First of all, this paper does not add much new stuff. Most of content is adopted from previous literatures. Even the proposed "Big Data Platform" is mostly adopted from workshop papers before. The new content is less than 30%. The only part that interests me is the last section where it talks about citeseerx traffic. It was my first time to see that the author search takes 37% of all traffic.
Overall, this is another CiteSeer paper with some ideas, but it lacks appropriate justification.

Towards Building a Scholarly Big Data Platform: Challenges, Lessons and Opportunities

Zhaohui Wu[†], Jian Wu[‡], Madian Khabsa[†], Kyle Williams[‡], Hung-Hsuan Chen[†], Wenyi Huang[‡], Suppawong Tuarob[†], Sagnik Ray Choudhury[‡], Alexander Ororbia[‡], Prasenjit Mitra[‡†], C. Lee Giles[‡†]
[†]Computer Science and Engineering, [‡]Information Sciences and Technology
Pennsylvania State University, University Park, PA 16802, USA
{zzw109,jxw394,madian,kwilliams,hhchen,wzh112,szt5115}@psu.edu
{szr163,ago109,pmitra,giles}@ist.psu.edu

ABSTRACT

We introduce a big data platform that provides various services for harvesting scholarly information and enabling efficient scholarly applications. The core architecture of the platform is built on a secured private cloud, crawls data using a scholarly focused crawler that leverages a dynamic scheduler, processes by utilizing a map reduce based crawl-extraction-ingestion (CEI) workflow, and is stored in distributed repositories and databases. Services such as scholarly data harvesting, information extraction, and user information and log data analytics are integrated into the platform and provided by an OAI and RESTful API. We also introduce a set of scholarly applications built on top of this platform including citation recommendation and collaborator discovery.

Categories and Subject Descriptors

H.3.7 [**Information Storage and Retrieval**]: Digital Libraries; H.3.3 [**Information Search and Retrieval**]: Text Mining

Keywords

Scholarly Big Data, Information Extraction, Big Data

1. INTRODUCTION

The recent decade has witnessed a clear growth in electronic publishing, with a large amount of scholarly data now available online. Using Microsoft Academic Search (MAS) and Google Scholar, we estimated that there are at least 114 million English-language scholarly documents or their records [1] accessible on the Web and new scholarly documents are generated at a rate at tens of thousands per day [21]. To enable easy access of online scholarly data, academic search engines and digital libraries usually need to crawl scholarly documents from the Web, extract useful information from documents, and then ingest (store and index) that information and the documents. For small scale (less than millions) and slowly growing (less than thousands per day) data, traditional client/server architectures might be able to handle the data throughput by using multiple coupled physical machines, single pipeline data processing, and static crawling strategies, as was done previously by Citeseer [13]. However, many challenges arise in accessing fast growing, large scale, and heterogeneous "scholarly big data" using such traditional systems.

First, *traditional architectures may not easily scale up to satisfy the storage and service requirements for fast growing data.* To handle the increasing scale of data, more machines are usually added into existing architectures for storage and web services, which increases the risk of hard drive or controller failures in addition to the cost of human labor and management. Furthermore, more sophisticated application-oriented services other than just traditional user-oriented services (e.g. paper search and download) should be provided to enable more advanced and useful scholarly applications. For example, services such as author profiling or disambiguation are helpful for collaborator recommendation or expert discovery. However, building these services on large scale data not only requires powerful computational resources, but, more importantly, they need a smart resource management and schedule platform that efficientially and dynamically allocates its resources.

Second, *standard systems may not well support the high data throughput due to the bottleneck imposed by a single pipeline.* For example, the traditional Citeseer platform can only support an ingestion rate of around 2000 documents per day. To scale up to 100 times that rate or larger, simply increasing the number of pipelines by adding physical resources might not be efficient due to the variety of latencies characteristic of different components of the pipeline (i.e. crawling, extraction, and ingestion). Thus, a better data processing framework and set of tools are needed to achieve higher data throughput.

Third, *classical approaches may not efficiently collect data due to inaccuracy of data filtering and latency and lack of coverage caused by static crawling strategies.* Static crawling strategies use a crawling seeds list that cannot be updated in

[1]By scholarly documents, we mean journal and conference papers, dissertations and masters theses, academic books, technical reports, and working papers. Patents are excluded.

978-1-4799-5569-5/14/$31.00

an automated and efficient manner. We need better strategies to manage the seeds list so that new sources of data can be discovered and invalid sources can be ruled out in a timely fashion. Additionally, we need accurate document classifiers to filter out non-scholarly documents and classify scholarly documents into different categories, not only based on publishing formats, but also into fields or subjects.

In light of these challenges, we propose a platform that can effectively harness scholarly big data, based on the current CiteseerX system [2]. The core architecture of the platform is built on a secured private cloud using a virtualization technique powered by VMware, which provides an efficient and feasible means of resource allocation and failure control, where data is harvested based on a scheduled crawl, processed in a map reduce framework, and stored in both SQL and NoSQL databases as well as distributed repositories using the Hadoop distributed file system (HDFS). Services for scholarly data harvesting, information extraction, and user or log data analytics are integrated into the platform and provided with via an Open Archives Initiative Protocol (OAI)[3] and a RESTful API. These services allow for the development of various scholarly applications such as citation recommendation and expert discovery. The contributions of this paper can be summarized as follows:

- We introduce a scholarly big data platform that can provide a variety of services for mining scholarly data and building scholarly applications.
- We share our practical experiences in building such a platform including: setting up the private cloud, developing a scholarly focused crawler, implementing a crawl-extraction-ingestion workflow, making effective us of distributed repositories, and so on.
- We summarize the challenges, practical lessons and possible future opportunities for mining scholarly big data through multiple case studies involving services and applications that have been built using this scholarly data platform.

2. THE SIZE OF SCHOLARLY DATA

The first two questions to ask are "how big is the scholarly data?" and "how much of it is freely available?" To answer them, we estimate the number of scholarly documents available on the Web, using capture/recapture methods, by studying the coverage of two major academic search engines: Google Scholar and Microsoft Academic Search [21]. Our approach assumes that each academic search engine samples the Web independently for papers and contains a subset of available documents. Next, we consider each search engine to be a random capture of the document population at a certain time. Using the intersection of these two captures, we estimate the entire size of the population. Since obtaining the entire database of both academic search engines was not feasible, we approximate this overlap by randomly sampling from each search engine and then determining the size of the overlap in this random sample. Our results show that at least 114 million English-language scholarly documents are accessible on the Web, of which Google Scholar has nearly 100 million. Of these, we estimate that at least 27 million (24%) are freely available since they do not require a subscription or payment of any kind. In addition, at a finer scale, we also estimate the number of scholarly documents on the Web for fifteen fields: Agricultural Science, Arts and Humanities, Biology, Chemistry, Computer Science, Economics and so on as defined by Microsoft Academic Search. In addition, we show that among these fields the percentage of documents defined as freely available varies significantly, i.e., from 12 to 50% [21], as shown in Table 1.

Based on these numbers, we found that our existing system, namely CiteseerX, covers only around 10% of these freely available scholarly documents. This provides strong motivation for scaling the system up to a newer, more advanced big data platform that has the capability of handling 100 million scholarly documents and ultimately facilitate larger scale scholarly services and applications.

Table 1: The Estimated Number of Scholarly Documents on the Web in Different Fields.

Discipline	Size in MAS	Estimated Size	public
Agriculture Science	447,134	1,088,711	12%
Arts & Humanities	1,373,959	5,286,355	24%
Biology	4,135,959	8,019,640	25%
Chemistry	4,428,253	10,704,454	22%
Computer Science	3,555,837	6,912,148	50%
Economics & Business	1,019,038	2,733,855	42%
Engineering	3,683,363	7,947,425	12%
Environmental Sciences	461,653	975,211	29%
Geosciences	1,306,307	2,302,957	35%
Material Science	913,853	3,062,641	12%
Mathematics	1,207,412	2,634,321	27%
Medicine	12,056,840	24,652,433	26%
Physics	5,012,733	13,033,269	35%
Social Science	1,928,477	6,072,285	19%
Multidisciplinary	9,648,534	25,798,026	43%
Total Sum		121,223,731	36,703,036

Table 2: Physical Servers for Private Cloud

Server Type	#cores[1]	Memory	Storage[2]	Quantity
Processing	12	96GB	1TB	5
Storage	12	32GB	30TB	2

[1] CPU frequency 2.5GHz.
[2] After RAID 5 for each unit.

3. ARCHITECTURE

The overall platform is composed of three layers, as shown in Figure 1. The architecture layer demonstrates the high-level system modules and the work flow. It can be divided into two parts: the frontend (Web servers and load balancers) that interacts with users, processes queries and provides different web services; the backend (crawler, extraction, and ingestion) that performs data acquisition, information extraction and supplies new data to the frontend. The services layer provides various services for either internal or external applications by APIs. The applications layer lists scholarly applications that build upon the services.

3.1 Virtual Infrastructure

The CiteseerX architecture used to be built on 18 physical servers. Since February, 2013, the entire system was migrated to a private cloud, with this infrastructure illustrated in Figure 2. The cloud is built on top of two storage servers and five processing servers, with specifications tabulated in Table 2. We use VMware EXSi 5.1 as the hypervisor installed on each processing server.

[2]http://citeseerx.ist.psu.edu/
[3]http://www.openarchives.org/

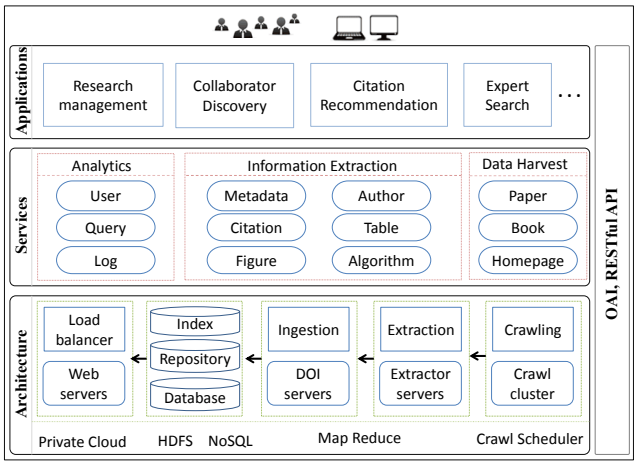

Figure 1: Overview of the platform.

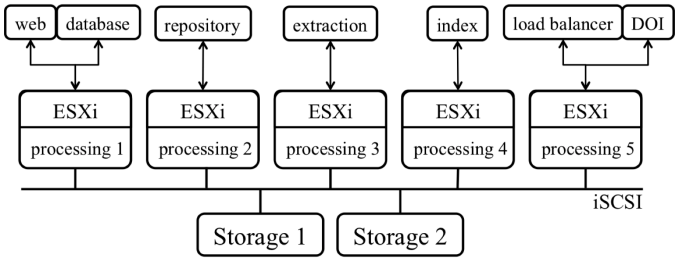

Figure 2: The private cloud infrastructure.

Migrating the entire system to a private cloud has proven to be the most cost-effective solution in upgrading the infrastructure [33]. The migrated system has become quite powerful due to the enhanced hardware and the cloud hypervisor also offers more reliability, scalability, maintainability and efficiency. First, if one processing server fails, the hypervisor can respond and move VMs on that server to another processing server in a matter of minutes. Second, a smaller footprint on the data center equates to less physical space as well as lower operating temperatures and thus more efficient use of power. This allows us to add additional physical servers to our cluster should we need more processing power or storage. Third, it is convenient to create and delete a new VM. By using a template based workflow, setup time can be reduced from days to minutes.

3.2 Focused Web Crawling

The crawl module actively crawls PDF files from the Web. These PDF files are first imported into the crawl database and repository before being passed to the extraction module. The web crawling process ensures that all documents available have (or used to have) at least one valid URL where it can be downloaded by an individual user. The focused crawler still faces some challenges for gaining higher crawling precision, recall/coverage and freshness.

The precision was achieved by applying a whitelist policy [32]. For each document URL (URL linking directly to a PDF file), we record its parent URL, which is the URL of a web page (most likely in HTML format) which contains the document URL. The crawl database contains more than 13 million document URLs and 2.4 million parent URLs. To apply this policy, we first create a whitelist by selecting high quality parent URLs which contain at least one *ingestable* document. We also setup the crawler so it does not go beyond the domains of whitelist URLs. Before the introduction of a whitelist policy, CiteSeerX used a blacklist file to filter out web domains it was not allowed to go or "crawl traps" it needed to avoid. While this policy gives more freedom for the crawler to visit new websites, it wastes a good deal f bandwidth for downloading non-academic documents. The CiteSeerX team also received complaints from websites with none or poorly written `robots.txt` files. As a result of the whitelist policy, the fraction of academic documents has increased from 30% to about 50% and we rarely receive security complaints by web masters.

While our initial focus was on papers pertaining to computer and information sciences, we have recently increased the diversity in three ways. First, we encourage users to submit URLs that we then crawl with a higher number of maximum hops than a regular crawl. This not only helps us locate documents but also new parent URLs that we can potentially use to obtain even more documents. Second, we download additional documents through crawl-truncated URLs in order to dig into the directories of discovered remote servers. Third, we import paper collections from other repositories such as arXiv and PubMed to incorporate papers from a breadth of disciplines. Recently, Microsoft Academic Search released their paper URLs and by crawling the first 7.58 million, we have collected 2.2 million documents[4]. This significantly expands the domain coverage of the CiteSeerX collection.

Freshness is a quality strongly associated with coverage, i.e., it reflects the coverage of recent documents. We achieve freshness by periodically updating the whitelist and recrawling URLs using a crawl scheduler, which we called *Scheduled Crawl*. Previously, user-submitted URLs were mixed with URLs into the Scheduled Crawl. Recently, we changed this by implementing two crawling instances, one for a scheduled crawl and another that is dedicated to crawling user submitted URLs. In this way, we prioritize the latter and make documents from these user-driven searches appear sooner in our production. The cycle for a scheduled crawl usually takes at least a few weeks before the whitelist is updated. A user-submission crawl cycles each day.

Our biggest challenge in the foreseeable future will still be coverage as the key issue is to find good seed URLs which lead to new academic documents. One way is to solve this problem is to take advantage of giant general search engines such as Google or Bing to retrieve papers by sending queries to their APIs. However, we are bounded by the usage limit of those APIs. As a result, obtaining a large corpus of documents is time-consuming and as a result, we need to carefully choose the queries we make in order to retrieve the most important papers first. An alternative is to leverage professional homepages of researchers [14], which usually contain fruitful and fresh sources of academic publications. However, the primary challenge facing a topical crawler is to effectively recognize these homepages among the myriad of other pages normally found in an institutional website such as course sites, news pages, student profiles, etc.

[4]URLs that match our blacklist were not crawled. A significant fraction of URLs do not link to PDFs, i.e., they link to metadata pages, so they do not give us full text files.

3.3 Document Classification

In CiteSeerX, the document classification module separates academic and non-academic documents and passes academic documents to the ingestion module. The text content is first extracted from PDF files. The older classification scheme utilized a rule-based approach in which a list of keywords/phrases are searched for in the text body, which is efficient and achieves an acceptable accuracy. A recent study based on a sample of 1000 papers randomly sampled from CiteSeerX database reveals precision of 90% and recall of 80% for academic documents.

To further improve the performance, we developed binary classification for categorizing crawled documents based on SVM which takes advantages of structural information from papers, such as document size, page size, aspect ratio, font size combined with keywords [8]. Moreover, we are working on classifying academic documents into multiple categories such as papers, books, reports, slides, etc. We have defined a complete schema for all categories in their scope and extract various features from each category. The goal is to provide CiteSeerX with functionality to achieve a higher quality classification in a timely manner and provide multiple formats of documents with topical linkages.

3.4 The CEI Workflow

The crawl-extraction-ingestion (CEI) workflow is the major part of the backend and continuously feeds new data into the index, database and repository. Fully automating and paralleling the workflow is highly desirable for CiteSeerX to scale up to the big data regime. We are developing a CEI pipeline (CEIP) to automate this workflow. The key components are a MySQL database and a job scheduler/tracker (JST). The database stores the job information including the crawler download path, the batch job ID, job status and corresponding timestamps. The JST periodically scans the crawl download directories for recently finished crawl jobs, automatically configures and starts the crawl document importer (CDI) to import the crawled documents. It then retrieves documents from the crawl database and distributes them to the extraction module. Each batch job has a unique job ID and a job status. The JST also periodically scans the extraction output folder for recently finished jobs and sends them over to the ingestion module. Both the extraction and ingestion modules can be parallelized using the Map Reduce method. The CEIP is depicted in Figure 3.

3.5 Distributed Repositories

CiteSeerX currently has a single repository of about 6 terabytes containing over 4 million document folders. Each document folder contains at least six files include the original PDF/postscript files and the associated metadata files. While it is relatively easy to read/write from a single repository, this presents potential scalability problems. First, due to some system constraints, the maximum size of a partition is about 10TB in the virtual environment. We can easily break this limit by doubling our collection, so a single repository is only temporary. Second, backing up that big repository is time consuming. Transferring all data in the repository takes more than a week. Even if we use the *rsync* for incremental backup, scanning 20 million files still takes tens of hours. Using multiple repositories, we increase the number of spindles which parallelizes data transfer. Splitting the repository into ten blocks can reduce the time

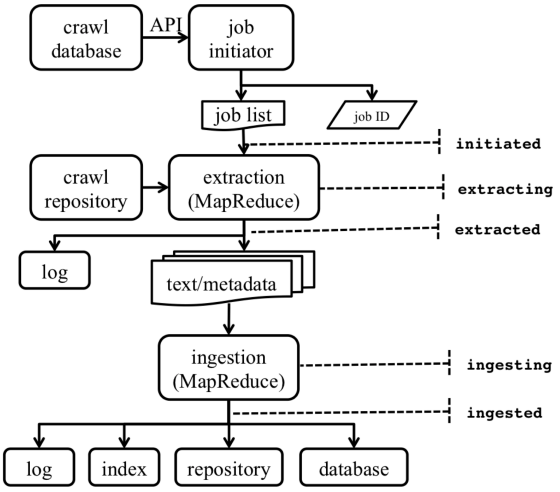

Figure 3: The CEIP workflow. Dashed lines indicate the points of job status change.

to copy the whole current repository to a few hours and incremental backup to less than an hour.

A promising solution is to use the Hadoop Distributed File system (HDFS), which uses commodity hardware to create multiple replicas of the same document on different machines. The HDFS has been under active development in the open source community, as well as by many consulting companies that provide enterprise-level support, i.e., Cloudera. Under this approach, an application only has to deal with a single repository, which exists on top of HDFS, while the reads and writes are handled by the file system itself.

3.6 Index Maintenance

The current indexing tool Solr (version 1.3) is not capable of performing real-time replication and as a result, we are only able to perform bulky backups. The index contains more than 2.6 million unique in-collection documents, and takes more than 170GB of space. As the repository grows, the index grows as well. Solr 4 has a replication feature and guarantees (almost) real-time synchronization. The biggest challenge in this upgrade is that Solr has changed its index format at least three times since Solr 1.3. There is no method for transferring the index directly from Solr 1.3 to Solr 4. As a result, we either have to go through intermediate versions (and iteratively update our index) before we get to Solr 4 or simply reindex all metadata. The first plan not only require lots of work but also adds unnecessary complexity in the case that an intermediate upgrade fails. The second plan starts from scratch but is more flexible, e.g., we can upgrade the index schema as needed.

3.7 Databases

The database stores metadata, citations, and other relevant information. Traditional CitesserX uses a relational database to store all those information. We are currently investigating the use of a graph database (Neo4j [5]) to store the structured data for CiteSeerX. There are several motivations for doing this. For instance, a graph is a natural representation of much of the structured data in CiteSeerX,

[5]http://www.neo4j.org/

such as the citation graph or the co-authorship graph. It is of interest to explore multi-hop paths on these graphs in order to gain insight into the nature of scholarly big data; however, it is generally infeasible to do this in real time with a relational database since multiple joins usually need to be performed. Thus, a graph database that has been designed to support these types of queries may be useful. That being said, relational databases are known to scale well and be reliable; thus, part of our migration investigation involves evaluating the performance of existing graph databases to see how they compare to relational databases at different scales and for different types of queries.

4. SCHOLARLY INFORMATION

The platform provides not only document search and download for users, but also various services to access finer scholarly information residing within scholarly big data, such as metadata, citations, algorithms, figures, tables and so on, which play important roles for building scholarly applications. Information extraction forms a crucial part of the platform and affects the overall usability and quality of the service due to the fact that the automatically extracted information is used as the metadata, which is used for searching and interacting with the site and for data collection. Given the fact that the platform integrates scholarly data from across the Web and is fully automated, extraction needs to be robust to variations among different document formats and styles and be scalable.

4.1 Metadata

Metadata of papers usually contains a title, authors, abstract, venue, volume and issue (for journals), page numbers, publisher, publisher address, publish date, copyright and ISBN (for books). It forms identifications of a scholarly document and thus plays essential roles for organizing and searching scholarly data. State-of-the-art techniques for metadata extraction are based on supervised machine learning such as Support Vector Machines (SVM) or Conditional Random Fields, which work reasonable well for academic papers within certain domains, but might have difficulties in scaling up to big data in diverse formats and styles from various fields. For example, CiteseerX uses a SVM-based header extractor (SVMHeaderParse) for papers in computer sciences [15]. However, we found this method performs poorly on academic books [34].

How can we efficiently extract metadata for 100 million scholarly documents with all possible publishing formats? It is unrealistic to get enough human-labeled instances to train a good extractor. Our practical solution contains two essential parts: first, "divide and conquer"; and second, active learning based on Web knowledge.

We train multiple extractors for documents of different publish genres including conference papers, journal, dissertations or theses, and academic books. A new document will first be labeled as one of the categories by the classifier of the crawler and then sent to the corresponding extractor.

In contrast to the previous SVMHeaderParse approach, trained on only a small (< 1000) human labeled dataset [15], we harvest the metadata from Web and then do text matching back to the title page of a document to label each line. For non-book documents, we harvest metadata from online digital libraries such as DBLP, Pubmed, MAS, and Google Scholar as ground truth; for book documents, we query metadata from Google Books through its API using ISBN search[6]. The ISBN can be accurately detected from books by search regular expression patterns after the string "ISBN". It includes two types, one for a 10-digit ISBN and another for 13. As regular expressions, it appears as:
$'i\backslash s?s\backslash s?b\backslash s?n(10|[\backslash s-]10)?[:\ -]?[\backslash s]\{0,5\}([\backslash dx-]\{13\})'$
$'i\backslash s?s\backslash s?b\backslash s?n(13|[\backslash s-]13)?[:\ -]?[\backslash s]\{0,5\}([\backslash dx-]\{17\})'$
We found 25% of our papers have matching counterparts in DBLP and 10% of books can get accurate metadata from Google Books, which gives us a large enough labeled data. However, this approach also yields a great deal of redundant information. For example, there are many papers from the same journal or conferences with few varieties in features. We thus apply active learning to select the representative training examples [36].

4.2 Citations

Citations play an more important role for scholars to assess and track the impact of scientific works, and to model the evolution of research. Metrics based on number of citations have been widely studied to measure scientific achievement for scholars. Citation networks have also been studied as plausible models of information flow and evolution.

To extract citations, we first need to accurately locate the Reference or Bibliography block, which usually has obvious indicators such as "References", "Bibliography" or "Sources". For papers, we do a reverse search from the end. However, books may have a bibliography at the end of each chapter. Thus, we need to search bibliography against the whole body of a book rather than in only the last few pages. If we find a line containing only one of the three keywords and the lines followed are ordered reference items, we identify it as a bibliography block. We search the ordered number at the beginning of each reference until there are no continuously increasing numbers found in the following 30 lines. Citations are then parsed using the ParsCit citation parsing tool [9]. Since it is primarily designed for papers, we improve the parsed results using an external name dictionary of authors and a thesaurus of venue names. Author names are collected from CiteSeerX and ArnetMiner databases while a thesaurus of venues is constructed based on rules and manual editing. Furthermore, the citation context for each extracted citation is stored, which facilitates further citation analysis.

While most previous work of citation analysis has focused only on citations from papers, we found citations from academic books were also valuable and should not be neglected in peer review research evaluation [34]. Thus, we will connect papers and books to make a complete citation network and provide for a more comprehensive research assessment.

4.3 Authors

Authors form an important component of a digital library. Therefore, most state-of-the-art digital libraries provide both document search and author search. In this section, we discuss several topics related to author mining in large-scale digital libraries and our experiences in mining author information.

4.3.1 Author Profiling

The documents published by an author are great sources to extract additional information. For example, the emails

[6]https: //www.googleapis.com/books/v1/volumes?q=isbn:ISBN&key=API Key

and the affiliations are usually printed along with the author names in a paper; the paper contents are commonly used to infer the research interest of the authors [5]. Papers published by other researchers can also be used to infer an author's information. In the past, we utilized the reference list and the referenced text extensively to obtain the list of venues a researcher has published in and the key concepts of target papers. In a complementary fashion, we also use researchers' professional webpages to retrieve more detailed author information.

4.3.2 Author Disambiguation

There are two types of author ambiguity cases. First, different authors may share the same name. For example, DBLP shows that the author "Li Chen" published more than 60 papers in several different domains in 2013. In this case, it is very likely that different "Li Chen"s have been incorrectly regarded as one author. Second, an author's name may sometimes be represented in different forms. For example, Dr. W. Bruce Croft could be recorded as "Bruce Croft", "W. B. Croft" or other name variations.

We use a Random Forest model trained on several features to disambiguate two authors a and b in two different papers p and q [28]. These features include the similarity between a and b's name strings, the relationship between the authoring order of a in p and the order of b in q, the string similarity between the affiliations, the similarity between emails, the similarity between coauthors' names, the similarity between titles of p and q, and several other features.

A live digital library ingests new documents actively. New documents may sometimes contain new evidence that can disprove or invalidate a previously learned author disambiguation model. We are interested in investigating online learning algorithms capable of incorporating new evidences into the model on-the-fly.

4.3.3 Coauthorship

Authors' collaborations are commonly inferred by their coauthored papers and represented by a coauthorship network, in which each node is an author, and two nodes are connected if the two authors have coauthored.

Researchers have investigated various collaboration patterns among scholars. The number of coauthors per paper and the number of papers an author publishes have been studied from a variety of disciplinary perspectives. We conducted a longitudinal analysis of the coauthorship network to reveal the evolution of collaboration and to predict future collaborations [4, 16, 3]. We also employed graph measures to discover experts of given queries [11].

4.4 Algorithms

Algorithms are ubiquitous in computer science and the related literature that offer stepwise instructions for solving computational problems. Researchers are constantly developing new algorithms to either solve new problems or algorithms that improve upon the existing ones. With many new algorithms being reported every year, it would be useful for the platform to have services that automatically identify, extract, index and search the ever-increasing collection of algorithms, both new and old. Such services could prove useful to researchers and software developers alike looking for cutting-edge solutions to their daily technical problems.

4.4.1 Algorithm Representations and Metadata

A majority of algorithms in computer science documents are summarized and represented as pseudocode [29]. Three methods for detecting pseudocode in scholarly documents have been developed including rule based (PC-RB), machine learning based (PC-ML), and combined (PC-CB) methods. The *PC-RB* method extends the state-of-the-art approach. The *PC-ML* method employs machine learning techniques to extract sparse boxes from a document and classify each box as either pseudocode or not using a novel set of 47 features. *PC-CB* captures the benefits of both former methods. The best performance in terms of F1 is achieved by the *PC-CB* method with the combination of the rule-based method and the majority vote of *Logistic Model Trees*, *Random Forest*, and *Repeated Incremental Pruning to Produce Error Reduction (RIPPER)* classifiers.

4.4.2 Extracting and indexing

Indexable metadata is extracted for each detected pseudocode. Currently, the metadata of a pseudocode includes its caption, textual summary, and that of the document containing such a pseudocode (i.e. title, year of publication, abstract, etc.). The metadata is then indexed using Apache Solr[7]. The search is done by textually matching the search query with the textual metadata of the extracted pseudocodes, and the retrieved pseudocodes are ranked by the TF-IDF similarity score between the query and the pseudocode metadata.

4.4.3 Challenges and future works

Being able to extract algorithm-specific metadata would allow for categorizing, filtering, and deeper analysis of the extracted algorithms. Algorithm-specific metadata includes the algorithm's name, inputs, outputs, runtime complexity, target problems, and data structures. Extracting algorithm specific metadata is challenging because such information normally cannot be extracted directly from the content in the document containing the algorithms. Authors use different styles when writing about their proposed algorithms, and some may not even provide all the desired pieces of metadata. Another challenge occurs when a document contains multiple algorithms, where further disambiguation is needed to map the extracted matadata to correct algorithms.

Knowing what an algorithm actually does could shed light on multiple applications such as algorithm recommendation and ranking. Previously, we made a first attempt to mine semantics of algorithms by studying the algorithm co-citation network, where each node is a document that proposes some algorithms, and each edge weight is the frequency of algorithm co-citation [31]. We clustered the co-citation network in order to obtain groups of similar algorithms.

One of our ongoing projects involves studying how algorithms influence each other over time. This study would allow us to not only discover new and influential algorithms, but to also study how existing algorithms are applied in various fields of study. In order to do this, we propose the construction and study of the algorithm citation network, where each node is an algorithm, and each direct edge represents how an algorithm *uses* other existing algorithms.

We showed that algorithm usage can be captured by analyzing the algorithm citation context, and proposed a clas-

[7] http://lucene.apache.org/solr/

sification scheme for algorithm citation function in scholarly works [30]. The scheme consisted of 9 classes representing possible ways in which an algorithm can be used. The 9 classes can be divided into 3 groups based on the authors' attitude towards the cited algorithms: *favorable*, *neutral*, and *critical*. Our future work explores the possibility of building an automatic classifier to classify algorithm citation contexts, which would allow the automatic construction of a large scale algorithm citation network.

4.5 Figures

Academic papers usually contain figures that report experimental results, system architecture(s), demos and so on. Often, the data present in such figures cannot be found within the text of the documents. Thus, extracting figures and associated information would be important to better understand the documents.

Figures can be embedded in PDF documents as raster graphics (JPEG, PNG formats) or vector graphics (SVG, eps). While it is trivial to extract raster images from PDF documents (as they are embedded as data streams), it is extremely hard to extract vector graphics from PDF documents. Vector graphics contain a set of instructions such as "draw a line, rectangle or bezier curve" or "paint a text on screen". PDF itself is a vector graphics format. The operators for text and graphics are interleaved in a PDF document and it is hard to understand which instruction corresponds to a figure and which do not. Chao et al. [2] proposed an approach for extraction of figures from PDF documents, where each page in the document is converted into an image and that resulting image is analyzed through segmentation algorithms to detect text and graphics regions. We extended this work to improve the accuracy of the segmentation algorithm through clustering techniques and other heuristics. Among 230 random page images, we were able to extract all figures correctly from 222 page images.

We used positional and font related information extracted from digital documents for accurate extraction of figure metadata. We proposed a rule based system that used these features. These features are usually extracted using PDF processing libraries such as PDFBox or XPdf but are not generic enough. Therefore, we devised a machine learning based system based on syntactic features extracted from only the text of the documents[6].

We extracted 90,000 figures and metadata from 160,000 documents published in chemical journals. We built a search engine on top of that extracted metadata. We have incorporated a special ranking function to improve the search results. The search engine has been integrated in an universal search engine which allows chemists to search on chemical name, formula, full text, authors, tables and figures extracted from documents[7].

Extraction of data from figures is a hard task and has been attempted before[24] with limited success. Naturally, to extract data from a figure we need to understand the "type" of the figure first: whether it is a line graph, scatter plot, bar chart, etc. We focused on analyzing line graphs as they are common in research papers and specifically used for reporting experimental results.

Following [25], we developed a classification algorithm for a binary classification problem where we classify each figure as a line graph or not. We trained our model on a set of 475 figures extracted from chemistry documents and tested on 50 figures. Overall classification accuracy for our model was 85%. We developed a suite of image processing algorithms to find the X axis, Y axis, axes labels and data points in the axis from line graphs. Future work involves extraction of curves from the plotting region.

Table 3: Collection and Usage Statistics

Statistic	Value
#Documents	3.5 million
#Unique documents	2.5 million
#Citations	80 million
#Authors	3-6 million
#docs added monthly	300,000
#docs downloaded monthly	300,000-2.5 million
Individual Users	800,000
Hits per day	2-4 million

4.6 Others

Other types of scholarly information we have experience with include tables [23], acknowledgments [22], table of contents [38], and back-of-the-book indices [37, 35]. Tables are widely used to demonstrate experimental results or statistical data in a condensed way and thus would be a valuable source of information to search. Acknowledgments are argued to be an important attribution of gratitude that in some cases refers to at least as much contribution as that of the last author [22]. Table of contents serve as guidelines for long documents such as conference proceedings, dissertations and master theses, and academic books. Back-of-the-book indexes, usually appearing near the end of a book, are collections of words or phrases, often alphabetically arranged, that help users locate information within the book.

5. ANALYTICS

Besides various services for mining scholarly information, we also provide analytic services regarding users, queries and logs. To enable those services, we mined information from our server logs that includes the client's IP address, the client's name, the time of the request, the requested URL, the referrer URL, the user agent, the response code, the amount of bytes sent back, and the session identifier. The log files are processed using *Pig* scripts [8], and each of the above mentioned fields is imported into a Apache Hive table stored in Hadoop's HDFS [9]. We present some of our analytic study results based on the logs generated during the months of September 2009 to March 2013, totaling 100 GB of compressed files. The total number of imported hive entries are 3,317,634,711.

Table 3 shows various approximate statistics related to the size of the collection as well as its usage. There are 800,000 individual users in total and 2-4 million hits per day. We found that requests have been received from more than 200 countries world wide. Table 4 lists the top 10 countries from which requests originate along with their percentage of traffic. The top 10 countries account for more than 70% of CiteSeerX traffic, and 64% of download requests. The top sources of traffic in order are Google, direct traffic, Baidu, Bing, DBLP, Yahoo, Wikipedia, ScientificCommons, and Google Scholar with Google accounting for 69% of

[8]http://pig.apache.org/
[9]http://hive.apache.org/

Table 4: Traffic from the Top 10 Countries

Country	Traffic %	Country	Download %
USA	28.86%	USA	21.95%
China	19.22%	China	11.72%
India	5.42%	India	10.66%
Germany	5.36%	UK	5.27%
UK	3.28%	Germany	3.8%
France	2.57%	Iran	2.39%
Iran	1.59%	Canada	2.33%
Japan	1.54%	France	1.96%
Canada	1.52%	Australia	1.84%
Australia	1.39%	Malaysia	1.62%
Other	29.25%	Other	36.41%

Table 5: Search Type Distribution

Search Type	Number	Percentage
	42,134,882	55.92%
Author	28,502,533	37.82%
Document	4,636,115	6.15%
Table	35,073	0.04%
Total	75,344,617	100%

the traffic. It is interesting to note that there are differences between the top 10 countries sorted by the number of downloads and the top 10 of traffic.

CiteSeerX offers three types of search where users can perform *document search*, *author search*, and *table search*. Each of these types of search is powered by a separate index. An advanced search page is available on the document index to complement the single query box default search. During the period of this study, CiteSeerX received 75,344,617 search requests. The proportion of search types are presented in Table 5. The first row indicates missing search types which default to a *document search*. As seen in the table, there is a significant interest in searching for author names with 37% of the search requests targeting the authors index. Equally popular was advanced search where it was found that 38% of the *document search* used the advanced search box.

By examining the queries with type *document search* we found that the average length of a query is 3.85 terms. This is significantly higher than the average Web search query length of 2.35 terms [26] found in AltaVista and similarly in Excite [18]. However more recent studies of Web search query length put the average at 2.9 [39] and 3.08 [27]. It is, therefore, believed that the number of query terms has increased over time [27], which explains the difference in the number of query terms we found with that observed in the Elsevier's ScienceDirect OnSite query logs where the reported average query length was 2.27 [20].

The majority of the users, or 92.58% of them, looked at the first result page only, however, users were more likely to look at 5 results pages than looking at 2, 3 or 4 pages as shown in Table 6. This indicates that users either found what they want on the first result page (or did not find it and gave up), or were willing to explore more results in order to find a satisfactory result. In the case of of *author search*, 99% of the users browsed one result page only.

Table 6: The Number of Result Pages a User Looks at When Doing Document Search

Number of Pages	Percentage
1	92.58%
5	2.43%
2	1.32%
3	1.09%
4	0.7%

Table 7: Number of Document Views per Query

#views	ratio	cumulative ratio
1	83.87%	83.87%
2	10.31%	94.18%
3	1.84%	96.02%
4	1.84%	97.86%
5	0.35%	98.21%
6	0.63%	98.83%
7	0.13%	98.96%
8	0.26%	99.22%
9	0.15%	99.37%
10	0.16%	99.53%
> 10	0.47%	100.00%

We study the number of document views after submitting a query. We ignore the sessions where users issue a query but do not click on any returns. The results are presented in Table 7. Most users click very few documents from the returned list. This suggests that most users follow the *Cascade Model* – a popular model where users view search results from top to bottom and leave as soon as they find a document of interest [10]. The multiple browsing models [12], i.e., the models that assume a query will lead to more clicks, play minor roles on CiteSeerX.

6. DATA, CODE AND API

We believe data sharing is important to foster collaboration and research. However, due to the large size and potential copyright issues, challenges exist in sharing our data. Although the data is crawled from the public Web by obeying site crawling policies, it is possible that some copyrighted material is collected. From time to time we receive requests from authors and publishers to remove documents. However, we still believe it is beneficial to share data.

We use the Open Archives Initiative Protocol [10] to share the metadata. By accessing the OAI Harvest URL [11], it is possible to download the metadata for all papers. This is the easiest way to access the platform data and seems to be widely used with an average of 4983 requests per month. We also make dumps of our databases available on Amazon S3 for download. This has the benefit of alleviating some cost of distributing the data since the cost of download traffic is paid for by the user. Challenges still remain in how to distribute the core repository, which contains the actual PDF papers and extracted text. Besides the copyright issues, there are other, more technical challenges inherent in distributing the data, which currently is larger than 6TB [1]. Furthermore, this repository is growing at a rate of about 10–20GB per day thereby making it a challenge to keep this repository synchronized with others over the Web.

CiteSeerXand related projects (such as the extraction mod-

[10]http://www.openarchives.org/
[11]http://citeseerx.ist.psu.edu/oai2

ules) are usually open-sourced under the permissive Apache License Version 2. The motivation for doing this is to allow other research groups to run their own versions as well as to allow the community to make improvements to CiteSeerXthat can be used to yet further improve the service itself. The source code for CiteSeerXwas previously hosted on SourceForge[12]; however, recently the source code has been migrated to GitHub[13] to enable better collaboration.

The platform provides a RESTful API in addition to the OAI service. For example, CiteSeerExtractor [14] is a standalone Web service that provides a RESTful API for information extraction from scholarly documents. Based on the extraction modules currently used in the platform, CiteSeerExtractor can be integrated into any application that needs to perform scholarly information extraction. This greatly simplifies the information extraction process and allows for centralized extraction that can easily be improved without needing to distribute the improvements. CiteSeerExtractor is an open source project publicly available on our GitHub page, thereby making it easy for other research groups to deploy their own versions of it and allow us to benefit from any community improvements to the software.

7. SCHOLARLY APPLICATIONS

7.1 Citation Recommendation

RefSeer [15] is a citation recommendation system that builds upon the platform mainly using citation information. Given either text from an abstract/description or part of a paper, RefSeer presents both topic-related global recommendation and citation-context based local recommendation.

For global recommendation, RefSeer internally computes from the text a topical composition based on topic modeling [19]. This model associates terms in the citation context and assumes that the words and citations within the citing paper are generated from a topic-word and topic-citation multinomial distribution, trained over all documents in the CiteSeerX repository. For local recommendation, RefSeer uses a citation translation model to learn the "translation" probability of citing a document given a word $\Pr(d|w)$ [17]. It assumes that the citation context to be the "descriptive language" and the "reference language" to consist of references, where each referenced paper is considered as a "word". By applying translation model on the two languages, we can compute the probability of a citation given a word.

7.2 Collaborator Discovery

CollabSeer [16] is a search engine for discovering potential collaborators for a given author [4]. It discovers potential collaborators by analyzing the structure of a user's coauthor network and research interests. Currently, CollabSeer supports three different network structure analysis modules for collaborator search: Jaccard similarity, cosine similarity, and our relation strength similarity. Users can further refine the recommendation results by clicking on their topics of interest, which are generated by automatically extracting key phrases from previous publications.

[12]http://citeseerx.sourceforge.net/
[13]https://github.com/SeerLabs
[14]http://citeseerextractor.ist.psu.edu:8080/static/index.html
[15]http://refseer.ist.psu.edu/
[16]http://collabseer.ist.psu.edu/

7.3 Expert Recommendation

CSSeer [17] is an expert discovery and related topic recommendation system mainly targeted at Computer Science. It is a highly automated system with little manual involvement. Given a term or phrase q for which an expert is needed, the experts of q are recommended based on both textual relevance and the quality of their related published documents. In addition, a list of related topics of q is provided for reference. Compared to other few publicly available expert recommenders, our system solely provides experts of related topics. Users are more likely to obtain a comprehensive list of experts by browsing through the experts of related topics provided by CSSeer [5].

8. CONCLUSION AND FUTURE WORK

We described an effort in building a scholarly big data platform that aims to support searching and mining of (nearly 100 million) scholarly documents in a large-scale setting. Specifically, we presented an architecture based on a virtual infrastructure using a private cloud with the design of the key modules, which included a focused crawler, a crawl-extraction-ingestion workflow, and distributed repositories and databases. We developed various services enabled by our data crawling, information extraction and analytics. These services can be used to mine the scholarly information residing in large scale documents and be used to build various user-oriented applications, such as those that perform citation recommendation, expert recommendation and collaborator discovery.

Our goal is to keep investigating this platform, both practically and theoretically. We are now integrating crawled MSA data which could significantly increase the size of our data. We are also constructing a scholarly knowledge graph that links together all types of scholarly information and entities, which could be billions of entities and links. This could enable our platform to provide more semantic-based and knowledge-based services.

9. ACKNOWLEDGEMENTS

We gratefully acknowledge support from the National Science Foundation and comments from the reviewers.

10. REFERENCES

[1] C. Caragea, J. Wu, A. Ciobanu, K. , Williams, J. Fernández-Ramırez, H.-H. Chen, Z. Wu, and C. L. Giles, "Citeseerx: A scholarly big dataset," in *Proceedings of ECIR*, 2014, pp. 311–322.

[2] H. Chao and J. Fan, "Layout and content extraction for pdf documents," in *Document Analysis Systems VI*. Springer, 2004, pp. 213–224.

[3] H.-H. Chen and C. L. Giles, "Ascos: an asymmetric network structure context similarity measure," in *Proceedings of ASONAM*, 2013, pp. 442–449.

[4] H.-H. Chen, L. Gou, X. Zhang, and C. L. Giles, "Collabseer: a search engine for collaboration discovery," in *Proceedings of JCDL*, 2011, pp. 231–240.

[5] H.-H. Chen, P. Treeratpituk, P. Mitra, and C. L. Giles, "Csseer: an expert recommendation system based on citeseerx," in *Proceedings of JCDL*, 2013, pp. 381–382.

[17]http://csseer.ist.psu.edu/

[6] S. R. Choudhury, P. Mitra, A. Kirk, S. Szep, D. Pellegrino, S. Jones, and C. L. Giles, "Figure metadata extraction from digital documents," in *Proceedings of ICDAR*, 2013, pp. 135–139.

[7] S. R. Choudhury, S. Tuarob, P. Mitra, L. Rokach, A. Kirk, S. Szep, D. Pellegrino, S. Jones, and C. L. Giles, "A figure search engine architecture for a chemistry digital library," in *Proceedings of JCDL*, 2013, pp. 369–370.

[8] K. W. S. D. G. M. K. P. T. Cornelia Caragea, Jian Wu and C. L. Giles., "Automatic identification of research articles from crawled documents." in *Proceedings of WSDM-WSCBD*, 2014.

[9] I. G. Councill, C. L. Giles, and M. yen Kan, "Parscit: An open-source crf reference string parsing package," in *Proceedings of the LREF*, 2008, pp. 661–667.

[10] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey, "An experimental comparison of click position-bias models," in *WSDM*, 2008, pp. 87–94.

[11] S. Das, P. Mitra, and C. L. Giles, "Ranking experts using author-document-topic graphs," pp. 87–96, 2013.

[12] G. Dupret and B. Piwowarski, "A user browsing model to predict search engine click data from past observations." in *SIGIR*, 2008, pp. 331–338.

[13] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: an automatic citation indexing system," in *Proceedings of DL*, 1998, pp. 89–98.

[14] S. D. Gollapalli, C. L. Giles, P. Mitra, and C. Caragea, "On identifying academic homepages for digital libraries," in *Proceedings of JCDL*, 2011, pp. 123–132.

[15] H. Han, C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox, "Automatic document metadata extraction using support vector machines," in *JCDL*, 2003, pp. 37–48.

[16] J. Huang, Z. Zhuang, J. Li, and C. L. Giles, "Collaboration over time: characterizing and modeling network evolution," in *Proceedings of WSDM*, 2008, pp. 107–116.

[17] W. Huang, S. Kataria, C. Caragea, P. Mitra, C. L. Giles, and L. Rokach, "Recommending citations: translating papers into references." in *Proceedings of CIKM*, 2012, pp. 1910–1914.

[18] J. B. B. J. Jansen and T. Saracevic, "Real life information retrieval: A study of user queries on the web," *SIGIR Forum*, vol. 32, no. 1, pp. 5–17, 1998.

[19] S. Kataria, P. Mitra, and S. Bhatia, "Utilizing context in generative bayesian models for linked corpus." in *Proceedings of AAAI*, 2010, pp. 1340–1345.

[20] H.-R. Ke, R. Kwakkelaar, Y.-M. Tai, and L.-C. Chen, "Exploring behavior of e-journal users in science and technology: Transaction log analysis of elsevier's sciencedirect onsite in taiwan," *Library and Information Science Research*, vol. 24, no. 3, pp. 265 – 291, 2002.

[21] M. Khabsa and C. L. Giles, "The number of scholarly documents on the public web," *PLOS one*, 2014.

[22] M. Khabsa, P. Treeratpituk, and C. L. Giles, "Ackseer: a repository and search engine for automatically extracted acknowledgments from digital libraries," in *Proceedings of JCDL*, 2012, pp. 185–194.

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

[24] X. Lu, S. Kataria, W. J. Brouwer, J. Z. Wang, P. Mitra, and C. L. Giles, "Automated analysis of images in documents for intelligent document search," *IJDAR*, vol. 12, no. 2, pp. 65–81, 2009.

[25] M. Savva, N. Kong, A. Chhajta, L. Fei-Fei, M. Agrawala, and J. Heer, "Revision: Automated classification, analysis and redesign of chart images," in *Proceedings of UIST*, 2011, pp. 393–402.

[26] C. Silverstein, M. Henzinger, H. Marais, and M. Moricz, "Analysis of a very large AltaVista query log," Digital Systems Research Center, Tech. Rep. 1998-014, 1998.

[27] M. Taghavi, A. Patel, N. Schmidt, C. Wills, and Y. Tew, "An analysis of web proxy logs with query distribution pattern approach for search engines." *Computer Standards and Interfaces*, vol. 34, no. 1, pp. 162–170, 2012.

[28] P. Treeratpituk and C. L. Giles, "Disambiguating authors in academic publications using random forests," in *Proceedings JCDL*, 2009, pp. 39–48.

[29] S. Tuarob, S. Bhatia, P. Mitra, and C. Giles, "Automatic detection of pseudocodes in scholarly documents using machine learning," in *Proceedings of ICDAR*, 2013, pp. 738–742.

[30] S. Tuarob, P. Mitra, and C. Giles, "A classification scheme for algorithm citation function in scholarly works," in *Proceedings of JCDL*, 2013, pp. 367–368.

[31] S. Tuarob, P. Mitra, and C. L. Giles, "Improving algorithm search using the algorithm co-citation network," in *Proceedings of JCDL*, 2012, pp. 277–280.

[32] J. Wu, P. Teregowda, J. P. F. Ramírez, P. Mitra, S. Zheng, and C. L. Giles, "The evolution of a crawling strategy for an academic document search engine: whitelists and blacklists," in *Proceedings of WebSci*, 2012, pp. 340–343.

[33] J. Wu, P. Teregowda, K. Williams, M. Khabsa, D. Jordan, E. Treece, Z. Wu, and C. Giles, "Migrating a digital library to a private cloud," in *Proceedings of the IC2E*, 2014.

[34] Z. Wu, S. Das, Z. Li, P. Mitra, and C. L. Giles, "Searching online book documents and analyzing book citations," in *Proceedings of DocEng*, 2013, pp. 81–90.

[35] Z. Wu and C. L. Giles, "Measuring term informativeness in context," in *Proceedings of NAACL-HLT 2013*, 2013, pp. 259–269.

[36] Z. Wu, W. Huang, C. Liang, and C. L. Giles, "Crowd-sourcing web knowledge for metadata extraction," in *Proceedings of JCDL*, 2014.

[37] Z. Wu, Z. Li, P. Mitra, and C. L. Giles, "Can back-of-the-book indexes be automatically created?" in *Proceedings of CIKM*, 2013, pp. 1745–1750.

[38] Z. Wu, P. Mitra, and C. L. Giles, "Table of contents recognition and extraction for heterogeneous book documents," in *Proceedings of ICDAR*, 2013, pp. 1205–1209.

[39] Y. Zhang, B. J. Jansen, and A. Spink, "Time series analysis of a web search engine transaction log." *Inf. Process. Manage.*, vol. 45, no. 2, pp. 230–245, 2009.