4 Workflow

The outline of the process of adding documents to SeerSuite instances is shown in Figure 2. Documents are harvested from the web using focused crawlers (step 1). These documents are first converted from PDF or PostScript format to text with application tools such as PDFBox and TET or GhostScript for PostScript documents in the step labelled 2.

In step 3, to prevent processing of documents such as resumes and non-academic documents part of the harvested collection, SeerSuite uses a regular expression based filter on the converted text file. The converted, filtered text is processed using state of the art automatic metadata extraction systems. These include the Support Vector Machine based header parser [22], which extracts metadata such as titles, publication information, authors and their affiliation, and abstract from the document. The ParsCit [25] citation extraction system extracts citation and context information from the document.

The ingestion system identifies unique documents and requests a document identifier for the document. If the document is found to be a checksum duplicate based on content, URL mappings are updated to include alternate URL(s) in step 4. In the same step, the repository is updated with a complete set of files including the document in the original format. The converted text files, citations, crawler and document metadata are placed in the relevant document directory under the repository tree. In addition to the individual metadata files, a file copy of the complete document metadata is stored in the form of an xml file. With updates and corrections, the xml files are updated and stored with a version tag. In the main database papers, authors citation and url mapping tables are updated, triggering updates to the graph database. Updates to the graph database ensure that the citation relationship of the incoming metadata to the data already existing in the collection is accurately maintained.

Documents in the database have a time stamp indicating time of update, helping the maintenance scripts perform incremental updates of the index. Incremental updates are crucial in reducing the time required for maintenance. With step 5, new or updated metadata are indexed. The maintenance script scans the database for updated and new documents and creates an in-memory indexable document, including the document metadata fields and citation information gathered across the main and citation graph databases. This document is then indexed by the main Solr index over the REST interface with an update command. The maintenance system optimizes the index after each update, using an built in feature in Solr.

Statistics provide users with a perspective of the collection from a citation, document and author ranking using aggregated citation information. Statistics are generated from the graph database, in the form of text files, which are then presented to the user through the statistics servlet interface. Maintenance scripts are manually scheduled or run by the administrator.

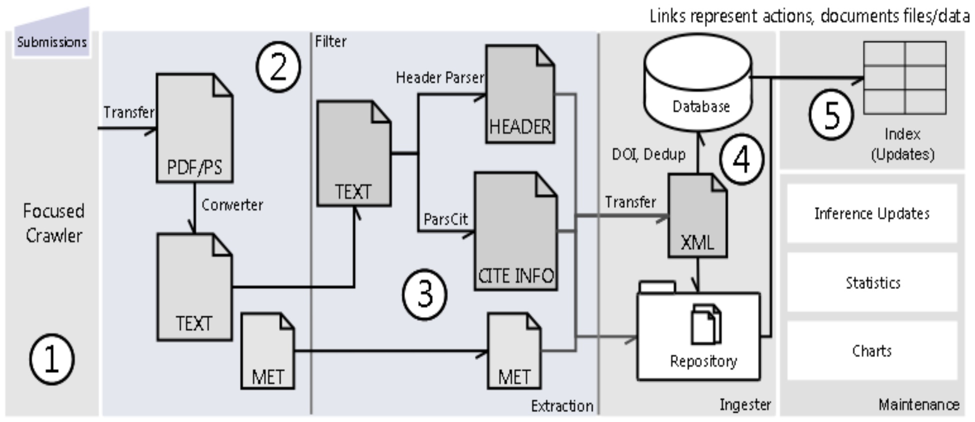


Figure 2: SeerSuite Workflow: Links represent actions and documents data

5 An Instance of SeerSuite: CiteSeer^x

CiteSeer^x serves as a flagship deployment of SeerSuite. It utilizes Apache Tomcat as the supporting platform with MySQL as the RDBMS. The deployment spans multiple machines with varied configurations. Components are distributed based on functionality on these machines. A pair of level 4 load balancing servers direct traffic to a pair of webservers. The load balancers use a connection based metric to determine to which server a particular request will be directed. To ensure availability, the load balancers are configured as a high availability asymmetric cluster that uses open source linux high availability software.

The web servers host the application on the Apache Tomcat platform with the Tomcat instances as part of an Apache Tomcat TCP cluster with session information shared across the cluster. The application processes these requests and processes them with the help of the index, database or the repository.

In case of an search query, the application translates the query to a suitable format and dispatches the query to the Solr indices. The results are processed and presented to the user. CiteSeer^x uses indices for document and citation objects, table objects and disambiguated author objects. Requests for metadata are handled by the MySQL database system containing the main graph and the user databases. The repository is responsible for storing cached documents and the text and metadata files. Requests for cached files are handled by the application with support from the repository stored on a storage server. The repository system is shared with the ingestion system and web servers, using the Global File System within the cluster.

The processing system is maintained separately from the web application and data storage infrastructure. The document processing systems are responsible for converting and extracting the metadata from converted text files. This operation is distributed across machines by dividing the incoming data into distinct sets, each set being processed by individual machines across the cluster.

The deployment is supported by a staging and development system, where new features are introduced and reviewed before being introduced in the production system. Major components in the system are backed up either using component level replication services and or by file level backups. In addition to backup on site, off site backups are utilized to ensure redundancy.

CiteSeer^x depends on operating system based security and application security provided by firewalls plus intrusion detection systems and the underlying framework. Application logs and Tomcat error and access logging provide audit trails for bug fixes and troubleshooting.

Infrastructure adopted by CiteSeer^x has led to several issues. Frequent freezes occur due to deadlocks involving the shared file system. Hardware failures have led to loss of data across the repository database. In such cases back ups have helped restore services. The ability to recover from these unfortunate losses showcase CiteSeer^x robustness.

5.1 Focused Crawling

For the initial crawling seeds, CiteSeer^x assimilated the complete collection of documents and their URLs from CiteSeer with some exceptions. These documents were ingested into the system by utilizing the already existing information in the CiteSeer databases.

Though equipped with Heritrix, the new CiteSeer^x uses a customized focused crawler, which runs incrementally. The crawler is run on a daily basis and can fetch several thousand new documents every day. The system maintains a list of parent URLs where documents were previously found. The parent URL's include aca-

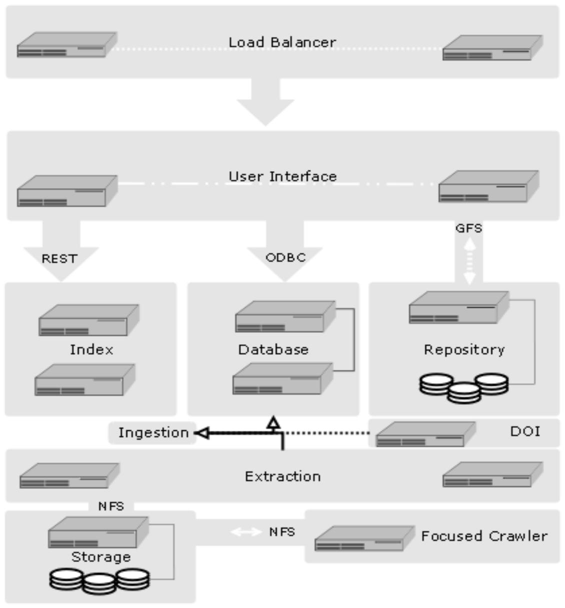


Figure 3: CiteSeer^x Deployment

demic homepages containing lists of online publications.

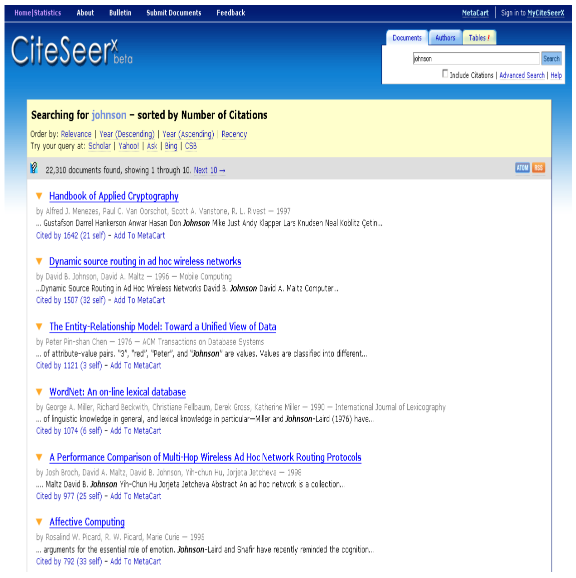
CiteSeer^x has nearly two hundred thousand unique URLs containing links to publications. The crawl process begins with the crawl scheduler, which selects a thousand parent URLs based on an estimated likelihood of these pages having new documents. The selected URLs are fed to a seed crawler. The seed crawler revisits these thousand URLs and also follow links up to two hops from each scheduled URL. By parsing fetched pages, the seed crawler will retrieve all document links eligible for processing. These represent documents which will be later downloaded. If a document link is found on a new URL, this URL will be added to the parent URL list. This enables the parent URL list to expand. The crawling system also maintains a list of document URLs and a list of document checksum hash values for all previously crawled documents. These two lists help avoid downloading duplicate documents. When the seed crawler outputs a list of discovered document URLs, it first compares them to the maintained document URL list and filters out duplicate URLs. Therefore, documents with same URLs will not be downloaded again. Only new document URLs will be fed to the document crawler.

The document crawler then simply fetches all documents in PDF or PostScript from the input list. After the documents are crawled, their checksum values are calculated based on their content and compared to the maintained checksum list by the ingestion system. Thus, content based duplicates are removed. Finally, only new documents are ingested into CiteSeer^x. User submissions to CiteSeer^x are directly submitted to the crawler seed listing. A user interface allows tracking of the documents obtained from a submitted URL. This allows the user to determine, whether a document has been ingested into CiteSeer^x.

6 User Interface

A user interacts with SeerSuite through a number of web pages. The most common of these include results of searches, document details, and citation graphs. We describe a set of pages, whose coverage spans most of the functionality provided by SeerSuite. The user interfaces are built using jsp, backed by the controller servlets. A navigation panel allows the user quick access to the main pages. The access control to pages across the application is based on the login system provided as part of MyCiteSeer component.

6.1 Search Interface



Searching for johnson – sorted by Number of Citations

Order by: Relevance | Year (Descending) | Year (Ascending) | Recency
Try your query at: Scholar | Yahoo! | Ask | Bing | CSB

22,310 documents found, showing 1 through 10. Next 10 →

Handbook of Applied Cryptography
by Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, R. L. Rivest — 1997
... Gustafson Darrel Hankerson Anwar Hasan Don ***Johnson*** Mike Just Andy Klapper Lars Knudsen Neal Koblitz Çetin...
Cited by 1642 (21 self) - Add To MetaCart

Dynamic source routing in ad hoc wireless networks
by David B. Johnson, David A. Maltz — 1996 — Mobile Computing
...Dynamic Source Routing in Ad Hoc Wireless Networks David B. ***Johnson*** David A. Maltz Computer...
Cited by 1507 (32 self) - Add To MetaCart

The Entity-Relationship Model: Toward a Unified View of Data
by Peter Pin-shan Chen — 1976 — ACM Transactions on Database Systems
... of attribute-value pairs. "3", "red", "Peter", and "***Johnson***" are values. Values are classified into different...
Cited by 1121 (3 self) - Add To MetaCart

WordNet: An on-line lexical database
by George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, Katherine Miller — 1990 — International Journal of Lexicography
... of linguistic knowledge in general, and lexical knowledge in particular—Miller and ***Johnson***-Laird (1976) have...
Cited by 1074 (6 self) - Add To MetaCart

A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols
by Josh Broch, David A. Maltz, David B. Johnson, Yih-chun Hu, Jorjeta Jetcheva — 1998
.... Maltz David B. ***Johnson*** Yih-Chun Hu Jorjeta Jetcheva Abstract An ad hoc network is a collection...
Cited by 977 (25 self) - Add To MetaCart

Affective Computing
by Rosalind W. Picard, R. W. Picard, Marie Curie — 1995
... arguments for the essential role of emotion. ***Johnson***-Laird and Shafir have recently reminded the cognition...
Cited by 792 (33 self) - Add To MetaCart

Figure 4: Document Search Results

The search interface in SeerSuite allows users to make queries across document, author, and table metadata either by using the default query interface or an advanced search interface. Search results are generated as a result of a user query. The results can be sorted according to the relevance, citations, and recency of the document ingestion. Since SeerSuite indexes document metadata across entities such as title, authors and their affiliation, year of publication, publication venue, keywords and abstract, the user can make queries which span one or more document or citation entities through the advanced query interface.

Each search result includes the title, author names, publication information and a short snippet of the text containing the query terms. A javascript interface enables users to view the abstract of a document from a mouse over the down arrow associated with each search result. The process and content of the search results are unique and can be ranked based on citations, relevance and recency.

SeerSuite applications use a Solr instance configured to index metadata fields defined by SeerSuite across documents in the collection. SeerSuite uses a simple ranking algorithm based on the default ranking algorithm provided by Solr for ranking results. These results are boosted based on the location of where the results were found and citations, e.g. a result containing the query term in the title or author names have higher rank. New ranking methods are readily implemented in Solr.

Relevance based searches are used by default in SeerSuite. The ability to rank documents based on the number of citations is an optional default.

SeerSuite uses author normalization for the metadata extracted from the documents to provide a comprehensive set of variations of the author names. The normalization allows authors to be searched for through name variants.

In addition to viewing results on the search results page, the user can subscribe to search feeds, which update the user when new results are available for a particular query.

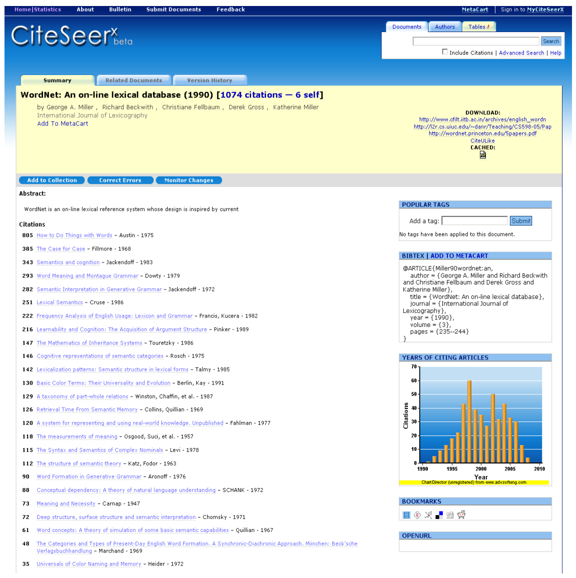
6.2 Document Summary

The document view in figure 5, provides the user a view of metadata information aggregated for each document. This view contains extensive metadata about a individual document including - title, author names, venue of publication, year of publication and the citations contained in the document. The tabbed interface allows the user to browse citation relationships and the metadata version information. Link to the source of the document along with the option to download a cached copy are provided. In case the document is found at multiple URLs, the page lists all the alternate document URLs. The citation linking information provided along with the document includes links to documents cited by this document. The documents cited by the document are ranked by how well they are cited in the collection. A graph illustrating the citations to the document across years which can be useful for identifying trends and impact is also displayed.

The document summary page provides several MyCiteSeer interaction points. Add to Collection, Correct Errors and Monitor changes links allow the user to insert documents to his collection, correct metadata and monitor the document.

Corrections to the metadata involve several changes to the document metadata and citation relationships. SeerSuite examines these relationships, updating the citation graphs as necessary. It either creates or updates the citation cluster established for the current document after the corrections are submitted. A versioning system enables the administrator to track changes to document metadata. A user can view the modifications to the document through the versions tab, which displays all versions of the document metadata and the attribution for each update or correction.

The document view page contains data spanning multiple databases and the generation of this page is resource intensive. In addition to providing metadata, the docu-



WordNet: An on-line lexical database (1990) [1074 citations — 6 self]

by George A. Miller , Richard Beckwith , Christiane Fellbaum , Derek Gross , Katherine Miller
International Journal of Lexicography
Add To MetaCart

DOWNLOAD:
http://www.cfilt.iitb.ac.in/archives/english_wordn
http://l2r.cs.uiuc.edu/~danr/Teaching/CS598-05/Pap
http://wordnet.princeton.edu/5papers.pdf
CiteULike
CACHED:

Abstract:

WordNet is an on-line lexical reference system whose design is inspired by current

Citations

805 How to Do Things with Words – Austin - 1975
385 The Case for Case – Fillmore - 1968
343 Semantics and cognition – Jackendoff - 1983
293 Word Meaning and Montague Grammar – Dowty - 1979
282 Semantic Interpretation in Generative Grammar – Jackendoff - 1972
251 Lexical Semantics – Cruse - 1986
222 Frequency Analysis of English Usage: Lexicon and Grammar – Francis, Kucera - 1982
216 Learnability and Cognition: The Acquisition of Argument Structure – Pinker - 1989
147 The Mathematics of Inheritance Systems – Touretzky - 1986
146 Cognitive representations of semantic categories – Rosch - 1975
142 Lexicalization patterns: Semantic structure in lexical forms – Talmy - 1985
130 Basic Color Terms: Their Universality and Evolution – Berlin, Kay - 1991
129 A taxonomy of part-whole relations – Winston, Chaffin, et al. - 1987
126 Retrieval Time From Semantic Memory – Collins, Quillian - 1969
120 A system for representing and using real-world knowledge. Unpublished – Fahlman - 1977
118 The measurements of meaning – Osgood, Suci, et al. - 1957
115 The Syntax and Semantics of Complex Nominals – Levi - 1978
112 The structure of semantic theory – Katz, Fodor - 1963
90 Word Formation in Generative Grammar – Aronoff - 1976
88 Conceptual dependency: A theory of natural language understanding – SCHANK - 1972
73 Meaning and Necessity – Carnap - 1947
72 Deep structure, surface structure and semantic interpretation – Chomsky - 1971
61 Word concepts: A theory of simulation of some basic semantic capabilities – Quillian - 1967
48 The Categories and Types of Present-Day English Word Formation. A Synchronic-Diachronic Approach. Miinchen: Beck'sche Verlagsbuchhandlung – Marchand - 1969
35 Universals of Color Naming and Memory – Heider - 1972

No tags have been applied to this document.

```
@ARTICLE{Miller90wordnet:an,
   author = {George A. Miller and Richard Beckwith
and Christiane Fellbaum and Derek Gross and
Katherine Miller},
   title = {WordNet: An on-line lexical database},
   journal = {International Journal of
Lexicography},
   year = {1990},
   volume = {3},
   pages = {235--244}
}
```

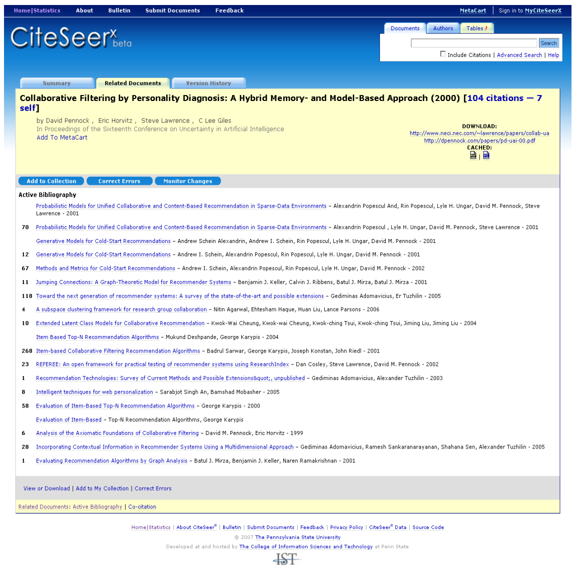
Figure 5: Document Summary

ment also allows the user to download the bibtex of the article or collect the bibtex of the article in a meta cart for download later. The page also provides several bookmarking links and the ability to copy document data provided on the page using browser plugins.

6.3 Citation Relationships

The citation graph generated and stored as part of the ingestion process and updated as part of the maintenance process allows SeerSuite to provide users with tools for citation analysis. Citation based relationships such as active bibliography and co-citations are available for each document through the related documents tab in the document summary page. This relationship provides valuable information to users, allowing users to track documents of their interest. Such analysis is helpful in exploring topics and literature surveys. Active Bibliography provides links between documents citing the same set of documents and is one way grouping of documents. Another method is by identifying Co-citations which are links between documents which cite the same documents to a particular document. Figure 6 shows the Active Bibliography of a document in CiteSeer^x.

The citation graph is dynamic, changing as a result of corrections and other metadata updates. SeerSuite indexes the citation relationships with the document. Therefore, in addition to the database, rendering citation relationships such as active bibliography and co-citations requires queries to the database followed by queries to the index. The listing of each citation in the relationship is based on a similarity document measure implemented at the index. The links utilize the cluster ID, which is mapped to the document in case the document is available in the collection. In case where the document is not available, it points to the index listing of documents citing the citation entity. The citation relationships and ranking of authors, documents and citations are also summarized in a detailed year by year list in the statistics pages.



Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach (2000) [104 citations — 7 self]

by David Pennock , Eric Horvitz , Steve Lawrence , C Lee Giles
In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence
Add To MetaCart

DOWNLOAD:
http://www.neci.nec.com/~lawrence/papers/collab-ua
http://dpennock.com/papers/pd-uai-00.pdf
CACHED:

Active Bibliography

	Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments – Alexandrin Popescul And, Rin Popescul, Lyle H. Ungar, David M. Pennock, Steve Lawrence - 2001
70	Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments – Alexandrin Popescul , Lyle H. Ungar, David M. Pennock, Steve Lawrence - 2001
	Generative Models for Cold-Start Recommendations – Andrew Schein Alexandrin, Andrew I. Schein, Rin Popescul, Lyle H. Ungar, David M. Pennock - 2001
12	Generative Models for Cold-Start Recommendations – Andrew I. Schein, Alexandrin Popescul, Rin Popescul, Lyle H. Ungar, David M. Pennock - 2001
67	Methods and Metrics for Cold-Start Recommendations – Andrew I. Schein, Alexandrin Popescul, Rin Popescul, Lyle H. Ungar, David M. Pennock - 2002
11	Jumping Connections: A Graph-Theoretic Model for Recommender Systems – Benjamin J. Keller, Calvin J. Ribbens, Batul J. Mirza, Batul J. Mirza - 2001
118	Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions – Gediminas Adomavicius, Er Tuzhilin - 2005
4	A subspace clustering framework for research group collaboration – Nitin Agarwal, Ehtesham Haque, Huan Liu, Lance Parsons - 2006
10	Extended Latent Class Models for Collaborative Recommendation – Kwok-Wai Cheung, Kwok-wai Cheung, Kwok-ching Tsui, Kwok-ching Tsui, Jiming Liu, Jiming Liu - 2004
	Item Based Top-N Recommendation Algorithms – Mukund Deshpande, George Karypis - 2004
268	Item-based Collaborative Filtering Recommendation Algorithms – Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl - 2001
23	REFEREE: An open framework for practical testing of recommender systems using ResearchIndex – Dan Cosley, Steve Lawrence, David M. Pennock - 2002
1	Recommendation Technologies: Survey of Current Methods and Possible Extensions&quot;, unpublished – Gediminas Adomavicius, Alexander Tuzhilin - 2003
8	Intelligent techniques for web personalization – Sarabjot Singh An, Bamshad Mobasher - 2005
58	Evaluation of Item-Based Top-N Recommendation Algorithms – George Karypis - 2000
	Evaluation of Item-Based – Top-N Recommendation Algorithms, George Karypis
6	Analysis of the Axiomatic Foundations of Collaborative Filtering – David M. Pennock, Eric Horvitz - 1999
28	Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach – Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, Alexander Tuzhilin - 2005
1	Evaluating Recommendation Algorithms by Graph Analysis – Batul J. Mirza, Benjamin J. Keller, Naren Ramakrishnan - 2001

View or Download | Add to My Collection | Correct Errors

Related Documents: Active Bibliography | Co-citation

Home|Statistics | About CiteSeer^x | Bulletin | Submit Documents | Feedback | Privacy Policy | CiteSeer^x Data | Source Code
© 2007 The Pennsylvania State University
Developed at and hosted by The College of Information Sciences and Technology at Penn State

Figure 6: Active Bibliography

7 MyCiteSeer

SeerSuite aims to provide the user with services which improve the efficiency of the user in accessing information. MyCiteSeer plays a crucial role in providing and supporting these services. MyCiteSeer allows users to store queries, document portfolios, tag documents, and monitor and track documents of interest. The portal space is available after user registration and login. We briefly describe the user interface of MyCiteSeer.

Figure 7 shows the index page for MyCiteSeer. The index page serves as the landing page, with the menu providing links to other pages including the profile, collections tags and monitoring pages.

The profile page presents the user with interfaces to update information stored as part of his profile on MyCiteSeer including the password for the account. In the case where API support has been enabled, the profile page also allows the user to request an API key.

The collections page allows the user to view collections of documents stored within the account. These collections are user defined sets of documents, aggregated under their profile for ease of access. The user can use a collection to download bibliographic data for all documents in a collection.

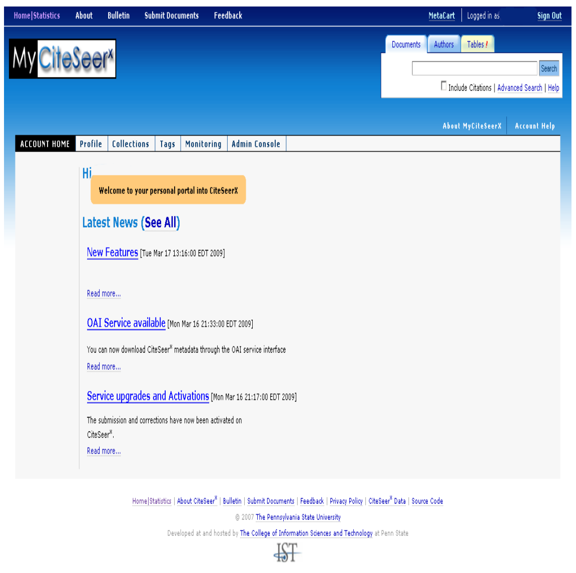
Tags provide the user with a listing of tags defined by the user and link to the documents tagged with that tag. The tag portal page allows the user to view and delete tags defined by the user and the documents these tags are linked to.

The monitoring page allows users to track changes to a document in the SeerSuite collection. Any updates to document metadata, including the citation graph linked with the document for documents in the monitored collection are sent to the user through e-mail registered with the system.

In addition to providing the user with a portal, MyCiteSeer enables SeerSuite to utilize crowd sourcing or distributed error correction [27] for corrections to document metadata. By assessing weights based on prior corrections, the evidence based system can detect malicious changes.

The application program interface component utilizes MyCiteSeer user data for generating the access key and controlling access to services provided. An user marked as an administrator has additional functionality available to him through MyCiteSeer. A subset of configuration and administrative interfaces are made available through an admin console.

The portal framework shares the structure and storage with the main application. While the servlets for MyCiteSeer are developed with SeerSuite in mind, the interaction with SeerSuite applications can easily be extended to other projects and services. The MyCiteSeer component interacts with SeerSuite ingestion and maintenance modules, using listeners. The maintenance and ingestion service provide notifications on objects being updated or processed through these listeners.


Home|Statistics About Bulletin Submit Documents Feedback
MetaCart | Logged in as Sign Out
MyCiteSeerX
Documents Authors Tables !
Search
Include Citations | Advanced Search | Help
About MyCiteSeerX Account Help
ACCOUNT HOME Profile Collections Tags Monitoring Admin Console

Hi

Welcome to your personal portal into CiteSeerX

Latest News (See All)

New Features [Tue Mar 17 13:16:00 EDT 2009]

Read more...

OAI Service available [Mon Mar 16 21:33:00 EDT 2009]

You can now download CiteSeer^x metadata through the OAI service interface

Read more...

Service upgrades and Activations [Mon Mar 16 21:17:00 EDT 2009]

The submission and corrections have now been activated on CiteSeer^x.

Read more...

Home|Statistics | About CiteSeerX | Bulletin | Submit Documents | Feedback | Privacy Policy | CiteSeerX Data | Source Code
© 2007 The Pennsylvania State University
Developed at and hosted by The College of Information Sciences and Technology at Penn State
IST

Figure 7: MyCiteSeer^x Index Page

8 Other Interfaces

8.1 OAI

The Open Archives Initiative (OAI) provides an efficient protocol metadata dissemination framework for data sources such as a SeerSuite. A low barrier mechanism, OAI is particularly suitable for SeerSuite applications, enabling instances that provide metadata sharing, publishing and archiving. SeerSuite supports interfaces compliant to the OAI-Protocol for metadata harvesting (PMH) [11] previously established with CiteSeer [35].

In the earlier CiteSeer system, modified CGI scripts were utilized for handling queries and generating compliant content for the OAI. In contrast, requests made to the SeerSuite OAI interface are handled by servlets, which translate requests into data access calls. The servlets assemble results in an OAI compliant manner, which is presented to the client. SeerSuite supports the full complement of OAI-PMH version 2 verb requests and provides content in the Dublin Core format. Thus, all document metadata is accessible to a client through the OAI interface.

8.2 API

Application Programmable Interfaces (APIs) are central for programmatic access to the repository. Through REST [17] web services, SeerSuite supports the needs of both programmers aiming to access metadata from the digital library and software agents looking to exchange information between digital libraries. SeerSuite API is a revamped version of the previously developed CiteSeer API [34] which was SOAP/WSDL-based.

The goal of the SeerSuite API is to share metadata with developers and software agents. Moving to a REST-based web service from a SOAP/WSDL-based one reduces the size of requests and the responses exchanged between the client and the server. Hence, developers can retrieve information faster, and total network traffic is reduced. The version of the API deployed in CiteSeer^x provides access to the papers, authors, citations, keywords, and citation contexts. The API caller may provide a SQL-like query to be executed as a filter on the matching set. The resource URI formats are shown in table 1. Objects are identified by document IDs (*docid*), author IDs (*aid*) and citation IDs (*cid*). SeerSuite API can output the results in both XML and JSON formats depending on the callers preference.

Type	URI Format
Paper	`http://host/papers/[docid]`
Author	`http://host/authors/[aid]`
Citation	`http://host/authors/[cid]`

Table 1: CiteSeer^x API Resource URI Formats

SeerSuite adopts Jersey [9] as a library to build the RESTful web service, which in turn implements the JAX-RS [10] reference. SeerSuite requires users looking to use the API to have a valid MyCiteSeer account. Account information in MyCiteSeer is used to generate an Application ID (*appid*) which has to be passed in every HTTP request as a mechanism of authentication. Daily limits for users are monitored and can be managed by administrators for performance.

9 Federation of Services

CiteSeer^x includes several unique services, which are not part of the SeerSuite application framework. Provisioning for these services is an unique aspect of the SeerSuite framework. Many of these services have evolved as a result of research and are still being developed. The developer builds and operates these services independently, sharing hosting infrastructure with the main application. Separate tables and databases and index operations maybe provisioned for each service. In the following sections, we briefly discuss Table search and author disambiguation search.

9.1 Table Search

Tables in documents often contain important data not present elsewhere. Table search services are based on TableSeer developed as part of the project [29]. Table search automatically extracts tables metadata, and indexes and ranks tables present in a SeerSuite collection. While table search shares components of the web application and shares the repository with SeerSuite, the index and extraction components are independent of SeerSuite.

The SeerSuite interface utilizes the main application framework for interaction with the table index. The queries results from the index are again processed and presented by the main application framework. Independent operation of the index from the main index allows for more efficient query processing and ranking of table search results. The results utilize the SeerSuite file system infrastructure view the result of particular pages of the tables in cached documents. The ingestion, maintenance and updates services for Table search are independent of SeerSuite, allowing for flexibility in research and development. Some aspects of the table search ingestion system require access to the document metadata such as title, author not extracted as part of Table extraction, which are acquired from the main SeerSuite instance metadata.

Table search has served as a template for the development of similar services such as algorithm and figure search, which are in development.

9.2 Author Disambiguation

Author disambiguation enables users to identify whether records of publications in a SeerSuite collection refer to the same person. The author disambiguation service provided by SeerSuite is based on an efficient integrative framework for solving the name disambiguation problem. A blocking method retrieves candidate classes of authors with similar names and a clustering method, DBSCAN, clusters papers by author. The distance metric between papers used in DBSCAN is calculated by an online active selection Support Vector Machine algorithm(LASVM) [24]. This system has been utilized in CiteSeer^x. The disambiguation application identifies distinct authors based on header information which includes author affiliation and co-authorship.

The implementation makes use of already existing author object data in the main database, and generates cluster IDs for disambiguated authors that are stored in a separate table and index. Results for disambiguated author queries are handled by main application framework by interacting with the main database and the index. A profile page exists for each disambiguated author, with author affiliation, impact, and publications garnered from the SeerSuite instance. The profile page also provides a link, if available, to the author homepage obtained through a system HomePageSeer. An incremental algorithm replacing the offline batch algorithm currently used is in development.

10 Usage

CiteSeer^x receives nearly two million requests from across the globe. A significant portion of this traffic is as a result of document views, downloads and searches. An analysis of access logs is presented in this section.

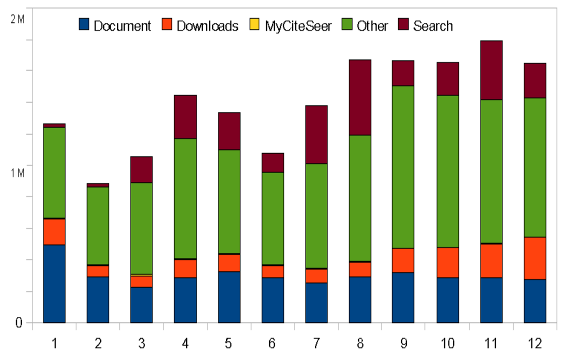


Figure 8: CiteSeer^x Traffic in 2009

The graph in figure 8 shows average hits per month for CiteSeer^x during the year 2009. The search grouping includes requests for document and author search. The ’other’ grouping includes queries for citing documents, legacy mappings (redirects from CiteSeer), OAI, and requests to author profile and static pages. Document related requests include downloads and summary views. The graph indicates a growth in the number of hits, driven by downloads, views of citation relationships, and search. The number of document views have grown by a lesser margin. During this time, the collection of documents in CiteSeer^x grew by 200,000 documents with updates to document metadata through corrections.

The majority of the referrals to CiteSeer^x are through pages hosted on CiteSeer^x (67 %). A number of users (29 %) arrive without a referrer,(i.e., users landing directly on CiteSeer^x or requests by crawlers). Redirection from CiteSeer contribute (1 %); references from Google, Google Scholar (2 %,1 %), Yahoo, and Bing (all $<$1 %).

Figure 9 shows a division of traffic from different countries in 2009 identified using GeoIP. Among users of CiteSeer^x, nearly half of users are from the United States. Taiwan, Germany, China, India, UK, France, and Canada are other major sources of traffic. Traffic from two hundred and twenty countries are grouped under ’other’.

Along with valid accesses, CiteSeer^x experiences a variety of attacks every day. These attacks involve access to forbidden areas, portscans, SQL injection attacks, double decoding, buffer overflow, and cross scripting attempts.

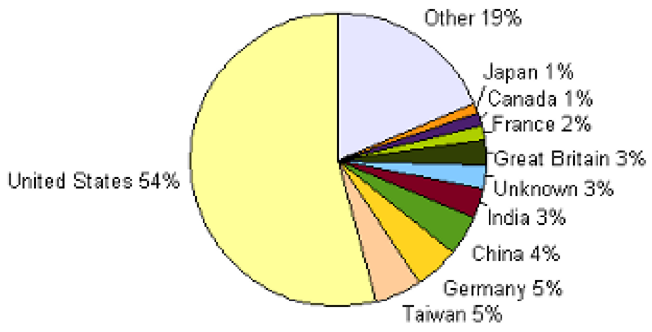


Figure 9: Country Profile

11 Collaboration and Distribution

SeerSuite has been developed in collaboration with several groups across the world. This collaboration includes exchange of ideas, development of modules and hosting of services. Independent copies of CiteSeer^x are maintained by these research groups at University of Arkansas, National University of Singapore and King Saud University.

Data collected as part of crawls, document metadata, anonymized user log data are available on request. Several research groups have already taken advantage of these datasets. The source code for SeerSuite has been released under the Apache software license version 2 on sourceforge.

To improve adoption of SeerSuite based systems and provide the user an efficient way to explore and deploy instances. A virtual appliance with SeerSuite has been made available for such a purpose. The appliance uses open source components and contains a complete deployment of SeerSuite. This allows users to explore and operate instances of SeerSuite, in the pre-installed VM. The use of the virtual appliance also benefits developers and testers.

User feedback and comments have helped improve both user the experience and the troubleshooting bugs. SeerSuite enables other innovative and valuable tools to be built using metadata extracted and published. Such efforts include projects such as PaperCube [14] and JabRef.

12 Lessons Learned and Future Work

By adopting a multi-tier architecture, open source applications and the use of software frameworks, SeerSuite has improved upon the client server architecture initially adopted by CiteSeer. In addition this is an approach that has provided reliable scalable services in CiteSeer^x.

One of the lessons learned is in the provisioning of infrastructure. To allow SeerSuite instances to grow to large sizes without constraints on storage and computation, virtualization and distributed computing need to be utilized and managed.

The number of requests for metadata and data from the CiteSeer^x collection emphasizes the need for developing automated methods data sharing such as API and OAI services for SeerSuite instances. There have also been requests for a content based similarity and duplicate detection service, which is under development.

With the new design adding features in SeerSuite is more straight forward. However, further work on separating components is required to support systems which make use of a smaller subset of services. These systems may not include the citation graph service. Among other services, MyCiteSeer should be developed as a stand alone service. This will easily enable other services such as HomePageSeer that take advantage of login based access. Development of MyCiteSeer as an independent service, shared across many Seer instances is under consideration. This would allow users to share information across projects and instances.

One of the continuing major challenges is new and high quality metadata extraction. Another is that modules will need to be refactored to support massive crawls using parallelization at the module level.

A number of optimizations have been implemented in SeerSuite, prominent among these is the citation relationships stored in the index. This allows citation relationship queries which form document summary views to be more readily handled. Such an optimization has the drawback in that updates to these relationships are only available to the user once there has been updates to the index.

A federation of services model adopted for newer services benefits both users and developers. This model gives users a peek into new features and researchers an easy a way to include new services. From a development standpoint, this is also useful since services can be tested and users can provide feedback.

Improvements to the User Interface will be required to support upcoming features. Ranking of results and improving relevance in search are active topics of research. Performance analysis tests for SeerSuite to identify possible improvements and issues are being designed.

13 Summary

We have described the design, architecture, and deployment of SeerSuite. We believe SeerSuite overcomes many of the issues in an earlier system, CiteSeer, which, due to its design, limited growth and extensions to other services. SeerSuite is designed to take advantage of open source applications, frameworks and state of the art components. It also allow users to readily build mashups and related applications. The use of loose coupling of modules and federated services enables SeerSuite to easily offer new features and components.

The user interface allows search, document summary views, and citations clustering. We identify workflows for generating these pages. Various features such as tagging, building collections and correcting documents were identified for the MyCiteSeer portal. In addition to providing services through the web user interface, SeerSuite also provides services through the OAI/PMH and API interfaces.

The statistics and usage pattern for CiteSeer^x, a SeerSuite instance, provide information about growth in traffic and the profile of users based on the country of origin. We show that SeerSuite is a collaborative venture with open source code and virtual appliances that encourage adoption and research. We believe that SeerSuite will continue to improve and support a wide variety of services and user needs while remaining scalable, reliable and robust.

14 Acknowledgments

We gratefully acknowledge partial support from the NSF and useful comments by Yves Petinot.

References

[1] ACM Portal. http://portal.acm.org/portal.cfm.

[2] ActiveMQ. http://activemq.apache.org/.

[3] CiteULike. http://www.citeulike.org/.

[4] DBLP. http://www.informatik.uni-trier.de/~ley/db/.

[5] Dspace. http://www.dspace.org/.

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

[7] Greenstone. http://www.greenstone.org/.

[8] Heritrix. http://crawler.archive.org/.

[9] Jersey API. https://jersey.dev.java.net/.

[10] JSR 311. https://jsr311.dev.java.net/nonav/releases/1.1/index.html.

[11] Open archives initiative - protocol for metadata harvesting v.2.0. http://www.openarchives.org/OAI/openarchivesprotocol.html.

[12] RePEc. http://repec.org/.

[13] Solr. http://lucene.apache.org/solr/.

[14] Bergstrom, P. Papercube. http://papercube.peterbergstrom.com.

[15] Councill, I. G., Giles, C. L., Iorio, E. D., Gori, M., Maggini, M., and Pucci, A. Towards next generation citeseer: A flexible architecture for digital library deployment. In *Research and Advanced Technology for Digital Libraries, ECDL 2006* (2006), pp. 111–122.

[16] Councill, I. G., Li, H., Zhuang, Z., Debnath, S., Bolelli, L., Lee, W. C., Sivasubramaniam, A., and Giles, C. L. Learning metadata from the evidence in an on-line citation matching scheme. In *JCDL* (2006), pp. 276–285.

[17] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, Irvine, California, 2000.

[18] Garfield, E. "Science Citation Index"a new dimension in indexing. *Science 144*, 3619 (1984), 649 – 654.

[19] Giles, C. L., Bollacker, K. D., and Lawrence, S. Citeseer: An automatic citation indexing system. In *Digital Libraries* (1998), pp. 89–98.

[20] Giles, C. L., Councill, I., Teregowda, P., and R., J. P. F. CiteSeer^x. http://citeseerx.ist.psu.edu, 2010.

[21] Ginsparg, P. Can Peer Review be better Focussed. http://people.ccmr.cornell.edu/~ginsparg/blurb/pg02pr.html.

[22] Han, H., Giles, C. L., Manavoglu, E., Zha, H., Zhang, Z., and Fox, E. A. Automatic document metadata extraction using support vector machines. In *JCDL '03: Proceedings of the 3rd ACM/IEEE-CS joint conference on Digital libraries* (2003), pp. 37–48.

[23] Han, H., Manavoglu, E., Zha, H., Tsioutsiouliklis, K., Giles, C. L., and Zhang, X. Rule-based word clustering for document metadata extraction. In *SAC* (2005), pp. 1049–1053.

[24] Huang, J., Ertekin, S., and Giles, C. L. Efficient name disambiguation for large scale databases. In *The 10th European Conference on Principles and Practice of Knowledge Discovery in Databases* (2006), pp. 536–544.

[25] Isaac Councill, C. L. G., and Kan, M.-Y. Parscit: an open-source crf reference string parsing package. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)* (Marrakech, 2008), European Language Resources Association.

[26] Kahn, R., and Wilensky, R. A framework for distributed digital object services. *International Journal on Digital Libraries 6*, 2 (2006), 115–123.

[27] Lawrence, S., Bollacker, K., and Giles, C. L. Distributed error correction. In *DL '99: Proceedings of the fourth ACM conference on Digital libraries* (1999), p. 232.

[28] Li, H., Councill, I., Lee, W.-C., and Giles, C. L. Citeseerx: an architecture and web service design for an academic document search engine. *Poster Session 15th International World Wide Web Conference* (2006).

[29] Liu, Y., Bai, K., Mitra, P., and Giles, C. L. Tableseer: automatic table metadata extraction and searching in digital libraries. In *JCDL* (2007), pp. 91–100.

[30] McCallum, A., Freitag, D., and Pereira, F. C. N. Maximum entropy markov models for information extraction and segmentation. In *ICML* (2000), pp. 591–598.

[31] Mitra, P., Giles, C. L., Sun, B., and Liu, Y. Chemxseer: a digital library and data repository for chemical kinetics. In *CIMS '07: Proceedings of the ACM first workshop on CyberInfrastructure: Information Management in eScience* (2007), pp. 7–10.

[32] Mitra, P., Giles, L., and Mueller, K. Chem$_x$Seer. http://chemxseer.ist.psu.edu.

[33] Nii, H. P. Blackboard systems, part one: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine 7*, 2 (1986), 38–53.

[34] Petinot, Y., Giles, C. L., Bhatnagar, V., Teregowda, P. B., Han, H., and Councill, I. Citeseer-api: towards seamless resource location and interlinking for digital libraries. In *CIKM* (2004), pp. 553–561.

[35] Petinot, Y., Teregowda, P. B., Han, H., Giles, C. L., Lawrence, S., Rangaswamy, A., and Pal, N. ebizsearch: an oai-compliant digital library for ebusiness. In *JCDL* (2003), pp. 199–209.

[36] Steve Lawrence, C. Lee Giles, K. B. CiteSeer. `http://citeseer.ist.psu.edu`, 1998.

[37] Tan, Q., Mitra, P., and Giles, C. Metadata extraction and indexing for map search in web documents. In *Proceeding of the 17th ACM CIKM* (2008), pp. 1367–1368.