Design Considerations for a Sustainable Scholarly Big Data Service

Jian Wu
Computer Science
Old Dominion University
Norfolk, Virginia, USA
j1wu@odu.edu

Shaurya Rohatgi
Information Sciences & Technology
Pennsylvania State University
University Park, Pennsylvania, USA
szr207@ist.psu.edu

Manoj K. Angadi
Computer Science& Engineering
Pennsylvania State University
University Park, Pennsylvania, USA
mxa5887@psu.edu

Kavya S. Puranik
Computer Science & Engineering
Pennsylvania State University
University Park, Pennsylvania, USA
kzp5555@psu.edu

C. Lee Giles
Information Sciences & Technology
Pennsylvania State University
University Park, Pennsylvania, USA
clg20@psu.edu

ABSTRACT

The advancement of web programming techniques, such as Ajax and jQuery, and datastores, such as Apache Solr and Elasticsearch, have made it much easier to deploy small to medium scale web-based search engines. However, developing a sustainable search engine that supports scholarly big data services is still challenging often because of limited human resources and financial support. Such scenarios are typical in academic settings or small businesses. Here, we showcase how four key design decisions were made by trading-off competing factors such as performance, cost, and efficiency, when developing the Next Generation CiteSeerX (NGX), the successor of CiteSeerX, which was a pioneering digital library search engine that has been serving academic communities for more than two decades. This work extends our previous work in Wu et al. (2021) and discusses design considerations of infrastructure, web applications, indexing, and document filtering. These design considerations can be generalized to other web-based search engines with a similar scale that are deployed in small business or academic settings with limited resources.

CCS CONCEPTS

• **Information systems** → Search engine indexing; Document filtering; *Digital libraries and archives.*

KEYWORDS

digital library search engine, citeseerx, infrastructure, architecture

ACM Reference Format:
Jian Wu, Shaurya Rohatgi, Manoj K. Angadi, Kavya S. Puranik, and C. Lee Giles. 2018. Design Considerations for a Sustainable Scholarly Big Data Service. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Google has dominated the search engine market for more than 20 years. However, a general search engine like Google may not satisfy all search needs from users. Vertical search engines have played vital roles in searching focused document collections. Digital library search engines (DLSEs) are a type of vertical search engine that serves digital academic documents. Examples of DLSEs included CiteSeer [10], Microsoft Academic ([21]; retired in 2021), Google Scholar [12], and Semantic Scholar [9]. Google Scholar obtains its documents from Google with a large portion of metadata directly from publishers or automatically extracted using a proprietary extractor. Google Scholar does not cache documents. Instead, the search engine result page (SERP) redirects users to URLs where the documents could be downloaded, such as CiteSeerX. Semantic Scholar maintains a copy of open-access papers but the search architecture is proprietary and unlikely to be deployed for building institutional DLSEs, most likely due to the highly customized design and running cost. CiteSeerX released a framework called SeerSuite [20], which could be used for deploying a DLSE for an institutional document collection. With the updates of software, such as operating systems (OSs) and web development frameworks, many components that were used for building SeerSuite exhibit security vulnerabilities and/or no longer supported. The infrastructure that was used to support relatively medium-size document collection has shown scalability bottlenecks, and is no longer suitable for a sustained service for scholarly big data. Therefore, it is necessary to design a new framework to provide an accessible, usable, scalable, and sustainable scholarly big data service. The new framework, called the Next Generation CiteSeerX (NGX), is designed as a general framework that can be deployed for other digital documents.

Previously we [23] analyzed the strengths and weaknesses of the CiteSeerX design, and proposed a new design for the NGX with a revised architecture, enhanced hardware, and software framework. In this paper, we extend [23] and focus on several key design considerations for the frontend and backend. We compare different design options and justify our choices based on analytical or experimental results. Our goal is to present these design decisions and articulate the factors to consider when making trade-offs so as to justify our decisions. Although these decisions were made for NGX, we feel such designs will be appropriate in the development of other vertical similar scale search engines.

2 RELATED WORK

Early vertical search engines were relatively small scale due to hardware and software constraints, e.g., the virage image search engine [2]. After 2000, geographic search engines received serious attention from major search companies. In [16], the design considerations of building a geographic search engine are discussed, such as the choice between vector data model and a raster data model. In [13], design considerations are outlined when migrating from CiteSeer to CiteSeerX, where a new data model and stratified architecture is introduced. In [4], different strategies are discussed for search engine caching, although not limited to vertical search engines. Since 2010, medical search engines and large scale DLSEs architectures were proposed. For example, the iMed designers incorporated query interfaces and investigated how to make it easier for users to find answers [15]. Similarly, the designers of Yale Image Finder focused on backend designs featuring customized layout analysis over images published in academic journals [28]. In [19], comparing and making choices of AI technologies was discussed for Arnetminer, a social-oriented academic search engine. In [26], a new framework was proposed that incorporated different design factors aiming at building a scalable and maintainable cloud-based scholarly big data platform. Many ideas were adopted in the current CiteSeerX system. Recently, knowledge graphs are used for powering search engines. For example, Huang et al. discussed design considerations for building an oil gas information intelligent search engine, including the choice of ontology, the named entity recognition model, and integration between Neo4J and Elasticsearch (ES) [11].

Our work is different in the sense we focus on comparing different design considerations when migrating a large-scale production system to a new infrastructure, which is hosted in an academic setting with relatively limited resources. How to balance the usability, maintainability, scalability, and sustainability is challenging. Please note that we use NGX to identify the new system and CiteSeerX to identify the current system.

3 NGX DESIGN OVERVIEW

To support scaling up the document collection and the new design, we deploy the NGX on a new hardware infrastructure, shown in Table 1. The new architecture contains a frontend and a backend.

The frontend includes the web application, implemented using Vue.js, Nuxt.js, Python, JavaScript, HTML, and CSS, which were chosen for more efficient development and maintainability through a component-based design and for better performance. The web application inherited several key features from CiteSeerX, allowing users to search, sort results, browse metadata, and download papers. The CiteSeerX source code was packaged in a hierarchical structure, which can be compiled by Apache Ant. This requires installation of many prerequisite software and setting up the environment, which takes time and may cause unexpected compatibility issues, e.g., operating system (OS) upgrades. In NGX, we containerize our client and server applications using Docker to streamline our deployment process and increase the scalability. Using Docker images also makes it easier to share the framework with the community so it can be deployed in other environments and used for building a DLSE with different corpora. To provide a single-host and secured deployment, we adopted *docker-compose* that runs multiple containers as a single service. Each container runs in isolation but can interact with each other when required. Figure 1 shows the high-level architecture of the client and server application. We implemented secure communications between remote server processes using the SSL protocols of NGINX, an open source software for web serving and other network services.

The backend includes an Extraction-Ingestion System (EIS), which automatically converts PDF documents into a searchable text format, extracts metadata and content components (e.g., text, figures, tables) from documents crawled from the Web, and indexes them. All searchable data are stored in Elasticsearch (ES). The core component of EIS is PDFMEF (PDF Multi-entity Extraction Framework) [22]. It is a customizable and scalable framework for extracting content from scholarly documents. PDFMEF encapsulates various content classifiers and extractors, such as Apache PDFBox, a learning-based academic paper classifier [5], an academic filter

Server	#Instance	Type	#Core	RAM	Storage
Web	x2	Cloud	2	16GB	30GB HDD
ES master	x3	Cloud	2	16GB	154GB HDD
ES data	x4	Physical	16	96GB	1.6TB SSD
EIS	x1	Physical	80	187GB	11TB SSD
Crawling	x1	Physical	24	60GB	30TB HDD
Repository	x2	Virtual	-	-	50TB HDD

Table 1: Major hardware used for deploying NGX. ES: Elasticsearch. EIS: Extraction-Ingestion System. Repository is on a storage array network (NAS) mounted to other servers.

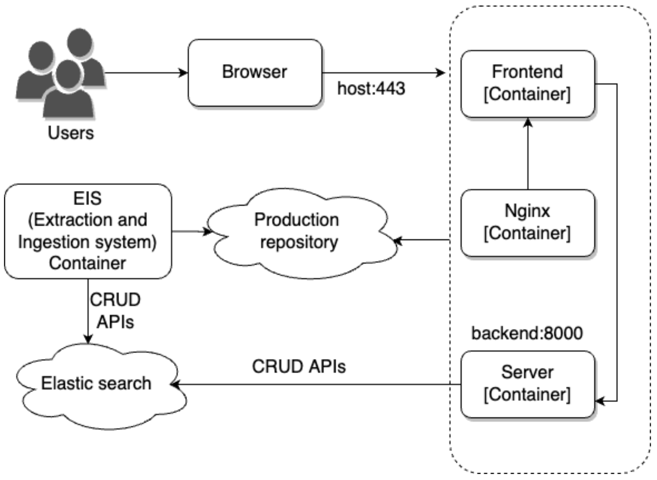

Figure 1: The front-end and back-end architecture of NGX.

[5], and figure extration PDFFIGURES2 [7], all to perform comprehensive information extraction tasks. We employed GROBID [14] for parsing and re-structuring raw documents into structured XML/TEI encoded documents. We then pipelined the XML to extract various types of information. The PDFMEF is containerized and runs as a separate entity. Containerizing PDFMEF allowed us to to scale the service horizontally and run multiple instances of the service as different containers. It also made the service easier and faster to deploy. The EIS framework scaled well for ingestion and, since ingestion to Elasticsearch is a network intensive operation, we used a multi-threaded service that makes full use of system resources. The system was designed to process document batches using all available cores to perform extraction and then ingestion. The ingestion process is coupled together with the extraction process so as soon the extraction service completes, the extracted information is directly ingested to Elasticsearch. The PDF files, renamed using their IDs, are copied to the repository.

4 DESIGN CONSIDERATIONS

4.1 Infrastructure: Physical vs. Cloud

CiteSeerX has been running in a private cloud since around 2013 [25]. The cloud offers several advantages such as elasticity and maintainability. For example, cloning a new virtual machine takes only a few minutes. Note that NGX will host at least 30 million academic documents with a high throughput. Because of this, it needs to be deployed in a distributed system. The first question to ask is whether NGX should be hosted in a private cloud or a physical cluster.

The decision was to host the system in a hybrid infrastructure. Specifically, we host the web servers, three master nodes of the ES cluster, and the static web server in the cloud. These servers carry relatively low computational load and consumes a relatively less IO bandwidth. The performance of these servers is unlikely to be affected by the shared disk channel. We host the EIS server, the web crawling server, and the ES data nodes in a loosely coupled physical cluster. Jobs running on these servers are usually computationally expensive and IO extensive. For example, it was shown that it took more than twice as long to finish a batch crawling job on a virtual machine (VM) versus on a physical server [25].

One trade-off we needed to make was the availability vs. performance because over a long time, disks may fail and one is now at risk of loosing data. The problem is less concerning for VMs because the cloud hypervisor can dynamically allocate resources and send warnings for replacing failed disks. For physical servers, disk failure can be catastrophic and bring the system down. To make the VMs in the cloud more sustainable, we expanded the current cloud infrastructure with new physical servers. Existing hardware in the cloud has been running for about 10 years, so we expect the expanded cloud to last comparably long. To mitigate the potential risks of data loss on the ES data nodes, we employ hot-plug solid state disks (SSDs) instead of hard drives (HDDs) because SSDs have exhibited improved performance and durability [17]. ES also shards data across four data nodes to ensure horizontal scalability and high availability.

4.2 Web Application: Modify vs. Refactor

The CiteSeerX web application was developed on top of Java and JavaScript to automatically generate user interfaces and has been running stably on several VMs for about 10 years. The web application is composed of servlets that correspond to user queries and interact with the index and database servers for keyword-based search, metadata retrieval and browsing, and file download. One consideration is whether to develop the web application by modifying the existing one or refactoring to a entirely new framework.

We decided to refactor the web application. This decision was made after consulting with a chief-technology officer of an insurance website. The decision was also made by trading-off between compatibility and complexity. Modifying the current version may be less complicated as many features have already been implemented. On the other hand, the current application contains several key compatibility issues that must be fixed. The rationales behind this decision include the following.

First, many components used to build the CiteSeerX web application are out of date or not being maintained. For example, CiteSeerX employed `log4j` (version 1.2.17), which has not been supported since 2015. Updating `log4j` will impact all Java programs that rely on this package. Another key issue is the OS version compatible with the web application. RHEL5/6 and CentOS 6, are no longer supported. To continue running the application on these operating systems would expose the application to multiple security vulnerabilities. Although CiteSeerX can be deployed on RHEL7/CentOS 7, the current application only supports 128-bit ciphers. TLS (Transport Layer Security) handshakes using all known versions of 256-bit cipher suites fail with the current version of the web application. Fixing this issue requires us to reexamine all components.

Second, the architecture change calls for significant corresponding changes at the frontend. One major change is the removal of the relational database. The CiteSeerX implementation relied heavily on the Data Access Object (DAO) pattern in Java. The DAO pattern is a structural pattern that allows web developers to isolate the application layer from the persistence layer (the relational database in our case) using an abstract API. Since the database is removed, this design pattern was unnecessary. In addition, the current search index was by Apache Solr but the NGX frontend only interacts with ES for both search and metadata retrieval. Cleaning the DAO pattern and changing the search API from Solr to ES would take a remarkably long time and errors may be accidentally introduced.

4.3 What to Index and Top k

Returning the most relevant documents to users is a key function of a DLSE. CiteSeerX implemented the search function using Apache Solr, which by default used a slightly modified version of BM25 [18]. The algorithm returned decent results within a sub-second time scale. Elasticsearch adopts BM25 as the default search algorithm, but as the collection increases, researchers face the challenge of finding the exact research papers from a massive amount of documents using limited text. One way to overcome this challenge is to use a two-stage ranking mechanism [27]. The first stage is usually a naïve bulk retrieval using BM25, aiming at a high recall by retrieving the top k results and the second stage uses a learning-to-rank model to fine-tune the ranking of documents retrieved in the first stage,

Table 2: Elasticsearch API response time on retrieving top-k documents from a bulky index (metadata + full text) and a leaner index (metadata only). The scenario with the shortest response time is in bold.

Index	k	Time (sec)
metadata + full text	500	1.2
metadata + full text	1000	2.1
metadata	**500**	**0.4**
metadata	1000	0.6

[27, 29] thus increasing the precision. Therefore, determining a proper value of k not only reduces the time spent in the first stage but also on the second stage.

A related design decision is *whether we should index the metadata only or index the metadata and the full text* . The metadata includes titles, authors, years, venues, and abstracts. This decision is unimportant when the document collection is relatively small, but as the number of documents grows to over 30 million - our goal, it is important to consider how to make a trade-off between the users' experience (response time) and information capacity of the search engine, with limited resources.

We decided to index the metadata only and set $k = 500$ in the first retrieval stage, given our infrastructure (see Table 1). We demonstrate this using a proof-of-concept experiment on a simplified setting consisting of a single node server and a client hosted on a separate computer. We compared the API response time of querying against all documents (corresponding to the "match_all" filter) indexed by two ES instances, each built for the same 1 million documents. One instance indexed only the metadata and the other indexed both metadata and full-text. We submitted single keyword queries and requested the API to return the top k ($k = 500$ or 1000) documents with their content. The results shown in Table 2 indicate that indexing metadata only with $k = 500$ took only 1/3 time for indexing metadata and full text. The retrieval time when $k = 500$ is about $57 - -66\%$ of the time when $k = 1000$. In particular, the retrieval time when indexing both metadata and full text exceeded 1 second ($k = 500$) and 2 seconds ($k = 1000$). Therefore, we decided to index the metadata only with $k = 500$. Note that the exact query time depends on the number of terms, the logical operator, and the filters used. The difference of experimental results in Table 2 demonstrates the advantage of our choice under the same setting.

4.4 Which Academic Filter to Use?

Similar to CiteSeerX, NGX also implemented a focused web crawler that actively harvests PDF documents from the Web. However, only a fraction of PDF documents are academic papers and that proportion depends on seed URLs ranging from 30% to 50% [24]. Therefore, an efficient academic filter is crucial to build a relatively clean collection. Previously, CiteSeerX used a rule-based academic filter, which simply checked if the PDF documents contain the word "reference", "bibliography" and their variants. Due to the simplicity of this rule, the F1 measure of this method was only 0.77 and the extraction pipeline introduced many non-academic PDF documents, such as

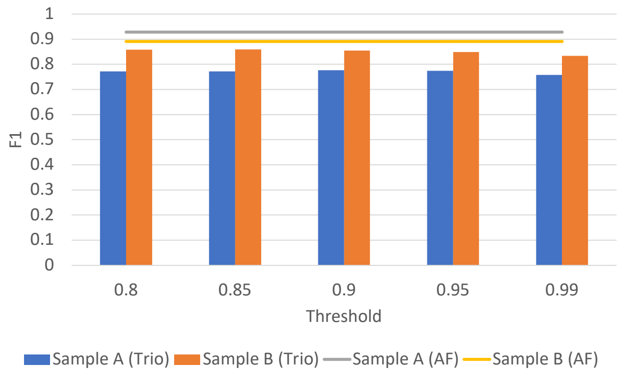

Figure 2: Comparison between the Academic Filter (AF) [5] and Trio on two independent samples drawn from the CiteSeerX crawl repository. The x-axis is the confidence threshold used for Trio to separate two classes. Trio results were obtained by a 5-fold cross validation (CV). AF results are directly quoted from [5]. The AF results for Sample A was obtained by 5-fold CV. The AF results for Sample B was obtained by testing the best model trained on Sample A on Sample B.

curriculum vitae or resumes [6]. To improve this, we developed a machine-learning-based method, incorporating four types of structural features, such as FileSize, PageCount, DocumentLength, etc. This method achieved a much higher performance with F1=0.90 tested on a corpus of randomly selected crawled PDFs [5]. Recently, Internet Archive developed an academic filter Trio [1, 3]. Different from [5], Trio was an ensemble model that incorporates three different classifiers, a BERT-based classifier [8], an image-based classifier, and a linear classifier. Each classifier computes a classification confidence between 0 and 1. The design question is whether we should adopt the academic filter by [5] or Trio.

To answer this question, we compared the performance of these two classifiers on two benchmark datasets [5], both consisting manually labeled PDFs crawled between 2008 and 2013. The difference is that one contains 1009 PDFs and the other contains 1000 PDFs. In the experiment, we test the academic filter by [5] and Trio on a task to classify PDF documents into academic vs. non-academic documents. Here, academic documents include conference papers, journal articles, technical reports, books, theses, dissertations, slides, and abstracts. The performance is shown in Figure 2. The result indicates that the Academic Filter by [5] outperforms Trio by at least 3% (for Sample B) and at least 15% (for Sample A). Therefore, *we decide to continue using the Academic Filter [5].*

5 CONCLUSION

We showcased four design considerations and justified their use. These include a hybrid infrastructure, a refactor of the web application, indexing the metadata only as opposed to the full text, retrieving the top $k = 500$ documents as candidates in a two-stage retrieval system, and employing a machine learning based academic filter based on structural features. These design considerations are not exhaustive but represent key aspects to consider for design of production systems of a similar scale.

ACKNOWLEDGMENTS

We gratefully acknowledge NSF partial support (Award #1823288).

REFERENCES

[1] Internet Archive. 2021. PDF Trio. Retrieved August, 2022 from https://github.com/internetarchive/pdf_trio

[2] Jeffrey R. Bach, Charles Fuller, Amarnath Gupta, Arun Hampapur, Bradley Horowitz, Rich Humphrey, Ramesh C. Jain, and Chiao-Fe Shu. 1996. Virage image search engine: an open framework for image management. In *Storage and Retrieval for Still Image and Video Databases IV*, Ishwar K. Sethi and Ramesh C. Jain (Eds.), Vol. 2670. International Society for Optics and Photonics, SPIE, 76 – 87. https://doi.org/10.1117/12.234785

[3] Paul Baclace. 2020. Making A Production Classifier Ensemble. Retrieved August, 2022 from https://towardsdatascience.com/making-a-production-classifier-ensemble-2d87fbf0f486

[4] Ricardo Baeza-Yates, Aristides Gionis, Flavio P. Junqueira, Vanessa Murdock, Vassilis Plachouras, and Fabrizio Silvestri. 2008. Design Trade-Offs for Search Engine Caching. *ACM Trans. Web* 2, 4, Article 20 (oct 2008), 28 pages. https://doi.org/10.1145/1409220.1409223

[5] Cornelia Caragea, Jian Wu, Sujatha Das Gollapalli, and C. Lee Giles. 2016. Document Type Classification in Online Digital Libraries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. 3997–4002. http://www.aaai.org/ocs/index.php/IAAI/IAAI16/paper/view/12343

[6] Cornelia Caragea, Jian Wu, Kyle Williams, Sujatha Das Gollapalli, Madian Khabsa, and C. Lee Giles. 2014. Automatic Identification of Research Articles from Crawled Documents *(WSDM 2014 Workshop on Web-scale Classification: Classifying Big Data from the Web)*.

[7] Christopher Clark and Santosh Kumar Divvala. 2016. PDFFigures 2.0: Mining Figures from Research Papers. In *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries, JCDL 2016, Newark, NJ, USA, June 19 - 23, 2016*. 143–152. https://doi.org/10.1145/2910896.2910904

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. 4171–4186. https://aclweb.org/anthology/papers/N/N19/N19-1423/

[9] Suzanne Fricke. 2018. Semantic Scholar. *Journal of the Medical Library Association : JMLA* 106, 1 (01 2018), 145–147. https://doi.org/10.5195/jmla.2018.280

[10] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. CiteSeer: An Automatic Citation Indexing System. In *Proceedings of the 3rd ACM International Conference on Digital Libraries, June 23-26, 1998, Pittsburgh, PA, USA*. 89–98. https://doi.org/10.1145/276675.276685

[11] Shujun Huang, Yuyan Wang, and Xiang Yu. 2020. Design and Implementation of Oil and Gas Information on Intelligent Search Engine Based on Knowledge Graph. *Journal of Physics: Conference Series* 1621, 1 (aug 2020), 012010. https://doi.org/10.1088/1742-6596/1621/1/012010

[12] Péter Jacsó. 2005. Google Scholar: the pros and the cons. *Online Information Review* 29, 2 (2022/09/05 2005), 208–214. https://doi.org/10.1108/14684520510598066

[13] Huajing Li, Isaac Councill, Wang-Chien Lee, and C. Lee Giles. 2006. CiteSeerx: An Architecture and Web Service Design for an Academic Document Search Engine. In *Proceedings of the 15th International Conference on World Wide Web* (Edinburgh, Scotland) *(WWW '06)*. Association for Computing Machinery, New York, NY, USA, 883–884. https://doi.org/10.1145/1135777.1135926

[14] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries* (Corfu, Greece) *(ECDL'09)*. Springer-Verlag, Berlin, Heidelberg, 473–474. http://dl.acm.org/citation.cfm?id=1812799.1812875

[15] Gang Luo. 2009. Design and Evaluation of the iMed Intelligent Medical Search Engine. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng (Eds.). IEEE Computer Society, 1379–1390. https://doi.org/10.1109/ICDE.2009.10

[16] Alexander Markowetz, Yen-Yu Chen, Torsten Suel, Xiaohui Long, and Bernhard Seeger. 2005. Design and Implementation of a Geographic Search Engine. In *Proceedings of the Eight International Workshop on the Web & Databases (WebDB 2005), Baltimore, Maryland, USA, Collocated mith ACM SIGMOD/PODS 2005, June 16-17, 2005*, AnHai Doan, Frank Neven, Robert McCann, and Geert Jan Bex (Eds.). 19–24.

[17] Alan R Olson, Denis J Langlois, et al. 2008. Solid state drives data reliability and lifetime. *Imation White Paper* (2008), 1–27.

[18] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389. https://doi.org/10.1561/1500000019

[19] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Ying Li, Bing Liu, and Sunita Sarawagi (Eds.). ACM, 990–998. https://doi.org/10.1145/1401890.1402008

[20] Pradeep B. Teregowda, Isaac G. Councill, R. Juan Pablo Fernández, Madian Khabsa, Shuyi Zheng, and C. Lee Giles. 2010. SeerSuite: Developing a Scalable and Reliable Application Framework for Building Digital Libraries by Crawling the Web. In *Proceedings of the 2010 USENIX Conference on Web Application Development* (Boston, MA) *(WebApps'10)*. USENIX Association, Berkeley, CA, USA, 14–14. http://dl.acm.org/citation.cfm?id=1863166.1863180

[21] Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft Academic Graph: When experts are not enough. *Quantitative Science Studies* 1, 1 (02 2020), 396–413. https://doi.org/10.1162/qss_a_00021 arXiv:https://direct.mit.edu/qss/article-pdf/1/1/396/1760880/qss_a_00021.pdf

[22] Jian Wu, Jason Killian, Huaiyu Yang, Kyle Williams, Sagnik Ray Choudhury, Suppawong Tuarob, Cornelia Caragea, and C. Lee Giles. 2015. PDFMEF: A Multi-Entity Knowledge Extraction Framework for Scholarly Documents and Semantic Search. In *Proceedings of the 8th International Conference on Knowledge Capture* (Palisades, NY, USA) *(K-CAP 2015)*. ACM, New York, NY, USA, Article 13, 8 pages. https://doi.org/10.1145/2815833.2815834

[23] Jian Wu, Shaurya Rohatgi, Sai Raghav Reddy Keesara, Jason Chhay, Kevin Kuo, Arjun Manoj Menon, Sean Parsons, Bhuvan Urgaonkar, and C. Lee Giles. 2021. Building an Accessible, Usable, Scalable, and Sustainable Service for Scholarly Big Data. In *2021 IEEE International Conference on Big Data (Big Data)*. 141–152. https://doi.org/10.1109/BigData52589.2021.9671612

[24] Jian Wu, Pradeep Teregowda, Juan Pablo Fernández Ramírez, Prasenjit Mitra, Shuyi Zheng, and C. Lee Giles. 2012. The Evolution of a Crawling Strategy for an Academic Document Search Engine: Whitelists and Blacklists. In *Proceedings of the 4th Annual ACM Web Science Conference* (Evanston, Illinois) *(WebSci '12)*. ACM, New York, NY, USA, 340–343. https://doi.org/10.1145/2380718.2380762

[25] Jian Wu, Pradeep B. Teregowda, Kyle Williams, Madian Khabsa, Douglas Jordan, Eric Treece, Zhaohui Wu, and C. Lee Giles. 2014. Migrating a Digital Library to a Private Cloud. In *2014 IEEE International Conference on Cloud Engineering, Boston, MA, USA, March 11-14, 2014*. 97–106. https://doi.org/10.1109/IC2E.2014.77

[26] Zhaohui Wu, Jian Wu, Madian Khabsa, Kyle Williams, Hung-Hsuan Chen, Wenyi Huang, Suppawong Tuarob, Sagnik Ray Choudhury, Alexander Ororbia, Prasenjit Mitra, and C. Lee Giles. 2014. Towards building a scholarly big data platform: Challenges, lessons and opportunities. In *IEEE/ACM Joint Conference on Digital Libraries, JCDL 2014, London, United Kingdom, September 8-12, 2014*. 117–126. https://doi.org/10.1109/JCDL.2014.6970157

[27] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 1271–1279. https://doi.org/10.1145/3038912.3052558

[28] Songhua Xu, Jamie P. McCusker, and Michael Krauthammer. 2008. Yale Image Finder (YIF): a new search engine for retrieving biomedical images. *Bioinform.* 24, 17 (2008), 1968–1970. https://doi.org/10.1093/bioinformatics/btn340

[29] Edwin Zhang, Nikhil Gupta, Rodrigo Frassetto Nogueira, Kyunghyun Cho, and Jimmy Lin. 2020. Rapidly Deploying a Neural Search Engine for the COVID-19 Open Research Dataset: Preliminary Thoughts and Lessons Learned. *CoRR* abs/2004.05125 (2020). arXiv:2004.05125 https://arxiv.org/abs/2004.05125