The Pennsylvania State University
The Graduate School

DESIGN AND DATA MINING TECHNIQUES FOR LARGE-SCALE SCHOLARLY DIGITAL LIBRARIES AND SEARCH ENGINES

A Dissertation in
Informatics
by
Shaurya Rohatgi

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2023

The dissertation of Shaurya Rohatgi was reviewed and approved by the following:

C. Lee Giles
David Reese Professor, College of Information Sciences and Technology
Dissertation Advisor, Chair of Committee

Kenneth Huang
Assistant Professor, College of Information Sciences and Technology

Amulya Yadav
Assistant Professor, College of Information Sciences and Technology

Daniel Kifer
Professor, Department of Computer Science and Engineering

Jeffrey Bardzell
Professor of Information Sciences and Technology
Program Head

Abstract

The exponential growth of digital libraries and the proliferation of scholarly content in electronic formats have made data mining and information retrieval essential tools for effectively managing, organizing, and disseminating knowledge. This thesis provides a comprehensive analysis of the advancements and challenges in these fields, with a focus on mathematical information retrieval from scholarly documents, figure captioning and classification of scientific images, and searching and re-ranking techniques for large-scale scholarly documents. We also explore the future of scholarly search, considering the potential roles of generative artificial intelligence and scientific question-answering systems in these domains. In the initial section of this thesis, we delve into the complex design and implementation aspects involved in building a large-scale digital library. We discuss various challenges and critical design decisions that were made to ensure the library's long-term sustainability and ease of maintenance. Furthermore, we provide an in-depth analysis of the unique and robust components of CiteSeerX, including its advanced crawling, extraction, ingestion, and production-ready capabilities. Our focus then shifts to the investigation of search and re-ranking techniques specifically tailored for large-scale scholarly documents. We delve into various approaches for indexing, searching, and ranking vast collections of scientific literature, proposing inventive methods for optimizing their performance and scalability. After successfully implementing our system and achieving an impressive index of over 15 million academic papers, we explore the numerous potential applications and opportunities that can arise from this extensive collection of scholarly articles. Following this, we conduct a thorough examination of cutting-edge mathematical information retrieval techniques for extracting and processing mathematical expressions from scholarly documents. We present an exhaustive review of existing approaches, shedding light on their strengths and weaknesses, and propose innovative methods that significantly enhance the accuracy and efficiency of mathematical information retrieval systems. Subsequently, we discuss a subset of CiteSeerX data that is focused on Computational Linguistics (CL) - The ACL Anthology Corpus. We provide the metadata, full-text, and citation graph for the CL domain. This dataset is then analyzed for deeper insights into the evolving direction of the field and the potential applications that

can be developed from it. One such application is addressing the challenges of figure captioning and classification of scientific images. We analyze state-of-the-art methods for extracting and processing image data from scholarly documents and propose a groundbreaking approach that effectively combines advanced image processing techniques with cutting-edge machine learning algorithms for highly accurate and reliable figure captioning and classification. Lastly, we discuss the future of scholarly search and the role of generative AI in scientific question answering. We envision a question answering system, which looks at the relevant literature and formulates an answer for the researcher's information need. To this end, we investigate the potential of large language models and search for enabling such capabilities and outline the challenges and opportunities that lie ahead in this exciting domain.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to my advisor, Dr. C. Lee Giles, and Co-PI Dr. Jian Wu (Old Dominion University) for their guidance and supervision throughout this project. Working with both of them, I have gained valuable knowledge about publishing and scholarly research. From suggesting which conferences to submit to, to editing my paper drafts, Dr. Giles and Dr. Wu have played a significant role in my work, and I am truly grateful for their contributions. I would like to express my appreciation to Dr. Richard Zanibbi (Rochester Institute of Technology) and his student Behrooz Mansouri for helping me gain a deeper understanding of mathematical information retrieval and explaining the small details and nuances of this under-resourced research area. I am also grateful to my committee members, Dr. Kifer, Dr. Yadav and Dr. Huang for there valuable insights and feedback.

I would like to express my gratitude for the financial support provided by various funding agencies throughout my thesis. This work has been generously supported by the Alfred P. Sloan Foundation under Grant No. G-2017-9827 and the National Science Foundation (USA) under Grant No. IIS-1717997. Additionally, this project was partially financed by the National Science Foundation under Grant No. 1823288. These financial contributions have been invaluable for the successful completion of my research.

In addition to my academic advisors and collaborators, I want to sincerely thank my internship mentor, Dr. Sergey Feldman, for believing in me and providing relentless mentorship. Words cannot express how much I appreciate his guidance during my year long internship at Allen Institute of Artificial Intelligence. I would also like to extend my gratitude to Dr. Waleed Ammar mentoring me in the later stage of my career, preparing me for the industry, and collaborating with me. His advice has been invaluable, and I feel fortunate to have him as a mentor and collaborator.

Also, the students who have contributed to the CiteSeerX project in the past - Kunho, Sai, Bharath, Zeba, Kevin, Jason, Sean, Kavya and Manoj. Thank you for being such amazing collaborators and thank you for your hardwork. Furthermore, I am grateful to the brilliant PhD students in my department — Saranya, Neisarg, Agnese, Adaku, Jason, Michi, Jooyoung, and Ankur — for their constant assistance

and insightful discussions. I also want to thank my friends outside my research, Ana, Sian, Akshay, Brooke, Drew, Rohitha, Meghna, Adway, Anjana, Pranav, Mukund, Sanjana, Pauline, Ashwin and Ruchi for being a friend and well-wisher.

I am deeply grateful to my loving parents in India, Sanjay and Ritu, for their constant encouragement and guidance. Their wisdom and endless support have been instrumental in shaping who I am today. I am also extremely thankful for my guardians in the US, Anurag and Arpita, who have stood by me and provided a strong foundation of love, understanding, and stability. Furthermore, my sister Harsha and brother-in-law Rahul deserve a special mention for their unwavering love and encouragement throughout my life. In addition, I would like to express my gratitude to my supportive and caring cousins, Eera and Enya. They have always taken an interest in my unique hobbies, helping me maintain my sanity and sense of humor during challenging times. Lastly, I cannot thank my partner, Saranya, enough for her steadfast love and belief in me, even during my darkest days. Her unwavering support has given me the hope and confidence I needed to overcome obstacles and persevere.

Chapter 1

Introduction

With the advent of scholarly big data, numerous attempts have been made to develop web-based services such as digital library search engines (DLSEs). However, the design and specifications of an accessible, usable, scalable, and sustainable DLSE have not been adequately represented and discussed in existing literature. We believe that these four characteristics are crucial for delivering a high-quality service for scholarly big data from both user and developer perspectives. This work examines the design, implementation, and operational experiences, as well as the lessons learned from CiteSeerX, a real-world digital library search engine. We evaluate the strengths and weaknesses of the current design and propose a new design that includes a revised architecture, improved hardware, and enhanced software infrastructure. The Alpha version of this new design has been implemented and tested. The updated system replaces MySQL and Apache Solr with a single instance of Elasticsearch, which serves the dual purpose of data storage and search. Elasticsearch operates by automatically distributing data across multiple servers and scaling out on commodity hardware. As the majority of scholarly documents are in PDF format, text and data are extracted from these files and fed into the DLSE. A significant enhancement is the integration of extraction and ingestion processes, resulting in a notable increase in document ingestion speed. The web application has been redeveloped to improve user experience by implementing a learning-to-rank model and providing more sophisticated search tools. Various other aspects of the system have also been enhanced. We believe that our design considerations and experience can be beneficial to researchers and engineers working on planning, designing, and upgrading systems of similar scales and functionalities in the future.

1.1 Mathematical Information Retrieval

Mathematical notations are one of the fundamental tools for design, modelling and analysis in the modern society. The efforts made for the improvement of society in the fields of Science, Technology, Engineering and Mathematics (STEM) are highly dependent on deriving, formulating or manipulating these mathematical expressions. These mathematical expressions include graphs, matrices, linear equations, differential equations etc. and even special objects like theorems, lemmas, functions - from various disciplines. The need of mathematical information in STEM disciplines is vital, but there is a lack of availability of tools and services which index and make this information available to search. This gap also ceases the discovery of mathematical information across domain where the terminology used is different to describe concepts, but math can unify them. For example, a researcher working in bio-medical engineering might find an equation in astronomy useful to study cardiac electrical signals. Here the terminology used will be different by the authors of the related work, but the structure of the math equations used might be very relevant to the researcher. Also, a researcher is very likely to stick to the published material in their domain and not examine other domains to look for answers.

Math literacy is important for the masses. A math search engine will play a vital role in facilitating learning [1]. It can reduce the anxiety in the society associated with math [2]. There are efforts to revise college material related to math. Math is a very cognitively intensive subject which requires students to practice and solve equations by hand to get the desired results. While typing queries in LaTeX can yield some great results on the web, a very few know those notations. For example, consider a high school student searching for $Ax = \lambda x$, it is not very likely that the student will know the LaTeX notation for the symbol λ. Even if the student knew that, the returned results will be mostly syntactically matching the formula, which means the returned results will have the exact same variables used. A slightly different notation for the same expression - $T(v) = \lambda v$ might be a relevant match for the information need for that student. Text search engines have changed the way we learn and explore a discipline. Similarly, a search engine for mathematical expressions can help masses explore common concepts from non-overlapping domains, sharing similar math semantics.

The Web contains a huge amount of mathematical information, which includes

large repositories of technical documents in the form of research papers, tutorials and course materials. This information is needed by both experts (researchers) and non-experts (students) in mathematics [3]. This research enables the public, students and researchers to search for mathematical content in repositories like Wikipedia and in technical document collections such as CiteSeerX [4]. While techniques related to textual search have matured, Mathematical Information Retrieval (MIR) [5, 6] is a relatively new research area. Most search interfaces for scientific article discovery – Google Scholar or arXiv support only textual queries. This implies a lack of ability of searching mathematics in relevant papers. The reason this has not been done before is the challenges which come with how math is different from text.

Search engines which can index and retrieve math and text are called *math-aware* search engines. [7] discuss what is missing from the mathematical information landscape. They highlight ease of exploration of the mathematical ideas as one of the key challenges. There framework informs the design of the proposed work which aims to address these challenges by creating a math-aware version of CiteSeerX that indexes math specifically and supports math formula retrieval. We chose CiteSeerX because it is one of the largest collections of openly available academic articles. We also include math information from Wikipedia, which enables us to serve both experts and non-experts.

1.2 ACL Anthology Corpus

Building scholarly corpora for open research accelerates scientific progress and promotes reproducibility in research by providing researchers with accessible and standardized data resources. Driven by advancements in natural language processing and machine learning technologies, the field of computational linguistics has experienced rapid growth in recent years. As the volume of research in the CL domain continues to expand, there is an increasing need for an open scholarly corpus that facilitates collaboration, knowledge sharing and the development of new methodologies.

The ACL Anthology is a digital archive of conference and journal papers in natural language processing and computational linguistics. Previous scholarly corpora built on ACL Anthology, such as ARC [8] and AAN [9] provide valuable citation and collaboration networks for researchers while still limited by full-text accessibility and data currency. We target their limitations and provide an up-to-date scholarly corpus

with structured full-texts to facilitate text analysis in the CL domain, such as research trend analysis [10, 11].

The proposed OCL corpus includes all conference, journal, and workshop papers hosted by the ACL Anthology starting from 1952 to July 2022. There are 74k peer-reviewed scientific papers, with 88k PDF files[1]. Under the high demand for full-text and logical structures from scientific papers [12], we process PDF files to extract their full-texts together with logical section structures. The results are stored in machine-readable files in JSON format.

1.3 COVIDSeer

The COVID-19 pandemic had brought together a plethora of scientists in various fields, resulting in multitude of research papers on the subject. This large number of papers has made it difficult to keep track of this information, even in this niche domain. This has motivated the development of many search engines[2] and datasets [13] that focus on SARS-CoV-2 and related papers.

The COVID-19 Open Research Dataset (CORD-19) is a collection of papers related to Severe Acute Respiratory Syndrome Coronavirus2 (SARS-CoV-2) pandemic and was first released on March 16, 2020 by the Allen Institute for Artificial Intelligence (AllenAI), in collaboration with their partners.

The initial dataset comprised approximately 28K papers, and has since grown to more than 100K papers through subsequent weekly updates at the time of this writing. The dataset consists of research articles published in response to COVID-19 in 2020 and articles on similar viruses (for example, SARS and MERS) drawn from several sources including PubMed Central (PMC), bioRxiv, and medRxiv preprint servers and the World Health Organization (WHO) Covid-19 Database. Its aim was to connect the computer science community with biomedical domain experts and policy makers to help identify effective treatments and management policies for COVID-19 [14].

The CORD-19 dataset was built by using metadata from publishers such as PMC by extracting text from their PDFs. But missing in the dataset were potential data

[1]One paper may have multiple versions of PDF files and there are slides and posters in this collection as well

[2]`https://discourse.cord-19.semanticscholar.org/t/cord-19-demos-and-resources/132`

sources such as key-phrases, figures, tables, some missing abstracts, and similar paper recommendations. To fill this missing information, we used the tools developed in SeerSuite [15, 16] to enhance the CORD-19 dataset with fields mentioned in Table 5.1. We build upon the COVID-19 Fatcat Snapshot[3] which is a superset of papers in CORD-19. It was released by the Internet Archive (IA) and includes PDF files of these articles. PDFs were NOT included the original CORD-19. PDFs have richer information than just text but extracting it is challenging. We used the tools in SeerSuite[4] to mine such PDF data.

To ease the exploration of this enriched data, we developed a search engine, COVIDSeer[5]. The frontend and backend which will be described later were designed to accommodate regular updates to the dataset. We use GROBID [17] to extract the missing abstracts and PDFFigures [18] to extract the captions, figures, and tables. We use Elasticsearch[6] to index and search the data. The scalable implementation of Lucene in Elasticsearch allows real-time querying and filtering based on authors, years, venues, and key-phrases. In addition, multiple machine learning based tools were used for key-phrase extraction [19] and similar paper recommendations. Work presented here used the CORD-19 dataset released on April 10, 2020 which contains 51,045 scientific publications of potential relevance to coronavirus and other closely related areas in virology, epidemiology, and biology.

We present an overview of the information extraction process followed by a detailed technical description of search architecture and discuss the adopted methodology for extraction and indexing and distinctive features of our information retrieval system.

1.4 Open Domain Scientific Question Answering

Recent research has revealed that advanced auto-regressive large language models (LLMs) [20], such as LlaMa and GPT-4, demonstrate exceptional skill in generating fluent, coherent, and informative natural language texts. This impressive performance can be primarily attributed to the considerable amount of world knowledge encoded within these models, coupled with their ability to generalize effectively from this vast knowledge base. The wealth of world knowledge embedded in these LLMs is derived

[3]https://archive.org/details/covid19_fatcat_20200410
[4]`https://github.com/SeerLabs/CiteSeerX`
[5]COVIDSeer: `https://covidseer.ist.psu.edu/`
[6]`https://www.elastic.co/elasticsearch/`

from the extensive training data they are exposed to during development. This data allows the models to learn and capture a wide range of information, enabling them to generate high-quality, contextually relevant responses when prompted with various queries or tasks. Furthermore, their capacity to generalize allows them to make inferences and draw conclusions based on the knowledge they have acquired, making them incredibly versatile and valuable tools for a variety of applications. However, it is crucial to acknowledge that the knowledge encoding process within LLMs is not perfect and is, in fact, subject to lossiness. This means that some information may be missing or inaccurately represented within the models, which can potentially limit their performance or lead to the generation of erroneous or misleading content. Additionally, the process of generalization, while powerful, can sometimes result in hallucinations, where the models generate outputs that are not well-grounded in their knowledge base or are based on incorrect assumptions.

In tasks such as scientific question answering, these hallucinations can cause confusion. For instance, ChatGPT generated a fabricated citation within a text it produced. This led researchers to believe that the article existed and was authored by another researcher. Consequently, they contacted the actual researcher working in that field, who had to politely clarify that the citation was false and they had not written it. The drawbacks of hallucinations in the LLMs can be overcome by grounded generation [21]. Recent work has shown that if we augment more context and provide LLMs with external knowledge relevant to the questions asked the hallucinations can be greatly decreased. They also demonstrate that the usefulness of the LLMs can also be increased when augmented with relevant knowledge. In this paper, we introduce S2QA, a system designed to enhance the scientific question-answering capabilities of GPT-4 by incorporating knowledge through scientific information retrieval. As illustrated in Figure TODO, the user inputs a query, which is then utilized for search and re-ranking in order to augment the context for the Language Model (LLM). Later, this enriched context is employed as a prompt for the LLM to generate an accurate answer to the question. In addition to this, S2QA also provides the source of the paper from which it generates the answer. We also present a preliminary assessment of the system's performance on the PubmedQA task. Our approach, even without utilizing the context in the dataset, can surpass the performance of numerous fine-tuned models specifically trained for this task.

Chapter 2

Designing a sustainable Digital Library Search Engine

2.1 Introduction

Digital libraries serve as information systems that typically index extensive collections of digitized documents and offer web-based or API services. These services help users search, access, browse, and download documents stored either locally or remotely. Like numerous other big data systems [22], digital libraries comprise various modules responsible for different tasks. To offer high-quality services, digital libraries need to be accessible, usable, scalable, and sustainable. Accessibility means that all or part of the services must continue to be available even during unexpected failures, such as a web server going down. Usability involves providing multiple interfaces for users to access data, including a user-friendly web interface, a search API, and an OAI-PMH service (Open Archives Initiative Protocol for Metadata Harvesting) [23]. Scalability implies that the system can ingest a large number of documents and accommodate high volumes of user traffic while maintaining reasonable response times. Lastly, sustainability ensures that the system operates with relatively low costs in terms of hardware, software, and maintenance.

Many institutional digital libraries utilize readily available open-source frameworks like *DSpace*, *Fedora Commons*, and *Blacklight*. DSpace and Fedora Commons offer underlying architecture for digital repositories; however, they lack comprehensive management, indexing, discovery, and delivery applications. Blacklight, a Ruby

on Rails engine, generates search interfaces on top of Apache Solr. Despite the ease of deploying an out-of-the-box web application, designing the ingestion pipeline and customizing the discovery interface (e.g., identifying similar papers) is not straightforward. Commercial digital library services such as Google Scholar, Microsoft Academic, and Semantic Scholar depend on parent search engines and/or contractual data services with publishers; hence, they are unlikely to be suitable as a general public framework. These digital library services tend to have high operating costs, making them unaffordable for institutions with limited budgets.

In this work, we share the experiences and lessons learned from operating CiteSeerX - a real-world Digital Library System Ecosystem (DLSE) that launched 20 years ago and has consistently evolved and improved since then. Contrary to commercial digital library services, CiteSeerX has been developed and maintained within an academic setting, with limited budgets and human resources. The source code is open-source[1]. The current system is relatively stable, featuring both advantages and disadvantages. We investigated the architecture and infrastructure, identifying the challenges faced in providing reliable service for a vast volume of scholarly big data. To address these challenges, we designed a new architecture that fulfills the aforementioned requirements, incorporating open-source software frameworks and customized workflows. It is essential to note that this is an experience work, and we are not presenting new research. Instead, we describe the unique perspective of our academic team on the engineering choices made over more than a decade to sustain and develop a system catering to real users.

2.2 Related Works

The origin of Digital Library Search Engines (DLSEs) can be traced back to the late 20th century. CiteSeer, launched in 1998 and later renamed "CiteSeerX" in 2008, was a pioneer in implementing automated indexing techniques to create a network of research papers, enabling users to find related papers using citation graphs [24]. Google Scholar (GS) was introduced in 2004, with its index derived from a crawl of full-text journal content provided by commercial and open access publishers, as well as author homepages [25]. As a comprehensive digital library, GS has become a major data source for scientists and evaluators to assess academic

[1] `https://github.com/SeerLabs/CiteSeerX`

contributions using citation counts and the h-index. Microsoft introduced Windows Live Academic Search around 2006, which later evolved into Microsoft Academic (MA) [26,27]. MA has claimed to have 248 million publications indexed[2]. In 2015, the Allen Institute for Artificial Intelligence (AllenAI) launched Semantic Scholar (S2), with 196 million paper records indexed at the time of writing. S2 integrates various AI techniques, including machine learning (ML), deep learning (DL), and natural language processing (NLP). Another example featuring modern NLP techniques is IBM's Science Summarizer [28]. Numerous digital library infrastructures have been built for specific communities, such as arXiv [29] and its sister websites (e.g., bioXiv, AgriRXiv, and MedRxiv), NASA's Astrophysics Data System (ADS), National Library of Medicine's PubMed [30], and Department of Energy's Office of Scientific and Technical Information (OSTI) system. Another type of DLSEs functions as a mixture of digital libraries and social networking websites, soliciting documents from users and offering various social networking features. Examples of this type include ResearchGate [31] and AMiner [32].

Although numerous DLSEs are available online, most publications on case studies and comparative studies tend to focus on data services and not advanced architectures, development, and operation of these systems (e.g., [33]). Entlich et al. (1997) described the architecture and content of the Chemical Online Retrieval Experiment (CORE) project, a library of primary journal articles in chemistry [34]. Lim and Lu (1999) discussed Harp, a distributed query system for legacy public libraries based on relational databases [35]. Lagoze et al. (2002) reported the technical and organizational infrastructure of the National Science, Mathematics, Engineering, and Technology Education Digital Library (NSDL) [36]. NSDL primarily supported metadata discovery of digital items through an Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) portal, with content accessed using open network protocols such as HTTP or FTP. The architecture and infrastructure of CiteSeerX are described in [37, 38]. Wu et al. (2014) introduced a big data platform for harvesting scholarly information and enabling efficient scholarly applications within a private cloud, outlining the basic components of the current CiteSeerX system [39]. The architecture of AMiner (previously called ArnetMiner) consists of five main components: extraction, integration, storage and access, modeling, and search, as shown in [40]. The system featured an author profile generator and used MySQL

[2] `https://academic.microsoft.com/home`

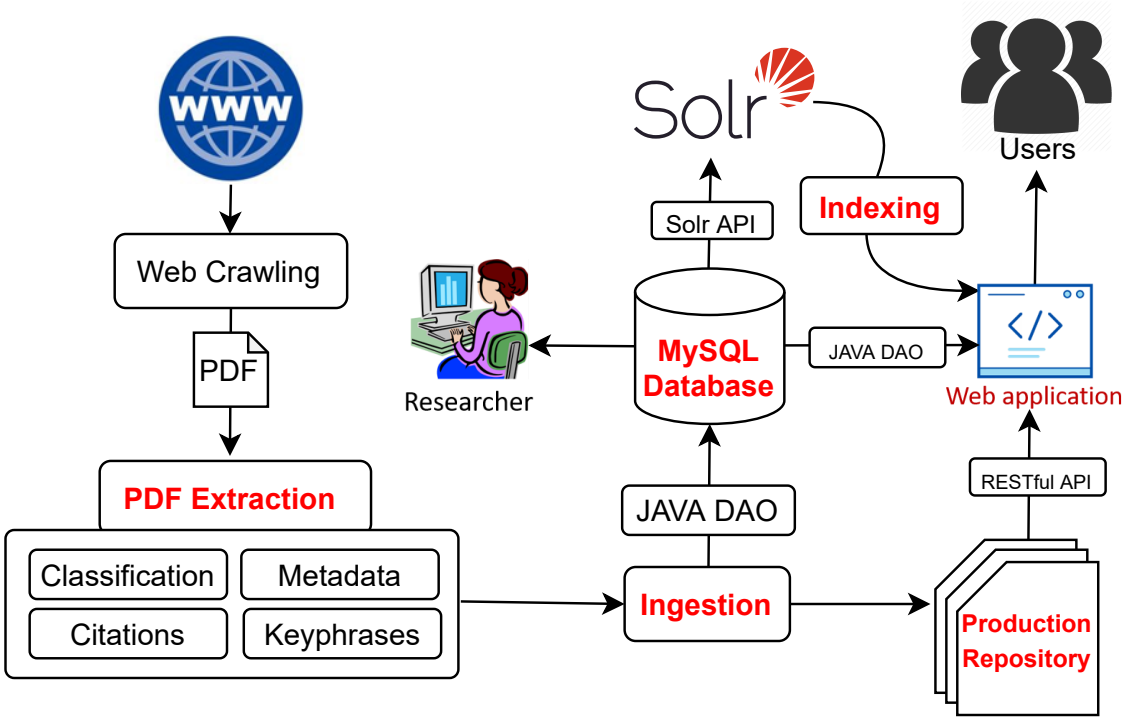


Figure 2.1: The high-level architecture of the existing system. Components marked in red are targeted for significant improvement in the new system.

for storage and indexing. Xia et al. (2017) reviewed big scholarly data management systems and depicted a general architecture of a digital library, comprising modules such as web crawling, document extraction, information extraction, filtering and categorization, database, repositories, data discovery, sharing, and data analysis [41]. However, there are no papers reporting the architecture and infrastructure of well-known DLSEs like GS, MA, and S2. This may be because many DLSE designs are proprietary and not for general purposes, or due to the high cost of maintaining the hardware and software infrastructure. Nonetheless, understanding and investigating the architecture of an information retrieval system can help in assessing the usability of output data by examining how data are manipulated throughout the pipeline. The experiences and lessons learned can also benefit future projects in building more robust and reliable systems.

2.3 Design and Utilization of the Existing System

2.3.1 Overview of Architecture

The present system as shown in Figure 2.1, represents a typical mid-size crawl-based digital library search engine (DLSE) [41]. Raw documents are acquired by actively crawling open access (OA) PDFs from the Web. These PDFs are then converted to text, categorized, and only academic documents are retained. A series of learning-based extractors are employed for extracting full text, metadata, citations, keyphrases, acknowledgments, and non-textual information, such as figures and tables. Metadata is entered into a relational database (MySQL), and various file types are saved into the production repository. The metadata in MySQL and the full text from the repository are indexed by Apache Solr, which powers the search function. The system offers a web-based user interface, an OAI-PMH API, and a database dump on Google Drive for research purposes.

2.3.2 Evolution of the System

The development history of our system can be divided into three phases [42]. In the first phase, a prototype was created around 1998. Major components included a web crawler, an indexer, and a search API written in Perl or C++, along with custom indexing and storage routines implementing the automatic citation indexing technique [24]. Seed URLs were manually curated homepages of computer scientists. The whole system was hosted on a single machine, with relatively low overall traffic but an upward trend.

During the second phase, the search engine expanded to a multi-server system. This involved adding multiple web servers with a front-end load balancer to enhance availability and scalability for higher incoming traffic. Apache Lucene, and later, Apache Solr, were introduced as the main indexer, making the search function more scalable and stable. The backbone software was rewritten in Java, and the web application was developed using a model-view-controller (MVC) architecture. Front-end user interfaces were generated using a mix of Java server pages and JavaScript. The web application was composed of servlets that interacted with the index and database for keyword search and used the Data Access Objects (DAO) to interface with databases and the repository. Metadata extraction was built using Perl and

operated in batch mode. The ingestion system fed data into a MySQL database and global network block device (GNBD), which was shared by web servers. The GNBD relied on the Global File System (GFS) as its file system, and documents were retrieved using an incremental web crawler developed with the Django framework.

The system functioned on a cluster of loosely coupled physical servers, housing nearly a million documents, and managed all incoming traffic during that time. However, after 4-5 years, the hardware became outdated and prone to failure, particularly with regard to hard drives and RAID controllers. Single hard drive failures could be restored due to the configuration of the disk arrays with RAID 5, but multiple hard drive and RAID controller failures could still occur and potentially lead to disaster. Simultaneously, the GFS system, along with the GNBD, often unexpectedly fenced out web servers and/or repository servers, causing the entire system to shut down. Scaling up the physical servers using commodity hardware was also inflexible.

In the third and current phase, the search engine migrated to a private cloud [38]. The software used was inherited from the second phase. Servers in the private cloud were monitored by ESXi, a virtual environment hypervisor developed by VMware for deploying and managing virtual machines. This substantial infrastructure upgrade made the entire system more elastic and easier to monitor. The infrastructure, operation, usage patterns, and research applications are further detailed in the following subsections.

2.3.3 Hardware and Software Infrastructure

The present infrastructure comprises three layers: storage, processing, and application. The *storage layer* consists of two physical servers dedicated to storing all types of data utilized by virtual machines (VMs). Each server features 12 cores, 32GB RAM, and 30TB SATA hard drives (HDDs). The *processing layer* includes five high-end servers connected to the storage layer, each having 12 cores, 96GB RAM, and 1TB SATA HDDs. The *application layer* features various VMs monitored by VMware ESXi 6. By employing a template-based workflow in a virtual architecture, setting up new VMs is reduced from a day to just minutes. All frontend and backend servers are VMs except for the web crawler server, which is hosted on a physical server to reduce completion time for batch crawls.

The system uses *heartbeat-ldirectord* for load balancing, providing the Linux Virtual

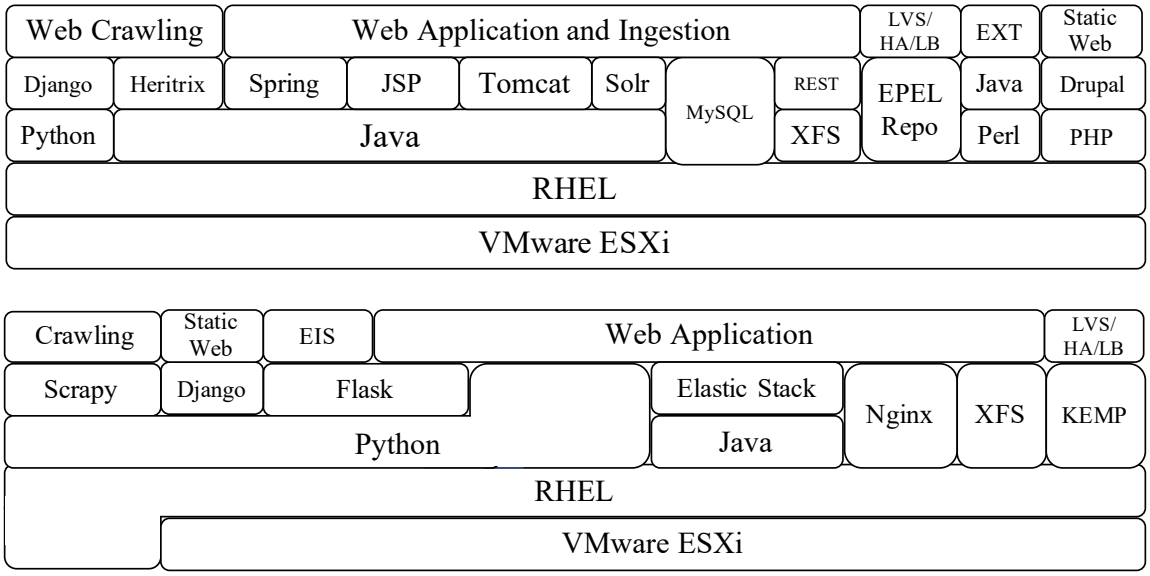


Figure 2.2: The software stack for the current system (listed at the top) and the new system (listed at the bottom) consists of various components. Synonyms are as follows: LVS – Linux virtual service, HA – high availability, LB – load balancer, EXT – extraction, RHEL – Red Hat Enterprise Linux.

Service (LVS) that carries the virtual IP of the landing page. Our system employs Heritrix for web crawling, Tomcat for web service, MySQL for database management, Solr for search, Apache PDFBox for text extraction, and several AI-powered software packages for classification [43], metadata extraction [44], near-duplicate detection [45], and author name disambiguation [46]. Additionally, the system provides APIs based on SOAP/WSDL, which exposes all functionalities for programmatic access. The system's search API enables software programs to access search results using Atom or RSS feeds through an OAI interface. The XFS file system is used to implement a RESTful API retrieving files from the repository server to the web server. Users can access these data and perform searching, browsing, and downloading through a web-based interface. Django and Drupal frameworks were used to build websites displaying web crawling statistics. The software stack for the current system is shown in Figure 2.2.

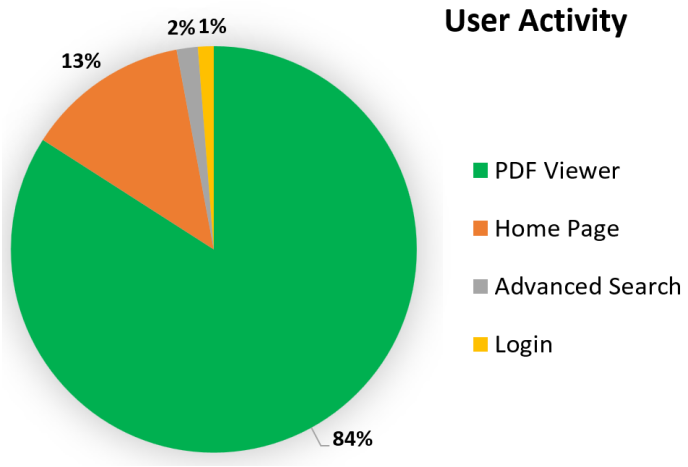


Figure 2.3: Usage based on page views from July 2017 to December 2020. Search engines primarily direct users to the PDF viewer page for an article, indexing the text from the article hosted by our system.

2.3.4 User Behavior

Between 2017 and 2020, the current system served over 23 million page views, with more than 18 million unique page views (Figure 2.3). Over 9 million unique users were served, with over 1 million returning users. The timeline of usage in terms of the number of visits for the top-5 countries is depicted in Figure 2.4. The number of US users peaked at 73,271 in April 2018. The US generally has the highest number of visitors, followed by China and India.

The-existing system has facilitated numerous research and educational activities by making research documents, associated metadata, and frameworks publicly accessible. Every week, the system receives requests for metadata and other datasets, primarily from research groups. The OAI is accessed approximately 100,000 times daily, and the data has been used for diverse research topics such as document type classification [43], keyphrase extraction [47], citation recommendation [48], collaborator recommendation [49], and citation context recommendation [50].

2.3.5 System Availability and Security

High availability and security are crucial for providing a 24/7 online service, and several strategies have been adopted to address these concerns.

Load Balancing The entry point of the system includes a pair of twin load

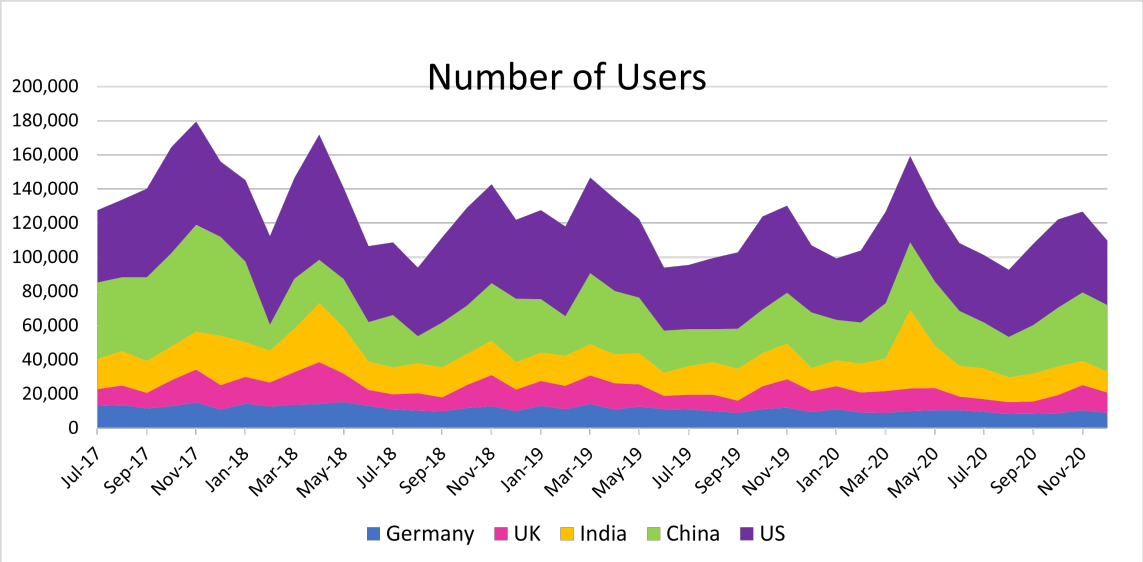


Figure 2.4: Number of users from thetop-5 countries based on the number of visits from June 2017 to December 2020. Usage spikes are observed during April and November of each year, coinciding with the end of semesters in most US universities.

balancers: one active and the other in standby mode. If the active load balancer fails, the standby load balancer automatically takes over the virtual IP and the traffic. All servers are protected behind an institutional firewall, preventing exposure of physical IPs or MAC addresses to external traffic. The system comprises three web servers with identical hardware specifications and web application deployment. The load balancer distributes incoming traffic across the three web servers using the weighted least connection algorithm.

Data Redundancy The system has three database servers, with two slaves replicating a master. The master database handles all metadata lookups, while one of its replicas supports the OAI service. The other replica remains on standby in case the master fails. The system also has two Solr servers in a master-slave configuration. Additionally, a backup repository periodically syncs with the master repository, and when the master repository fails, the backup operates in read-only mode. All disks are configured with RAID 5, allowing for one hard drive failure in a disk array. RAID 5's distributed parity spreads the stress of a dedicated parity disk among all RAID members, increasing write performance since all RAID members participate in serving write requests.

Traffic Control Iptables is a command-line firewall utility that uses policy chains

to allow or block traffic. When a connection attempts to establish itself on the system, iptables searches for a matching rule in its list. If not found, iptables defaults to a predefined action. We apply a series of sophisticated Iptables rules to the load balancer and web servers. One of its functions is to block certain IP addresses that attempt to send brute-force requests without respecting robots.txt, a plain text file indicating which pages or files a crawler can or cannot request from a website. By default, the system applies a threshold of up to 3,000 downloads from a single IP address, deemed sufficient for individual users. The web application controller maintains an allow-list for certain user-agents, such as *googlebot*, providing unlimited access. We also specify the "Crawl-delay" in the robots.txt file to control the request rates from web crawlers. In addition, the web application utilizes a custom module to filter out any tags embedded in the queries, preventing the injection of malicious code into the system, such as cross-site scripting (XSS).

Activity Monitoring A dedicated VM is set up to periodically send direct requests to the web, database, index, and repository servers. If the monitor does not receive the expected responses within a specified time (which varies depending on the type of servers), it will email administrators to resolve the issues. This method has proved effective in alerting administrators of high traffic volumes, Linux kernel panics, or hardware failures.

These strategies have proven useful in achieving high availability and security. However, some needed to be adjusted to accommodate changes in the new system's architecture, as detailed later. The load balancer was replaced with a hardware load balancer, which is more reliable. Due to the adoption of Elasticsearch, data are shared and distributed across multiple servers, enhancing system resilience against server failures. RAID configurations remain necessary for most servers, except index servers which already provide data redundancy through data sharding. RAID 10 will be adopted for certain mission-critical servers with many HDDs in the array, such as the extraction server, allowing a maximum of two drives to fail.

2.4 Pros and Cons of the Current System

2.4.1 Strengths

The present architecture of the system offers several advantages. Primarily, it is built on the LAMP (Linux-Apache-MySQL-PHP) architecture, with Apache replaced by Tomcat and PHP replaced by JavaScript and Java. The widespread adoption of the MVC design pattern has demonstrated its reliability for web applications of varying sizes. The primary tools utilized are open-source and boast a large user base, making it relatively straightforward to find solutions to technical issues on online forums such as Stackoverflow.com.

Regarding the backend, metadata and citation information are first ingested into MySQL, followed by Solr, which serves as a natural extension of the standard LAMP architecture. The role of a search platform like Solr is essential for a DLSE since MySQL cannot efficiently manage full-text queries. Separating data storage and search was a practical approach when managing data sizes of up to 10 million documents. As explained in Section 2.3.5, the current system employs various methods to enhance redundancy, thus avoiding single points of failure.

In the web-crawling cluster, document metadata are standardized, enabling the system to import data harvested in different formats and schema outputs. Storing the original downloading URLs is necessary for both re-crawling and legal purposes, which helps users file Digital Millennium Copyright Act (DMCA) claims.

2.4.2 Weaknesses

However, the existing architecture also exhibits several shortcomings. While some of them may be tolerable in the short term, neglecting them in the long run can have severe consequences, such as permanent data loss or a subpar user experience. The primary challenges stem from the ever-increasing size and growth rate of academic documents. Table 2.1 outlines the current and projected file counts.

One of the main issues lies in the ingestion module, the key bottleneck hampering document collection. The throughput of the web crawler, the extraction module, and the ingestion module stand at approximately 500k, 200k, and 50k documents per day, respectively. The slower ingestion system can be attributed to its single-threaded nature and the fact that the near-duplication detection module must first read from

Table 2.1: The current and projected data size. M=million.

Type	Current	Future	Notes
Full text papers	10M	35M	
Database size	550GB	1.8TB	MySQL
Largest table (rows)	250M	875M	
Database dump	300GB	900GB	.sql
Database buffer	38GB	128GB	10% cached
Indexed records	70M	245M	Apache Solr
Index size	360GB	1.2TB	
Index heap	36GB	120GB	10% cached
Repository	15TB	53TB	File system
PDF	10TB	35TB	
TXT	900GB	3TB	
XML	400GB	1.4TB	
Figures	10TB	35TB	Estimated
Crawl	43TB	150TB	Estimated

the MySQL database. The hit rate decreases as data size grows, which further slows down access.

Another concern arises when the collection size reaches 35 million, roughly the number of OA academic documents available online in 2014 [51]. With the MySQL database expanding to 1.8TB, the citation graph would include at least 245 million nodes and almost 1 billion edges. To maintain a decent hit rate, MySQL requires the hosting server to have at least 200GB of memory for caching a minimum of 10

The scalability issue impacts Solr as well. While Solr can easily scale up to over 80 million documents on a single server, increasing its memory heap may lead to a drop in performance due to garbage collection [52]. The current master-slave architecture is difficult to scale out, and SolrCloud necessitates ZooKeeper, which poses a significant overhead as the two are not well-integrated.

The growing data size also affects data backup and sharing processes. As shown in Table 2.1, the download service necessary to support all OA documents could amount to at least 53TB. Coupled with the database (1.8TB), the index (1.2TB), and a backup for each component, the total required disk space is approximately 109TB. Although feasible with a regular server, enabling such a repository to share and back up data is no simple task. The backup process for the entire repository can take weeks. Moreover, each document is linked to a variety of other file types such as TXT and XML, aggregating to at least $35 \times 4 = 140$ million files and folders. The current production repository stores all types of files linked to an academic document in a single directory. While this method enables convenient file access, navigating the entire repository and exporting specific file types (e.g., TXT only) can consume an exorbitant amount of time.

The frontend of the system also warrants improvement. The search engine relies on Apache Solr's default ranking function based on term frequency-inverse document frequency (TF-IDF). Although this algorithm efficiently captures matching terms for queries, it suffers from low relevancy for ambiguous search operations. The web-based user interface consists solely of a search box, offering no additional options to help users refine their search results based on subject categories [53, 54] or other properties.

From a developer's perspective, the system contains complex dependencies, and some software packages are no longer supported (e.g., Perl packages). Furthermore, despite the highly organized structure of the current code, the DAO usage contributes to an overly rigid and deep hierarchical structure, making it challenging to implement new features or modify existing functionality. Specifically, the ingestion module is tightly bound to the controller, rendering it difficult to unravel the numerous dependencies and parallelize the ingestion process.

2.5 Designing a New System

2.5.1 Design Considerations

To address the limitations of the current system and fulfill the four requirements outlined in Section 2.1, the new system was designed with the following core considerations: It should operate cost-effectively for a minimum of 10 years, even in the absence of catastrophic natural disasters; it should meet the needs of a growing

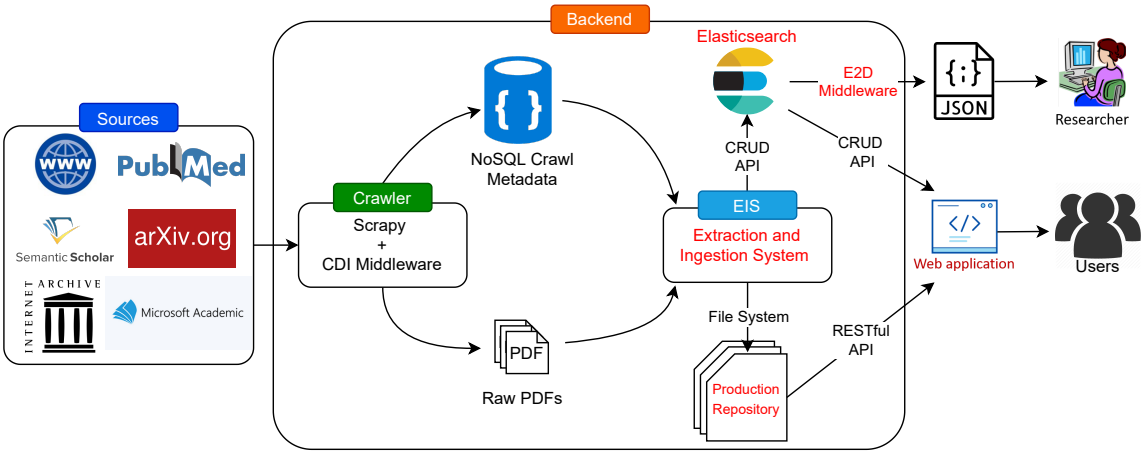


Figure 2.5: The architecture of the new digital library system.

user population while continuing to support scholarly big data for a wide research community; and it should offer a dependable service with enriched data through easy-to-use interfaces. Major changes implemented to achieve these goals include:

1. Developing a scalable web crawler capable of retrieving the majority (if not all) of OA academic documents.
2. Transitioning to a more scalable (potentially NoSQL) database system.
3. Enhancing and parallelizing extraction and ingestion processes with upgraded hardware to accelerate these operations.
4. Dividing the repository into specialized subrepositories for specific file types.
5. Redesigning the frontend to interact seamlessly with the new backend, providing users with additional search and navigation capabilities. Additionally, simplifying the code structure for easier future development and maintenance.
6. Balancing redundancy, cost, and complexity to achieve optimal availability and security levels.

2.5.2 Architecture

The detailed architecture of the new system is depicted in Figure 2.5. Notable changes incorporated into the new design are as follows: (1) A Scrapy-based breadth-first

parallel crawler has been developed to efficiently harvest PDF documents directly from established digital repositories such as Microsoft Academic Graph (OAG) and the Internet Archive Scholar. The CDI middleware [55] reformats crawl metadata into a standardized format, which is then fed into the Extraction and Ingestion System (EIS). (2) The EIS system houses a customizable and parallel extraction module that processes PDF documents, and outputs metadata, citations, figures and tables, mathematical formulas, and keyphrases, which are ingested directly into Elasticsearch and the production repository, reducing disk I/O by minimizing the number of intermediate files. (3) The web application establishes seamless interaction with Elasticsearch, simultaneously handling search and metadata retrieval requests. Elasticsearch, in turn, facilitates data fetching for the database used exclusively for research purposes.

2.5.2.1 Web Crawler

In order to obtain high-quality seed URLs, we rely on verified sources such as S2, MAG, arXiv.org, PubMed, Internet Archive, and other .edu domains, which offer an abundance of rich academic documents. This enables the crawler to download more valuable PDFs and fewer irrelevant ones.

We designed a multi-threaded, high-throughput Scrapy-based crawler that leverages up to 48 processes to download approximately 1 million PDF documents in 7-8 hours. The crawler subsequently stores the metadata, including source URLs, timestamps, and checksums (encrypted using SHA1), in the Elasticsearch NoSQL datastore. It is essential to note that even when crawling URLs from these reputable sources, a significant number of non-academic documents, such as presentations, books, and resumes, are still acquired. To filter out these unrelated documents, we employ a fundamental academic filter [56].

2.5.2.2 Enhanced Extraction and Ingestion System (EIS)

The EIS framework is designed to convert research-intensive academic documents in PDF format into searchable and easily accessible formats. This involves the extraction, clustering, and ingestion of individual documents into Elasticsearch, a highly-scalable and powerful search engine.

The EIS framework is built around PDFMEF (PDF Multi-entity Extraction

Framework), a highly customizable and scalable platform for extracting various types of information from scholarly documents [57]. PDFMEF is designed to work with multiple content classifiers and extractors, such as Apache PDFBox, a learning-based academic paper classifier, and pdffigures2 [58], ensuring comprehensive information extraction. We utilize GROBID [59] for parsing and restructuring raw documents into structured XML/TEI-encoded documents, which are then pipelined into extraction parsers for extracting an extensive range of information.

The EIS system begins with the crawled metadata containing PDF files' locations, which are then processed by PDFMEF to ingest them into the scholarly system. The following modules describe the EIS framework components:

Extraction In this module, PDFMEF extracts various entity types such as metadata, authors, citations, and figures from PDF documents. The code is containerized to allow for parallel extraction of elements like entities, math equations, and figures. It is adapted to run asynchronously in separate containers in batches to maintain consistent processing rates for larger and more complex document batches.

Clustering (aka deduplication or conflation) This module focuses on identifying near-duplicate paper records and citations in other papers. Utilizing the key-matching algorithm [60], we achieve near-real-time ingestion by indexing keys generated by concatenating title snippets and author names, which are used to match documents. This considerably increases extraction and ingestion throughput, processing about one million documents daily. Candidate clusters are further matched using similarity metrics from [56].

Index Schema Our index schema enables efficient searching performance by organizing all data into a single index, ensuring search results are free of near-duplicates. It also stores citation relation information in the 'cited_by' field, containing a list of paper IDs cited by the current cluster, simplifying the retrieval of clusters.

Ingestion During ingestion, the paper data and citation data are converted into entities that can be consumed by the Elasticsearch API. With a single index solution, each indexed document represents a document cluster containing both metadata and citation relationships. PDFs are renamed by their SHA1 values, stored at specific locations in the repository server, and accessed through a REST API via the search interface for end users.

Scalability EIS boasts excellent scalability for ingestion throughput requirements, benefitting from multi-core servers and higher batch sizes. The best throughput was

observed when processing 1,000 documents per batch with an ingestion rate of about 1 million PDFs per day on a machine equipped with dual AMD EPYC 7702P 64-Core hyperthreaded processors.

2.5.2.3 Why Elasticsearch?

Comparing Elasticsearch with other NoSQL datastores, namely Apache Solr and MongoDB, we found Elasticsearch to be superior in providing better horizontal scalability, documentation, and integration with visualization tools like Kibana and log analysis tools like Logstash. Although indexing speed is slightly slower than its counterparts, it still operates within an acceptable range.

2.5.2.4 Repository Architecture

We have developed a RESTful API to retrieve documents from an XFS repository to web servers, using an *varnish* to cache frequently downloaded documents automatically. This stable and easily deployable system solution compartmentalizes the repository into sub-repositories, which are stored in a storage-area network (SAN) hosted in our institution's data center. The web servers access the SAN storage through iSCSI, providing a highly scalable and multi-path accessible architecture.

In our new design, users make modification and correction requests, and metadata updates are performed on the backend. The read-only repository ensures incremental backup by merely tracking new documents.

2.5.3 Challenges and Enhancements

Developing the ingestion-time clustering and near-duplicate detection for a pre-existing paper presents various challenges. To prevent any impact on ingestion throughput, the system must determine in real-time whether a newly-crawled PDF is a near-duplicate of an existing one. While using the Simhash and Hamming distance approach [61] would be too complex and inefficient for our needs, we developed an innovative parallel version of the Key Mapping algorithm [60]. This new algorithm stores key-to-cluster mappings and matches candidate clusters with a similarity metric to avoid false positives. The parallel version also prevents creating duplicate PDFs during ingestion, as it leverages Elasticsearch's GET-by-id API to ensure real-time access to the keys

Table 2.2: Server hardware specifications for various functionalities are provided. The term vHDD refers to virtual hard drives. Only production servers are included in the list. All vHDDs utilize RAID 5 configuration.

Function	**Type (#)**	**Main Specifications**			
		#Core	**RAM**	**Storage**	**Disk type**
Web	VM (x3)	2	5GB	1TB	vHDD
Database	VM (x2)	4	16GB	3TB	vHDD
Index	PH (x2)	16	96GB	4TB	SSD[1]
	PH (x2)	16	64GB	8TB	HDD[1]
	VM (x3)	4	16GB	200GB	vHDD
Extraction	PH (x1)	40	196GB	4TB	SSD[2]
Repository	SAN	–	–	50TB	vHDD

[1] No RAID. [2]After RAID 10.

of the first PDF clustered. Furthermore, we plan to explore additional algorithms such as Locality Sensitive Hashing [62] and Bloom's filter [63] for this task.

In terms of backend API development, multiple revisions were necessary to determine the appropriate interaction with Elasticsearch while still allowing frontend requests. Although Django was considered, it is more suitable for SQL database-stored objects, as opposed to NoSQL servers like Elasticsearch. Consequently, we opted for FastAPI as it better aligns with our general-purpose needs.

The deployment of the new system using Docker containers also posed some challenges. To ensure that the backend API remains accessible when the web UI is down, we moved both the frontend and backend services into Docker containers. An Nginx proxy server routes requests to either the web application or the API service depending on the URL, allowing users to access data via the web UI or API through the same port. Specifications of our hardware hosting all these services are showcased in the Table 2.2

2.6 Re-Ranking through Dense Vector Retrieval in Large-Scale Digital Libraries

Our technique uses a standard cross encoder model, where the dense vector embeddings of the papers and the titles are pre-calculated and stored in a vector index.

With the rapidly growing volume of scientific literature and digitized resources, search functionality in large-scale digital libraries has become essential in retrieving relevant information. Traditional Information Retrieval (IR) systems and ranking methods rely on sparse vector space models such as Bag-of-Words (BoW) or term frequency-inverse document frequency (TF-IDF) and BM25 scoring for indexing and searching documents. However, these techniques may have inherent limitations in capturing the semantic relationships between search queries and documents. In this chapter we discuss a novel re-ranking method incorporating dense vector retrieval to enhance the retrieval performance in large-scale digital libraries.

We introduce a two-stage mechanism, where initially, the documents are fetched using conventional IR methods. Next, the retrieved documents are re-ranked based on their dense vector representations in the continuous semantic space using advanced language representation models. In this work, we adopt a scientific transformer-based pre-trained model, such as SPECTER-2 [64], to generate dense vector representations of both the query and the documents. To perform this computationally intensive task, we utilize approximate nearest neighbors (ANN) algorithms, allowing for efficient and effective dense vector retrieval at scale. We also experiment with ElasticSearch's scalable vector search index endpoints in its library.

Through extensive experiments on various benchmark datasets, our proposed method demonstrates a significant boost in retrieval performance over the traditional ranking methods. Further analysis reveals that the re-ranking through dense vector retrieval effectively captures the intricate semantic relationships between queries and documents, addressing the limitations associated with sparse vector techniques.

2.6.1 Related Work

In recent years, the Information Retrieval (IR) community has witnessed a surge in the development and introduction of various neural ranking models, such as DRMM [65], KNRM [66, 67], and Duet [68, 69], which have significantly transformed the landscape

of large-scale digital libraries. These models deviate from the traditional learning-to-rank methods that rely on hand-crafted features by employing embedding-based representations of queries and documents, and directly modeling *local interactions* between their contents. This has led to the emergence of a recent approach that *fine-tunes* deep pre-trained language models (LMs) like ELMo [70] and BERT [71] for estimating relevance, which has shown remarkable success in bridging the pervasive vocabulary mismatch between documents and queries [72, 73].

In a short span of time, numerous ranking models based on BERT have achieved state-of-the-art results on various retrieval benchmarks [74–77] and have been adopted by leading search engines like Google and Bing for deployment. However, the significant gains in retrieval performance brought about by these LMs come at a steep increase in computational cost [77, 78], posing a challenging tradeoff between effectiveness and efficiency. For instance, BERT-based rankers have been observed to increase latency by up to tens of thousands of milliseconds even with a high-end GPU, which has an adverse impact on user experience and can potentially diminish revenue [79].

To address this challenge, recent research has started exploring the integration of Natural Language Understanding (NLU) techniques with traditional retrieval models like BM25 to strike a balance between precision and efficiency. Some notable examples include the work of Nogueira [80], who expand documents with NLU-generated queries before indexing with BM25 scores, and Dai & Callan [81], who replace BM25's term frequency with NLU-estimated term importance. Although these approaches have successfully reduced latency, they generally compromise on precision as compared to BERT.

2.6.2 Our Model and architecture for Re-ranking

As shown in the Figure 2.6, we first get top-k papers from the ElasticSearch text index which will be used later. We also have an embedding service running on a GPU, which takes in the query and provides the vector embedding for that query. The model that we are using here is SPECTER-2 by Allen AI, which was fine-tuned on scholarly papers and queries [64]. Once, we have the embedding of the query, we use the inbuilt Approximate Nearest Neighbour function from ES library to get the top 500 nearest neighbours.

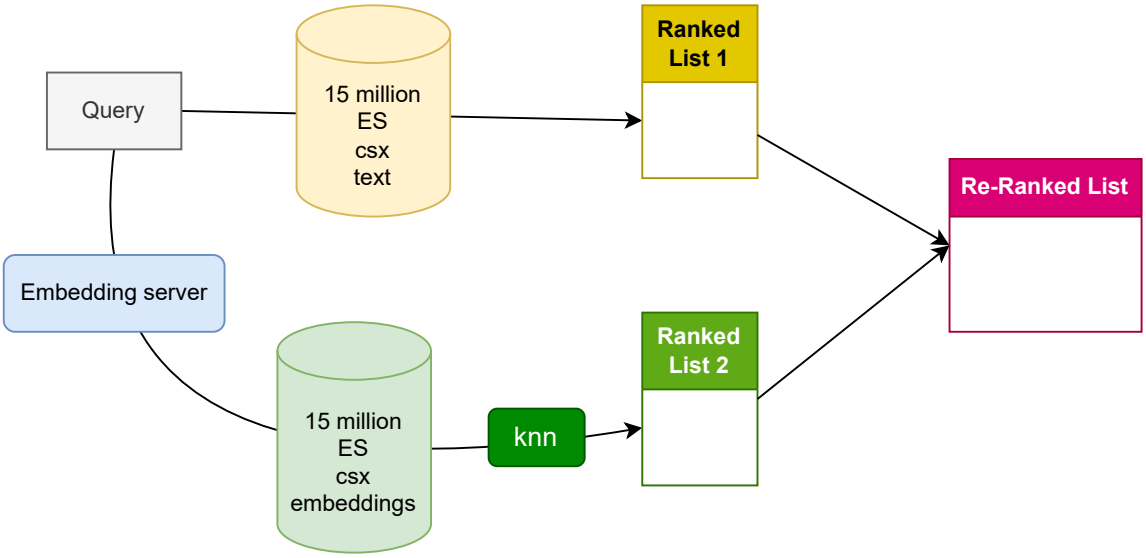


Figure 2.6: The high-level re-ranking pipeline. The most expensive step in this pipeline is getting the ranked List 1 from ElasticSearch(ES). Getting nearest vectors is fairly cheap and ES makes it scalable as well. The final step which is fusion of the ranked lists is $O(1)$

The way we are getting our embeddings is called a Bi-encoder[3] model as shown in Figure 2.7. In this model, the document embeddings are done offline and stored in a vector index. This reduces the cost of keeping a model in memory in a live system. Here we run the SPECTER2 model on a 8 GPU machine to get the embeddings for 15 million documents, which takes around 10 days to finish. Once we have this index and a query comes we just run an ANN search on the index to quickly retrieve the top-k documents in the index.

Reciprocal Ranked List Fusion

The Reciprocal Rank Fusion (RRF) is a method used to combine rankings from different sources or algorithms. It is particularly useful in information retrieval and meta-search engines. The RRF formula in LaTeX is:

$$\text{RRF}(d) = \sum_{i=1}^{n} \frac{1}{k + \text{rank}_i(d)} \tag{2.1}$$

Here's an explanation of the formula:

- $RRF(d)$: The Reciprocal Rank Fusion score for a document (or item) "d". - n: The number of different sources or algorithms being combined. - $rank_i(d)$: The

[3] `https://www.sbert.net/docs/pretrained_cross-encoders.html`

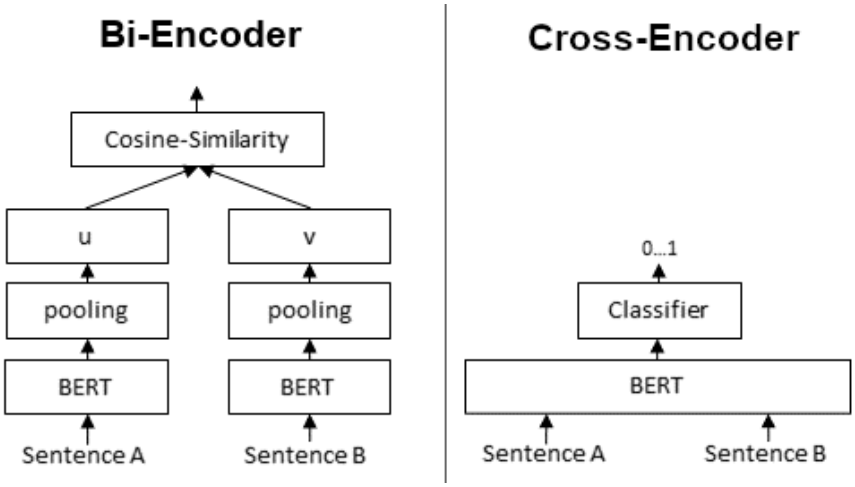


Figure 2.7: Bi-encoder vs. Cross encoder models for relevance scoring. Cross encoders are better at relevance ranking but are computationally much expensive to run while bi-encoders are not very precise but are cheaper to run.

rank of document "d" in the i-th source or algorithm. - k: A constant parameter that controls the weight of the rank positions. It is usually set to a small positive value (e.g., 60).

The formula computes the RRF score for a document by summing the reciprocal ranks of the document across all the sources or algorithms. The rank positions are offset by the constant parameter "k" to prevent the influence of very low ranks. The higher the RRF score, the better the combined ranking of the document.

2.7 Issues and Future Improvements

A major limitation of the current system is the absence of a management system overseeing all servers. For frontend management, Apache Kubernetes presents a promising solution for automating web application deployment, while Apache Airflow can improve backend extraction task management. In terms of web crawling, documents are currently batch-crawled from other digital repositories. To enhance freshness, we can develop an intelligent incremental crawler that predicts new document emergence from author homepages by analyzing crawl history [82]. Additionally, we can introduce asynchronous extraction and ingestion using message queues in Kafka to streamline the workflow and further increase throughput.

2.8 Conclusions

In this paper, we presented the design and implementation of a new digital library system as an improvement to the existing system. Aiming for greater accessibility, usability, scalability, and sustainability, we adopted valuable features from the current system while making significant changes to the architecture, hardware, and software. In the new architecture, Elasticsearch consolidates searching and datastore functionalities. We re-engineered the web application using Vue.js for frontend and Nuxt.js for server-side rendering (SSR), both providing a smaller learning curve, robust documentation, and code transparency to facilitate continuous development and convenient deployment. We also revamped the backend, integrating and parallelizing information extraction and ingestion modules, which significantly increased the ingestion speed to at least 1 million documents per day. Our Alpha framework is currently undergoing rigorous testing, with a prototype based on the ACL Anthology containing approximately 55k documents set for release before full deployment. The framework software will be made open-source once the system is fully deployed.

We anticipate that this new system will offer a more sustainable service, providing comprehensive, up-to-date, and semantically accessible scholarly big data. Additionally, we believe that this framework will contribute to the development of more scalable and customizable institutional digital library systems.

Chapter 3

Mathematical Information Retrieval in Scientific Text

There are three main research areas in MIR: **Search Interface** - Math cannot be limited to a text box to be queried, it requires a different input method like a canvas. This is only possible when the query is entered in the form of LaTeX, but this carries a condition that the documents should also be in tex format, which usually are not. The documents are mostly in MathML [1] or PDF format. **Formula Extraction and Recognition** deals with finding math in technical documents. This is a challenge, especially when the mathematical expressions are embedded within running text. **Indexing and Retrieval Techniques** explores how math should be indexed and how can this index can be queried on a scale. Math cannot be indexed like text [83] because of its inherent structure. Syntactically, there are many relations between variables and how they are positioned in an equation, which cannot be captured by linear text. Existing text indexing techniques are good at handling a linear sequence of character but not at spatial or hierarchical relationships [84]. This work focuses mainly on the indexing and retrieval techniques for math and how to improve them.

3.1 Related Work

We believe that building a MIR system which can be used by both experts and non-experts will reduce the elitism which is related to math. Systems built specifically

[1] `https://developer.mozilla.org/en-US/docs/Web/MathML`

for people with a particular set of skills create a divide in the society. The divide gets bigger if only the elite are given tools to enhance their skills further. This gives more power to the elite, which in turn gives them more control over the society. Karl Marx's work was the first one to identify technology as an unfair advantage to the ruling class [85]. It is important to have a free flow of information and knowledge between the experts and the non-experts as it helps the society grow as one unit. This gap encourages the anxiety related to math which is not good for common people. Experts can take advantage of this gap and then build more complex technologies finally making the common people pay for it. Building a open MIR system which enables people to search and understand how the high school math is being used to build a complex system with equations.

Looking at our system from an Actor Network Theory lens, we see math instructors as the actors in this Actor Network which pass on the skills to the non-experts and are constantly reducing the gap in the society. Math instructors have a knowledge sharing relationship with the non-experts which are the students in high schools or undergraduates. This MIR system is a non-human actor in this network. The students can be seen as *Intermediaries* in this network, meaning they do not make any difference in the network. In this case they will transport the knowledge given to them by the instructors without any transformation in the network. Math Instructors on the other hand are *mediators*, who understand other nodes in the network and are influencing the students with the skillset they possess. When we look at the interactions of the system with students and instructors this way we see that there is no power imbalance in the network, everything is flat and is interacting with each other to make transformations in the network. The mathematical knowledge is the **quasi-object** in this network, which will be transferred between the nodes. Math experts use their own terminology to define a publish scientific contribution. Only few of these very technical articles are apprehended by the non-experts. Our search engine will make these concepts searchable and easier to explore enabling non-experts to learn more about the concepts. The parts that the humans and technology play in the Actor-Network relationship show that the MIR system uses a post-humanist approach which is what we need in our current society.

Integrity is something that is truly lacking in our current society as the improvement of technology has led to avenues for plagiarism. People can search on online for anything, copy and paste it and not cite. From the artist who steal other lesser

known musicians' songs or beats to researchers who quote and do not cite another researchers' work. For mathematics field, checking plagiarism is even harder because it is hard to extract math expressions from articles. Therefore, the MIR system allows Math Instructors to be able to catch plagiarized work more easily due to the structure of the MIR system. Mathematical Instructors can use the search techniques built by our MIR system to search for similar work and detect plagiarism. Tools like *Turnitin* help instructors check for plagiarism in text. This tool at its core is nothing but a search engine which checks for phrases and paragraphs in the submitted text with the existing published work. Our system if tuned correctly for such a task can detect such unoriginal work which will help punish such acts and keep the quality of published work pristine. As the scientists who are publishing have more tools to explore around, they can copy work which has not gained traction or is not that commonly known. [86] mentions this as a "technical solution to a technical problem". This gives an advantage to the mathematical instructors and levels the field.

Math is a very special science as most of the other sciences justify their observations and findings while mathematics is one of the few ones which prove everything concretely. Experts cannot invalidate the concrete proves, if done correctly, to derive mathematical theories [87]. [88] report that the students and instructors use a search engine as their first point of reference when stuck with a problem. They preferred Google search over any other reference source like digital libraries. Given the plethora of information out there on the web, search engines can filter out the relevant information and understand the information need of the user. A MIR system can help the instructors in many ways. They can find references for a equation in other published work. These references can be recommended to the students to refer and get inspired from. The instructors can show actual use of math formulae in research to the students by just searching a familiar formula and then showcasing the derivative work. This will make the interaction more interesting and will even inspire students to carefully go through an important topic.

3.1.1 Collaboration with Mathematicians as Domain Experts

Mathematicians collaborating to solve complex problems has been studied extensively by [89, 90]. The authors created an online platform called **Polymath**, a massively collaborative online mathematical project, a place where domain experts and non-

experts like high school students can collaborate to solve and examine unsolved or complex math problems. Scientific discovery by collaboration is hard, as scientific knowledge creation is complex, it requires a very specific set of skills and has to be coordinated extensively. There are a very few examples of systems where highly skilled experts collaborate to contribute to a scientific discovery. Polymath on the other hand is a unique system which facilitates scientific collaboration. Math itself is a derivative subject where proofs and theorems proved previously by past mathematicians are used as preliminary work to build new theorems.

[89] highlight that new users of the platform need to be socialized, so that they can get familiarized with it and get up to speed with others. The platform was also great at incentivizing the members who engaged with it regularly or made significant contribution to the platform in the form of a solution or a blog post explaining something complex. Till now 16 projects have been proposed and launched under Polymath since 2009 [2]. Most of the work has concluded and the research results have been published by the contributors. This makes it an excellent example for collaborative systems and cooperative work. The study also focuses on design recommendations like filtering out important content as every post on the blog is linear with time, there is no way to rank the useful posts. Other suggestion was to highlight pending tasks for new-comers. Highlighting these tasks can help them identify where they can focus and start working. A paper in Nature [91] discusses why Polymath is so successful. One of the main reasons mentioned was the open nature of the platform. Instead of a hierarchy of mathematicians, every collaborator is open to explore whatever they want to do. This allowed free flow of ideas and perspectives. This also allowed unintended discoveries and connections to be made. These connections made complex scientific discoveries possible as math is all about solving smaller problems by making connections to the past proved theorems.

There is a need for a MIR system, but it should be designed in a way that it is intuitive and not frustrating for the user. Several contributions have been made in this area as well. [92] built a **Freehand Formula Entry System** which would convert the handwritten input via mouse to LaTeXstrings. The system even allowed users to individually correct the symbols in the math expression. This had not been done earlier by any system. While interacting with the system the users were not expected to know any special markup language. The interaction between the user and

[2] `http://michaelnielsen.org/polymath1/index.php?title=Main_Page`

the system was more natural and familiar. This was a step-up from template-based systems **Xpress** [93] where users were expected to type the symbol on a canvas instead of drawing on it. It presented a text editor based solution to mathematical input. But both these systems assumed a lot of things about the users and no preliminary feedback was taken before building these systems.

Consequently, a requirement study for MIR systems was done by [94] which provided useful insights to what the users want. The objectives of the study were - 1) what users want in a math search engines, with a focus on design and 2) are the current systems adequate. The study found that the users are comfortable keyword search instead of inputting equations, as it was not a easy task. There should be expression to keyword links (e.g. $a^2 + b^2 = c^2$ linked to Pythagorean Theorem) so that users are able to search for both. Expert users find the systems discussed in the previous section really useful(MathDex) but for the masses, keyword search is enough.

There are studies of the existing MIR systems through the lens of a Physicist [95]. This study compares all the systems and reviews their relevancy to physics. As physics is heavily relies on math, when a mathematical roadblock is hit in such a research, math search can be very useful in exploring other domains to find the solution. The study reinforces the need of a MIR system for cross domain concept examination.

Certain studies have been conducted to understand the information need for the MIR systems. Specifically, [96] try to understand the search behaviour of professional mathematicians by conducting interviews. The aim was to identify the characteristics which make the experts different from non-experts in math. After identifying these key characteristics, the authors made design recommendations for a MIR system. One of the interesting findings was appreciation of social interaction by the math experts. These interactions included **collaboration** and feedback exchanges.

3.1.2 Math Formula Notation

The need for a MIR systems emerged as the scientific documents contained mathematical expressions and formulas which are presented in different ways and needed different retrieval methods. Mathematical queries can contain both text and formula. In general search engines such as Google the user may include a formula in the query using LaTeX or MathML tags, which are treated as a word. In MIR systems, however,

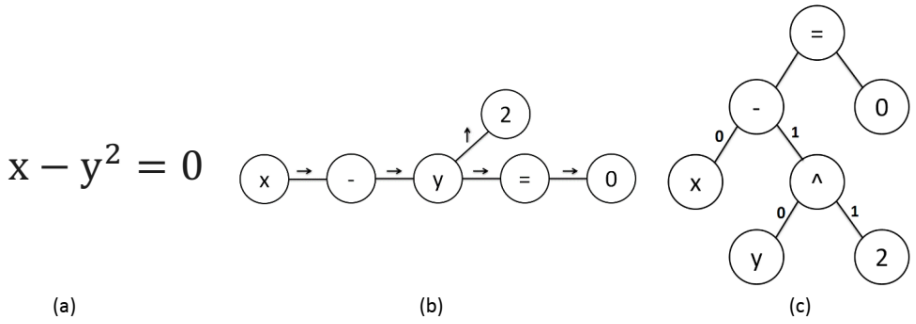


Figure 3.1: Formula (a) $x - y^2 = 0$ with associated (b) Symbol Layout Tree (SLT), and (c) Operator Tree (OPT).

for formula input, search engines usually provide a user interface. For formula retrieval previous works mostly considered text-based or tree-based models [97]. Math encodings are naturally hierarchical, defining formula appearance in symbol layout tree (SLT) encodings such as LaTeX or formula semantics in operator tree (OPT) encodings such as Content MathML. SLTs represent the placement of symbols on baselines along with their spatial arrangement [98]. Although SLT representations are mostly used online, they can be ambiguous. For instance, a symbol can be a variable in one context, and a operator in another. In contrast, OPTs are mathematically unambiguous. An OPT represents the operator and relation syntax for an expression. Figure 3.1 shows the SLT(b) and OPT(c) representation of formula $x - y^2 = 0$.

Nodes in SLT and OPT represent individual symbols and explicit aggregates such as function arguments in shape of (type!value). More precisely, the nodes can be numbers (**N!**n), identifiers such as variable names (**V!**v), text fragments, such as 'lim' and 'such that' (**T!**t), fractions (**F!**), radicals (**R!**), explicitly specified whitespace (**W!**) and finally, matrices, tabular structures, and parenthesized expressions (**M!**f rxc); with f showing fence characters such as parenthesis, r number of rows and c number of columns. In SLTs, operators do not have a type attribute but in OPTs, commutative and non-commutative operators have type (**U!**) and (**O!**) respectively. Providing examples, variables are shown with (V!v) where v is the name of the variable.

Considering an object O, there seven type of labels for edges, showing the spatial relationships between the nodes: **next** ('n') that references the adjacent object appearing to the right of O and on the same line, **within** ('w') references the radicand if O is a root or the first element appearing in row-major order in O if it is a structure

represented by M!, **element** ('e') references the next element appearing after O in row-major order inside a structure represented by M!, **above** ('a') references the leftmost object on a higher line starting at the position above O, **below** ('b') references the leftmost object on a lower line starting at the position below O, **pre-above** ('SUP') references the leftmost object of a prescripted superscript of O and **pre-below** ('SUB') references the leftmost object of a prescripted subscript of O.

3.1.3 Recognition of Mathematical Expressions

While searching is one of the challenges this research aims to address, mathematical expression recognition is what makes it's contributions even more important. Most of the existing systems and research only cares about getting as many relevant documents as possible. This might work for some users who know how to use these systems given the knowledge of LaTeX. Also, entering a math expression in a linear text box does not work for fractions, integrals or large summations (for e.g. $\binom{N}{k}$ cannot be written as $(N\ k)$). Hence, it is very important to build a novel query interface for MIR which accommodates the written nature of math. It should be intuitive and easy to use for the users as well.

[99] developed *min* a web interface for math search with handwritting detection. A user could *draw* math expressions and the recognition system would convert it into LaTeXand then search on different sources such as - WolframAlpha and Wikipedia. It is a step in the right direction. The system was given to non-expert users to try out. The feedback given by the users was that *min* is intuitive and simple to use. Though the system was a prototype and the performance were very slow and the recognition accuracy was not acceptable, it proved a point that there is a need for MIR systems with such capabilities.

Several efforts have been made to improve the recognition rate of mathematical expressions. [100, 101]. [102] use a continuous Hidden Markov Model which encodes each symbol class left to right. This system achieves a top-1 recognition rate of 82.9%. [103] present an extension of this work which was placed second in the CROHME 2011 handwritten math recognition task. But these techniques have been beaten by the recent state-of-the-art deep learning techniques such as auto-encoders [104, 105].

3.1.4 Math Formula Embeddings for Indexing and Retrieval

There has been a tremendous amount of research on continuous representations for words and paragraphs in Information Retrieval (IR) and Natural Language Processing (NLP), little work has been done on formula embeddings. This is unfortunate, as mathematics has a central role in scientific discourse. Vector representations for formulas have the potential to be applied in a variety of ways. For instance, [106] proposed a query auto-completion method for rare word prefixes using a convolutional latent semantic model. Many formulas are unique, and such a technique might be used for query auto-completion in math search. Formula vectors can also be used for formula retrieval: given a query, related formulas are retrieved and ranked based on their vector similarity to the query formula vector. As with words, mathematical formulas can express similar content in different ways, thus making distributed representations that offer a level of abstraction potentially useful. These advancements are so significant that there is a new field which has emerged called Neural Information Retrieval [73, 107]. This new research area includes sophisticated techniques such as word2vec for word representations and auto-encoders for related document search. While many information retrieval and natural language processing tasks benefit from distributed (i.e., dense vector) representations of words [108–110], embedding models to produce vector representations for mathematical formulas have not yet been as fully investigated.

There are various goals which can be achieved if we could represent a formula as a dense vector. Word embeddings have been use to expand queries [111]. A similar thing can be done for math retrieval, suggesting users with semantically similar math symbols can be very useful for tasks such as mathematical query auto-completion. Document ranking has been improved by using the distributed representations of the words in the document corpus. text [68, 110, 112]. [112] use the dense as well as the sparse representation of the words to rank documents. They "map the query words into the input space and the document words into the output space and compute a relevance score by aggregating the cosine similarities across all the query-document word pairs".

3.1.5 Math-Aware Query Autocompletion (QAC)

Query auto completion techniques have been reviewed in several papers [113–115]. Here, we provide a brief summarization of the types of methods proposed and their features.

QAC problems can be viewed as a form of ranking problem, given a query and a prefix-tree data structure. There are three categories of promising solutions making use of popularity, time, or similarity [114]. Popularity-based methods use the frequency of query candidates past popularity, as measured using document frequency (i.e., occurrences in sentences) or frequency in query logs. The term occurrence ranker (TO) uses TF-IDF combined with term popularity [113].

Time-based approaches are based on session information. In time ranker (TR), the scores depend on time elapsed from the most recent occurrences in query logs. The most-popular time ranker (MT) combines most popular (MP) ranker and TR in form of a convex combination [116].

Similarity-based methods weight query candidates by their similarity to user query logs or the documents previously clicked. Similarities can be measured as words, phrases, or context, e.g. n-gram similarity, semantic similarity, etc.

[117] constrain search results in a given category using entity names input by the user. For example, after the user picked "Donald Trump", an input prefix like "Sim" should not exactly prioritize famous singers like Paul Simon, who are not politicians. Although they appear important, they are unrelated to "Trump".

QAC can also be categorized into two broad categories – heuristic models and learning based models, depending on whether machine learning methods are applied or not [114]. The heuristic category includes classical QAC methods, which can be further divided into time-sensitive (e.g., most popular completion variances) and user-centered (e.g. personalization using session context [117] For example, user's actions such as skipped query completion and eye contact can also be used as implicit feedback. Learning based models adopt many features to classify and rank candidates. [118] investigated the correlation between queries whose popularity behaves similarly over time. They could then find the highest correlated queries above a certain threshold to an input query. Other learning features are related to patterns in access logs [119], entity names in queries [120], and demographics [121].

[119] learns a model from query logs of Yandex, a popular Russian search engine,

it defines a query-term graph to model likelihood of a sequence of query prefix. The graph they build is based on the steps that whether user examines the query suggested on the jth position or skips.

As stated earlier, all this has been done for textual queries and not for math formula retrieval. The first logical step to follow from the previous work is to build and evaluate a basic system using something preliminary like prefix-matching or pattern-matching to establish a baseline. The work can be then improved to get better results by using more sophisticated methods discussed above.

3.2 Problem Statement and Methodology

The main goal of this work is to build state-of-the art techniques for indexing and retrieval of math formulae. We propose two novel embedding based methods which will capture the semantics in the math formulae. We also propose preliminary work for math formula query autocompletion, which will be used to suggest math symbols for math query completion. All these techniques will be combined in a MIR system to make math in open academic articles in CiteSeerX and Wikipedia articles searchable. There are many components in building a MIR system. These include - **recognition and extraction** of mathematical expressions from documents, **indexing and retrieval**, and a new interface which includes **QAC** to suggest relevant symbol suggestions as the user enters them. This work makes contributions to the indexing and retrieval components of a MIR system. Specifically, this work makes efforts in capturing the semantics of a math formula. A lot of work has been done on indexing using tree structures of the formulae but a few look at the semantics. In the upcoming sections we discuss the approaches for capturing semantics and QAC for math.

In the following sections 3.2.1 and 3.2.2 we present two novel approaches for representing math formulae as dense vector representations. After that we discuss Math

3.2.1 Structure-based Formula Embeddings

Mathematical notation also offers the potential for representing some types of semantic relationships in ways that words alone would lack, and these semantic relationships

offer additional scope for the construction of embedding models. Formula representations are inherently hierarchical, either in the form of symbol layout trees (SLTs) that capture the placement and nesting of symbols on writing lines (e.g. in LaTeX), or as operator trees (OPTs), which more directly capture the mathematical semantics of the application of operators to operands (e.g., in Content MathML). (see Figure 3.1)

General embedding models such as DeepWalk [122] have been developed for graphs, including trees. These models are typically constructed by first traversing the graph in some way (e.g., breadth-first, depth-first, or random walk) to linearize the graph and then building an embedding model in the usual way. Current graph embedding models generally treat nodes as atomic units, but mathematical formulas exhibit more fine-grained structure (e.g., some symbols are variables whereas others are constants) that could be leveraged to produce improved embedding models. To embed mathematical formulas, we therefore propose first performing fine-grained formula analysis to produce a tree representation in which the nodes have an internal linear structure, then linearizing the tree representation to produce a total order on the nodes, then tokenizing the resulting linear structure to generate n-grams of sub-node features, and finally training an embedding model on those n-gram features. Using n-gram helps as mathematical formulas are unique or nearly unique mostly occurring only once in the collection. Our goal is to better generalize formulas by producing more robust vector representations for unseen and unique formulas.

fastText Model Applying n-gram embeddings is beneficial for our proposed model as mathematical formulas are often unique. For task such as query suggestion, users may insert unseen or even wrong formulas as input query which makes model such as word2vec less appropriate as they cannot handle unseen formulas. FastText is a n-gram embedding model [123] derived from the word2vec model by [124]. In word2vec, individual words are the smallest unit for vector representations, and internal word structure is ignored. FastText generalizes word2vec to accommodate word morphology, providing usable vector representations for rare and unseen words. As illustrated by [123], given a dictionary of n-grams with size G, for a word w that has set of n-grams ζ_w, a vector representation of word w is the sum of the vector representations of its n-grams and a scoring function which maps pairs of (word, context) will be calculated as:

$$s(w, c) = \sum_{g \in \zeta_w} z_g^\intercal v_c.$$

where, z_g is vector representation for the n-gram g and v_c is the vector representation of the word appearing in the context window of word w. FastText represents words as bags of n-grams, with n ranging from 1 to the length of the word. For instance, considering n-grams, $n = [3, 6]$, the word '*system*' (with no boundary symbols) is presented as *sys*, *yst*, *ste*, *tem*, *syst*, *yste*, *stem*, *syste*, *ystem* and *system*. In this model, the minimum and maximum number of n-grams for training is a hyper-parameter. After generating vectors for n-grams, the vector representation for a word is the sum of its embedded n-gram vectors.

Math Formula Embedding Unlike development of word embeddings methods, there have been few attempts for formula embedding. Early research on formula embedding was carried out by [125]. They developed a variant of the doc2vec algorithm, the distributed bag of words (PV-DBOW). [126] introduced expression trees and assigned each formula to a vector such that formulas with similar structure are close to each other in the vector space. [127] created embeddings for both symbol (symbol2vec) and formula (formula2vec). Symbol2vec was based on a Continuous Bags-of-Words (CBOW) architecture using negative sampling [128], while formula2vec uses a distributed memory model of paragraph vectors [126]. The proposed method by [129] generates embeddings for both words and equations with a larger context window size for equations than words. They also propose equation unit embedding, treating equations as sentences where the words are symbols, variables and operators, referred to as a unit. Despite these attempts, to best of our knowledge, previous research has not focused on isolated formula embedding. Isolated formula embedding focuses only on structure of formulas along with similarity of their variables and operators. In this way one can provide two embeddings, one based on similarity formulas and then another one for similarity of their surrounding text. Also, another shortcoming of these approaches is that embeddings are defined at the formula (word) level and therefore there is no embedding for unseen formulas or symbols. In math search, users may submit formulas in queries that do not exist in a collection. Based on this property, a character-level approach for formula embeddings may produce more robust vector representation.

3.2.2 Image-based Formula Embeddings

The way we see formulae is similar to when we look at images. We glance it left to right skipping all the complex SLT or OPT relationships that are there in the expression. The hypothesis here is that formula retrieval is closely related to image retrieval. If the formula are not closer to text and if really are closer to images, then the state-of-the-art algorithms for images should work on math formula images as well. **Image-Based Formula Representation** A lot of large scale systems have been developed which can search on images efficiently as surveyed by [130]. This work identifies three key issues in image retrieval - representation, organization and similarity measurement. Traditionally hand-crafted features were used to extract important features from an image to build a representation. These include color, shape, texture and structure. Gist [131] and Scale Invariant Feature Transform (SIFT) [132] have been used extensively to extract the features for images for years. These techniques have been improved slightly over the years until Convolutional Neural Networks (CNN) [133] were introduced. CNN's were found to beat all of the state-of-the art methods for Image Classification (ImageNet) task. CNN can extract high level features like edges and learn a dense representation of an image. A number of studies have shown that CNN's can be used for image retrieval and give the best results [134–137]. [138] use network weights trained for image classification like Alex-Net [139] and VGG-Net [133] to get the representations of the images. They exploit the last layer of the neural network as representations for image retrieval. Other works also follow similar strategies while tuning the weights of the networks for their data.

Image retrieval for MIR has been studies but not that comprehensively as compares to image standard retrieval. [140] employ a simple Content-Based Image Retrieval (CBIR) approach. This is the first effort made in the direction of using image retrieval techniques for math information retrieval. They show that image-based representations show some improvement over classical structure and LaTeX based matching. Another similar work by [141] uses Shape Contexts to represent the math symbols and instead of using k-means algorithm to cluster it uses Self Organizing Maps [142]. It is to be noted that this technique was designed only for retrieval of isolated math symbols and not complete mathematical expressions. [143] presents an application of CBIR for searching formula in Video Lectures.

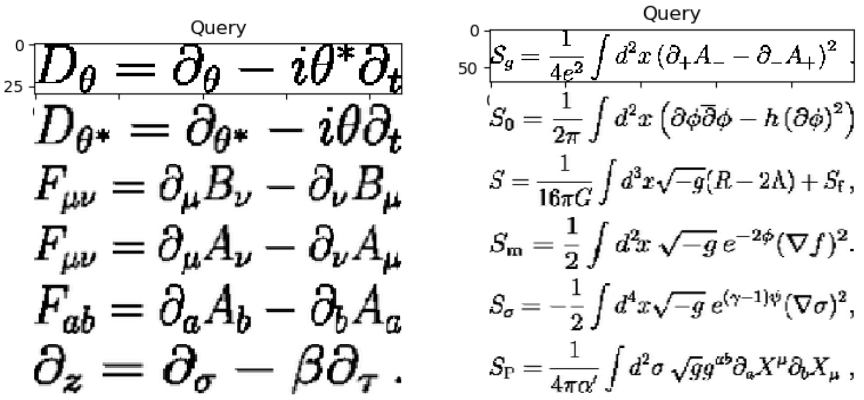


Figure 3.2: Initial Exploration of CNNs to represent Math Formula as images. The first formula in the box is the query whose five nearest neighbors have been listed below. a) The first returned formula is slightly different but still comes up as the first neighbor. Structural matching is not robust enough for such cases as it breaks with small changes in the formula b) This image demonstrates technique even works for larger formulae. The complexity of querying remains the same even if the formula is more complex unlike structure-based tree matching.

Math image retrieval has been studied and has shown to give the best results because of the abstract nature of the images. But it is difficult to scale, and the image representations are not at par. This is where CNNs can play a vital role.

Preliminary Experiments Experiments were conducted as a part of this work to see if the pretrained CNNs like VGG-Net can in fact capture the semantics in a math equation. The aim of this experiment was to firstly represent the math formula images as vectors and secondly exploring the nearest neighbours for those vectors in the vector space. We chose a math image dataset im2latex-100k [3]. This dataset has 103,536 images of formulae, of the same size [2339, 1654, 3] (row pixels, column pixels, Number of Channels - RGB). All these formulae have a unique LaTeX notation. All images were trimmed down to have only the formula in frame, to increase detail in the image. Then every image is resized to (224,224,3) to reduce computation complexity. Once we have the images in the standard size, we propagate them through the pretrained CNNs and finally extract the last layer of the network and call it a dense representation of the math expression image.

Model - Every formula image is fed into a VGG19 [4] pre-trained model(ImageNet

[3]http://lstm.seas.harvard.edu/latex/

[4]https://www.kaggle.com/keras/vgg19/home

dataset) to obtain feature vectors of a formula. Each vector is of the size 100,352. Once we have the vectors for each formula image, we can get the K-Nearest Neighbors of these formula images. The experiments were carried out on a machine with - 64GB of RAM, and a GTX 1080ti. As you can clearly see in figure 3.2 that the CNN's return very similar formulae even in a dataset with not so similar equations.

3.2.3 Math Query Autocompletion

Math Query autocompletion is a new research area and to the best of our knowledge, there hasn't been any previous work published in this domain. WolframAlpha and Symbolab are live systems which support math QAC, but the systems are closed, and from what we can observe the systems use prefix matching for candidate retrieval. Thus, math expressions re-ordered around commutative operators (e.g., $a+b=b+a$) or the ones using a different set of symbols than the query will not show up as candidates. Subexpression matching is also missing, as math is hard to tokenize. Formula auto-completion may have to deal with unseen subexpression completion (similar to the unseen prefix issue in text QAC). One possible strategy to this problem is to match expressions with more tolerance in structure and combine semantic embedding similarity to broaden the boundary of only suggesting formula queries to also suggesting text queries. The deficiency of math search query logs is another issue. In this work we use corpus data instead of query logs for query candidate retrieval. We also use prefixes to obtain candidates for query completions as opposed to the largest common substring between a query formula and each indexed expression. [144]. We also try other pattern matching strategies discussed in the following sections.

QAC Strategies

The different strategies to autocomplete a query have been discussed in the paper by [115]: exact match, prefix match, pattern match, and relaxed pattern match. Math in LaTeX form cannot be tokenized on spaces, so we use the first three approaches only as the last one requires tokenization on spaces.

The function $Prefix(S)$ can be defined as -

$$Prefix(S) = \{S[1:i] \mid i \in [1, |S|]\}$$

where $|S|$ is the length of the string, and $S[1:i]$ are the first i letters of S.

We use three strategies to auto complete formulas in LaTeX strings, listed below.

Here P represents the query prefix provided by the user, while T is the set of auto completion candidates.

1. **Exact Match (EM)** : This is the most basic matching strategy, and only returns True if the string in T is exactly present in the candidate set.

$$ExactMatch(P) = \{T_i \mid T_i \in T \wedge P = T_i\}$$

2. **Prefix Match (PRM)** : Prefix matching is one of the most common approaches for matching the query string P to the candidate set T. In this the prefixes of the query are matched with the document collection, which are considered as query logs. The queries which match the prefix are a part of the candidate set [145].

$$PrefixMatch(P) = \{T_i \mid T_i \in T \wedge P \in Prefix(T_i)\}$$

3. **Pattern Match (PAM)** : This mode performs a standard substring match over each token P_i of current query P. A substring search is carried out over the query log string T_i.

$$PatternMatch = \{T_i \mid T_i \in T \wedge (\wedge_{P*k \in PMatch(T_i, P^k))}\}$$

   Here k is the number of tokens from the query for which the candidates have to be retrieved. $Match(T_i, P^k)$ is an auxiliary function which returns $True$ if P^k is a substring of T_i and $False$ otherwise.

3.2.4 Corpora

We will include two data sources which are available openly and used extensively - Wikipedia and CiteSeerX. **Wikipedia** Wikipedia data has been provided by [146] as a part of National Institute of Informatics Testbeds and Community for Information Access Research (NTCIR)[5] MathIR task. Here the authors provide 319,689 Wikipedia articles, out of which 10% have <math> tags and the remaining others are distractors without <math> tags. The math documents have more than 590,000 formulae in them.

[5] `http://research.nii.ac.jp/ntcir/ntcir-14/`

These formulae are encoded in LaTeX, Presentation MathML and Content MathML. Each formula carries a unique Wikipedia page id followed by its unique id. For example, on the Wikipedia page Dual polyhedron - `https://en.wikipedia.org/wiki/Dual_polyhedron`, the first equation (x_0, y_0, z_0) is given the id *Dual_polyhedron:1* and so on. [147] provide tools to extract the SLT and OPT tuples explained in section 3.1.2.

CiteSeerX [148] [149] predict that at least 20% of the 10 million documents are about Physics and Computer Science. We expect to extract a lot of math formulas from CiteSeerX's academic document store. [150] present work which can extract math formulae from PDF documents by building bounding boxes around the math symbols. [151] build a tool called Maxtract, which can convert PDF's to LaTeX. Once we have the representations of math in LaTeXwe can proceed with the steps mentioned above to index the formulae on a scale.

Conversion to images The LaTeXstring obtained from the Wikipedia can be converted to PDFs using pdflatex [6]. Once we have obtained the pdf's of just the formulae, we convert them into images using pdftoppm[7] tool by Glyph & Cog, LLC.

3.3 Evaluation

3.3.1 Document Relevance Labelling

zanibbi2016ntcir provide an overview of the NTCIR12 MathIR task. As a part of this task, the organizers had released ground truth labels and a set of math formula queries from different topics. These queries also contain wildcard[8] queries, in which the symbol $*$ is placed instead of the math symbols. Each query is chosen from one topic. The task is further divided into 2 parts - arXiv Topics and Wikipedia Topics. arXiv topics are sophisticated and complex as these are more relevant to the Math experts who go through the published work specifically in the field of math. These topics are a part of the 105,120 scientific articles from arXiv. This collection of arXiv articles yields a total of 8,301,578 searchable units. On the other hand, Wikipedia topics have been chosen for the non-expert population, like graduate and undergraduate students. For our initial exploration and experiments we use all the 30 queries derived from the topics just from Wikipedia. We chose this because 500,000

[6] `https://www.tug.org/texlive/`

[7] `https://linux.die.net/man/1/pdftoppm`

[8] `https://nlp.stanford.edu/IR-book/html/htmledition/wildcard-queries-1.html`

formulae is a small number as compared to 8,000,000 and once we get our approaches working on this small corpus we can expand to arXiv as well.

Each query has a unique identifier - **MathWiki-1, MathWiki-2** and so on. Each query is expressed in three notations - Presentation MathML, Content MathML and LaTeXusing custom XML format [152]. These queries have been labelled by evaluators with varying mathematics background. For the arXiv task graduate student of mathematics were chosen. For the Wikipedia task which required less sophistication, evaluators were undergraduates and master's students. The evaluators judged the relevance of the filtered documents to the query. Every query was marked with a relevance score of 1.0, 2.0, 3.0 and 4.0 with 4.0 being the most relevant and 1.0 the least relevant document. The relevant documents for every query range from 40 to 100 in number. The scores from all the evaluators are pooled so that there is no discrepancy in the ground truth.

3.3.2 Metrics

Precision at k (P@k) As we know for a modern large scale search engine recall is not that important as it represents the score for all the relevant documents retrieved. Only a few people are interested in reading all the results. Instead looking at only top-k retrieved results and see how many of them are relevant. This has a shortcoming - we do not look at the order of the returned results but only the number of returned results. **Mean Average Precision (mAP)** mAP is a global metric which mean of the average precision of all the queries tried on an IR system. It still carries the deficiencies of *Precision at k* as a metric. **Binary preference-based measure (bpref)** [153] *bpref* is a very important measure in this case, as some judgments are incomplete. It is designed for such situations. It is mostly based on the relative ranks of just the judged documents. It penalizes the irrelevant documents which occur before the relevant documents.

$$bpref = \frac{1}{R}\sum_{r}\left(1 - \frac{|n\,ranked\,higher\,than\,r|}{\min\left(R, N\right)}\right)$$

Normalized Discounted cumulative gain (nDCG) [154] The metric penalizes the score if highly relevant documents appear lower in rank. The judged position of the document is reduced using a logarithmic operation. An ideal discounted cumulative gain is used to divide this score to get the NDCG score. Once we have the

NDCG score for all the queries we can average them out to get a global NDCG score.

$$\mathrm{DCG_p} = \sum_{i=1}^{p} \frac{rel_i}{\log_2(i+1)}$$

3.3.3 Task Set

There are mainly two datasets available for evaluation a math-aware search system. They both have relevance labelling for the query-document pair. We will discuss in detail what each dataset has to offer.

NTCIR-12 Math-IR Dataset [146] NTCIR-12 MathIR Task is a task dedicated to information access for mathematical content. The MathIR task makes use of two open corpora - arXiv and Wikipedia. The first corpus contains academic articles in the arXiv, while the second corpus Wikipedia articles in English. For each corpus, there were two tasks. Three subtasks contain queries with keywords and formulae (arXiv-main, Wikimain, and arXiv-simto), while the fourth considers isolated formula queries (Wiki-formula). The arXiv corpus has 105,120 scientific articles in English. The dataset contains articles from the arXiv categories math, computer science and physics to get a varied sample of technical documents containing mathematics. Each document is divided into paragraphs - which is a unit of retrieval for this dataset. This produces 8,301,578 documents with approximately 60 million math formulas. The second task is on Wikipedia dataset. The MathIR Wikipedia corpus contains 319,689 articles from English Wikipedia. There are over 590,000 formulae in the corpus.

Topics for these tasks have been chosen from a wide variety of areas and complexity based on the formula. There are a total of 29 topics in the arXiv-Main task and 30 topics in Wiki-Main task; these topics consist of keywords and formula in them. The relevance hits for each topic-document pair were labelled by participant from a pool of submissions. The relevance scores range from 1 to 4 1 being the least and 4 being the most relevant.

ARQMath lab at CLEF 2020 [155] Math Stack Exchange (MSE) dataset was used in the question answering task proposed by the authors. The task here is to find answers to new mathematical questions from 2019 among posted answers on a community question answering site (Math Stack Exchange). Queries are question postings held out from the test collection, each containing both text and at least one

formula. While several models have been proposed for text question answering, math question answering is in an earlier stage of development. To advance math-aware search and mathematical question answering systems, the authors have created a standard test collection for researchers to use for benchmarking. The total number of topics in this dataset are 77, which is much higher than the previous datasets. Also, the MSE dataset has use of informal language, which makes it different from the arXiv dataset which is sceintific documents only. A better understanding of informal language around math will push the boudaries of math-aware research and help make it accessible to the masses.

3.3.4 Current State-of-the-art MIR Systems

One of the first large scale math-aware search engines was developed for the NIST Digital Library of Mathematical Functions (DLMF) [9] by lozier2003nist. The documents included matrices, graphs, mathematical formulae, and methods of computation. DLMF enabled searching math via text queries in LaTeX notation. Here traditional Information Retrieval (IR) techniques of tokenization of formula as text along with keywords, inverted indexing and retrieval were used. The work used Apache Lucene [10], which is a very efficient implementation of Lucene [11] for text-based retrieval. Since then a lot of new MIR system have emerged, both open source and commercial. A similar technique has been used by various other math-aware search engines [156, 157].

Further improvements to this technique of searching and retrieval were discussed in the work by miller2003technical. This work highlights the short-comings of the previous work implemented in DLMF. These include some of the major math search issues and why conventional search methods are inadequate for math search - non-alphabetical symbols, rich structure of math (e.g. $sin(x + logx)$ is different from $sinx + logx$) and multiple representations for the same concept (irrelevant variations; discussed further). These issues motivated the need of indexing mathematical structures. New techniques such as Textualizing and Flattening of the math formulae were introduced to make the existing IR systems work better

These systems should be seen as prototypes for a MIR system. kohlhase2006search mentioned semantics in mathematics for the first time with respect to IR. Math

[9] `https://dlmf.nist.gov/`
[10] `https://lucene.apache.org/solr/`
[11] `http://lucene.apache.org/`

search is not a trivial problem. Search for math expressions is context dependent (e.g. $\binom{n}{x}$, C_k^n are the same), has identical representations for distinct math objects (e.g. $\int f(x)dx$ can mean Riemann Integral or a Lebesgue Integral) and should should consider irrelevant variations (e.g. $\int f(x)dx$ is same as $\int f(y)dy$). The outcome of this work was MathWebSearch. They used techniques such as tree indexing which was one of the first application of it in math retrieval. These techniques would facilitate the existing IR techniques to work for querying and indexing for mathematical expressions. But these techniques again had issues stemming from the multiplicity of linear forms that an expression can be mapped to. It should be noted that all these techniques were design improvement suggestions and only some of were implemented in live-systems.

Existing Systems : A recent survey by guidi2016survey mentions open source Mathematical Information Retrieval systems - Approach0 [12], Design Science's MathDex [13], MathWebSearch [14] and MiAS [15] (Math Indexer and Searcher). After that some new commercial systems have also come up in the recent months - SearchOnMath[16], Springer Latex Search[17], Formulasearchengine[18], Symbolab[19] and WolframAlpha [20] Almost all of these systems do not support semantic matching of expressions. Further, none of the above systems allow users to input their queries in a handwritten manner. There have been efforts in this space as well which will be discussed in the Section 3.1.3. As we have seen above there is a clear need of a MIR system designed with semantics in mind which is where our work comes. The key problem with using text-based approaches is that the text can hold limited information about the structure of an expression. Although this might be easier to implement and it works well in practice, textual matching can never align well with a good structure-based expression matching algorithm [1].

There are open source state-of-the art systems as well [158–160]. We lay down their performances for the NTCIR12 Wikipedia MathIR task in the Table 3.1. zhong2019structural built a system called Tangent-S, which uses the OPT based representation of the formula to capture the semantics, in terms of the operators to

[12] `https://approach0.xyz/`
[13] `http://www.mathdex.org/`
[14] `http://search.mathweb.org/`
[15] `https://mir.fi.muni.cz/mias`
[16] `http://www.searchonmath.com/`
[17] `https://link.springer.com/`
[18] `http://formulasearchengine.com/`
[19] `https://www.symbolab.com`
[20] `https://www.wolframalpha.com/`

Table 3.1: Formula Retrieval Scores for State-of-the art methods (average bpref@1000)

System	bpref Partial	bpref Full	Harmonic Mean
Approach0 [158]	**0.59**	**0.67**	**0.63**
Tangent-S [147]	**0.59**	0.64	0.61
MCAT [159]	0.57	0.57	0.57

operands in an expression as discussed previously. WikiMir built by gao2016math uses a mixture of keywords and formula representation to learn a ranking model using RankBoost. The system first processes the math query into Presentation MathML format. After that the indexed context keywords are used to get more information about the formula. Then the importance of the formula is calculated in the document, which helps them distinguish the relevant formula. Finally, a RankBoost is used to retrieve the formulae, which is then re-ranked using regular expression. kristianto2016mcat also build something similar but instead of using regex they use Lucene and the supervised learning algorithm is SVM instead of RankBoost.

3.4 Results

3.4.1 Ranking Text with Math Formula using BERT

The Intelligent Information Systems Research Laboratory at Pennsylvania State University participated in CLEF-2020 ARQMath track to contribute and develop new techniques for math-aware information retrieval. One of the main goals [161] of this track was to push mathematical question answering into an informal language scenario, specifically using data from Math Stack Exchange (MSE). We participated in Task-1: Answering Retrieval track, the goal of which was to return ranked lists of past answers given an actual recent post containing math formulas and text.

Question answering has elements of both relevance matching and semantic matching but remains a different task from document retrieval [162]. We use a two phase retrieval and re-ranking approach. In the first phase, retrieval is based on two runs: tf-idf representation using BM25 scoring and cosine similarity. Once we obtain the top 1000 candidates from the first phase, we re-rank them using contextualized BERT-

based embeddings from the math-aware language model trained on the MSE dataset. This language model makes these embeddings semantically rich. Finally, we fuse the results from BM25, cosine, and BERT-based re-ranking runs. Our contributions are:

- Achieve a competitive score that beats the best baseline in terms of the NDCG′ score.
- Achieve better results than the best submission for Text and Text+Math dependent posts.
- Propose a Masked Language Model[21] used for re-ranking candidate answers containing both text and math formulas.

Our paper has the following Sections. Section 3.4.2 discusses our two-stage retrieval approach which includes indexing and re-ranking phases. Section 3.4.3 describes our experimental setup, system configuration, and a detailed comparison of our results with the best submission. Conclusions are in Section 3.4.4.

3.4.2 Our Approach

Here we describe our two-stage cascade candidate retrieval and ranking approach. tf-idf with cosine similarity and an off-the-shelf BM25-based search platform [163] are used for the first stage. This is relatively computationally cheaper than the second more expensive re-ranking stage. For the second stage, we use a pre-trained language model to obtain semantically rich embeddings for the candidates obtained from the first stage. We then use these embeddings to rank the candidates using a cosine distance to the topic/query embedding. This ensures that the ranking captures semantics between the query and the documents. We only rely on the content of the posts which contains raw text and math formulas not using external information such as votes/scores, user_id, user score, post tags, and linked duplicate posts. **First-Phase: Retrieval** The aim of this phase is to get a sufficient number of relevant candidates to be re-ranked in the next stage. Past work has used BM25 scoring for this but we add cosine similarity ranking as well.

Indexing: We convert the MSE dataset from XML to JSON format so that it can be ingested by the search platforms we use. Because indexing all answers will

[21] `https://huggingface.co/shauryr/arqmath-roberta-base-1.5M`

potentially lose information in the questions, we generate a document by concatenating the question (Q) post text (title and body) with their corresponding answer (A) posts text (body). As such, the question body and title concatenated with the answer body become a document - one unit of retrieval. This allows us to remove questions without corresponding answers and represent the relevant posts as one large document because the relevant information the user is seeking could be in either the answer or the question post. This results in a total of 1,435,643 Q+A posts to be indexed.

We index the data using two off-the-shelf libraries - Elasticsearch (ES) and Anserini. We use two different libraries because each has its strengths and weaknesses. Anserini seems to perform better than ES in terms of relevance ranking and recall, which we observed when searching and evaluating the three training queries for the Question Answering Task provided by the organizers. Elasticsearch has a more scalable implementation of tf-idf with cosine similarity which is slow for a large dataset when using Anserini. For the formulas, we use the raw LaTeX strings. We re-rank answers based on contextualized text and formulas embeddings using RoBERTa.

Retrieval: Once all posts in Elasticsearch and Anserini have been indexed, we query the topics/queries for Task-1 to get the top-1000 candidates. For each task, each index is queried independently and later fused using Reciprocal Rank Fusion (RRF) [164] which then ranks the documents using a naive scoring formula. Given a set of posts P to be ranked and a set of rankings R from different scoring schemes, for each permutation on $1 \cdots |P|$, we compute

$$RRFscore(p \in P) = \sum_{r \in R} \frac{1}{k + r(p)}, \tag{3.1}$$

where the original work [164] suggests $k = 60$, a hyper-parameter which we keep constant. The fusion of results from two different search platforms above significantly increases the performance numbers as demonstrated in the Section 3.4.3.

Second-Phase: Re-Ranking Here we describe how we pretrain our language model and re-rank candidates obtained in the last phase.

BERT [165] is a self-supervised approach for fine-tuning a deep transformer encoder [166]. Given a sequence, BERT learns a contextualized vector representation for each token. The input representations are fed into a stack of multi-layer bidirectional transformer blocks, which uses self-attention to compute semantically rich text representations by considering the whole input sequence.

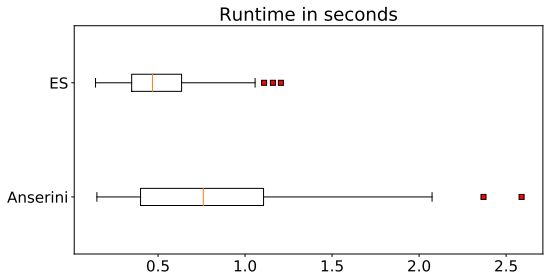


Figure 3.3: Comparison of run-times for all the topics averaged over 10 runs. Fastest topics to retrieve top-1000 for ES and Anserini were A.39 and A.88 respectively while the slowest topics were A.87 and A.47 respectively.

BERT-based systems [167] [168] [169] [74] have shown significant performance in the recent tasks of TREC-2019 deep learning track [170] with the Microsoft MARCO dataset [171]. Participants for these tasks used query-document pairs from the training data to train transformer-based models to predict relevance. However, such models rely on a massive amount of data, which is not available in our task. Therefore, instead of training for relevance, we leveraged an unsupervised model by calculating the semantic similarities of queries and documents and later using them for re-ranking.

We choose RoBERTa model [172] over BERT because in our experiments Vanilla RoBERTa achieves better NDCG′ scores than Vanilla BERT for the three preliminary training posts provided by the task organizers. Also, RoBERTa converges in fewer training steps than BERT as it gets rid of the computationally expensive next sentence prediction task. RoBERTa attains better performance on various NLP tasks than BERT. [173]. We start with the initial weights of the *roberta-base* model and further pretrain the model using MSE data. The language model is trained for a mask prediction task. Once we are done training we use the Masked Language Model (MLM) to get contextualized token embeddings averaged over the sequence length to represent the candidates from Phase-1 into a 768 dimension vector. Similarly, we obtain topic/query embeddings and rank the candidates using their cosine distance.

3.4.3 Experiments and Results

Our experiments were conducted on a 24 core machine with Intel(R) Xeon(R) Silver 4116 CPU @ 2.10GHz with 256GB of RAM with 4 RTX 2080 Ti GPUs. Default configurations were adopted in ES (cosine similarity; tf-idf) and Anserini (BM25).

Anserini runs on a single thread while ES uses a multi-threaded function. In Figure 3.3 we compare runtimes of BM25 ranking using Anserini and tf-idf based cosine similarity ranking using ES. The size of ES Q+A index is 3.6GB whereas for Anserini the size is 2.2GB.

Pretraining RoBERTa: We use the transformers[22] library's implementation of RoBERTa to train the MLM. We start with the original weights released by the authors for *roberta-base* and then further pretrain the model on the MSE dataset. Fortunately, BASE-vocabulary used in the original was able to cover the whole MSE dataset so did not train from scratch. Further, we reduce the batch size to 4 per GPU and increase step size to facilitate gradient accumulation. We kept the maximum sequence length of 512 to accommodate longer posts[23]. Q+A posts are usually longer than 512 tokens so we had to break the posts into chunks before feeding them to our system.

Once we are done with pretraining our MLM, we use it to extract the embeddings for each token in the candidate posts from the first-phase. For longer Q+A posts, we had to find a way of truncating them so that the sequence can fit in the 512 token window. We experimented with the $head - tail$ approach [174]. The $head - tail$ approach gave a lower NDCG′ score for the three preliminary train topics. Therefore, we use the *head* approach, in which we keep the first 510 tokens from the Q+A post and leave two token spaces for $[CLS]$ and $[SEP]$. This gives better results for the trained topics. This is likely because our Q+A posts include question text, so if you can find a similar question to a topic, then the answer to the similar question has a higher chance of being an answer to the topic.

We show in Table 4.3 how our system compares with other submissions and the baselines. We only include the best submissions for baselines and other systems. Our system BM25+tf+tf.BERT clearly beats the best baseline in-terms of NDCG′. Before the challenge, we had only three train topics provided by the organizers on which we could test our approach. For these three topics tf-idf with cosine similarity was running better than BM25 scoring, so we did not include BM25 in our final submission. Later we added BM25 scoring which showed a greater increase in the number of relevant retrieved documents. It can be seen that **tf.BERT** that selects candidates using tf-idf similarity and re-ranking using our pretrained RoBERTa model

[22]`https://github.com/huggingface/transformers`
[23]More training details - tensorboard link

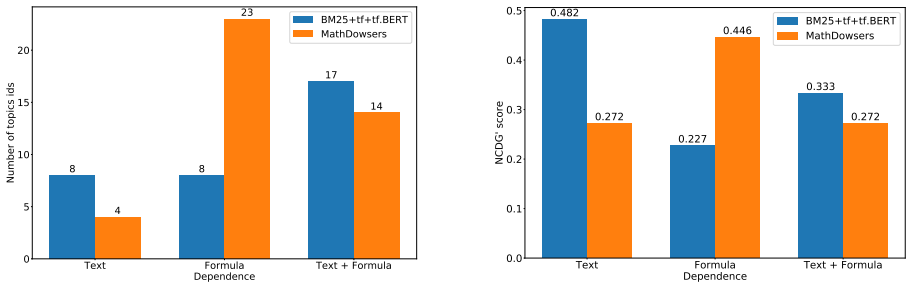
Table 3.2: Results for Task-1 compared with other submissions and best baselines. **tf** is tf-idf representation and cosine similarity-based ranking and **tf.BERT** signifies re-ranking using BERT of the candidates selected by **tf**. Reciprocal rank fusion is used to fuse the runs separated by + sign. (* represents our best run)

	Run Tag	Evaluation Measures			
		NDCG′	MAP′	P@10	Rel_Ret
Baselines	Linked MSE Posts	0.303	**0.210**	**0.417**	395
	Approach-0	0.250	0.100	0.062	441
Official Runs	PSU1 (tf+tf.BERT)	0.263	0.082	0.116	761
	PSU2 (tf)	0.228	0.054	0.055	761
	PSU3 (tf.BERT)	0.221	0.046	0.026	761
Other Systems	MIRMU	0.238	0.064	0.139	708
	MathDowsers (Task-1 Best)	**0.345**	0.139	0.161	804
Unsubmitted Runs	BM25+tf+tf.BERT*	0.314	0.097	0.149	**902**
	BM25+tf	0.304	0.098	0.151	875
	BM25	0.246	0.078	0.139	660

does not achieve the best performance. But when the rankings of tf.BERT and tf-idf are fused, NDCG′ [175] score is substantially boosted. This is because the BERT ranking solely relies on semantic similarity whereas the tf-idf replies only on word frequency. Fusing these two runs seems reasonable since the posts which have a higher rank in both the runs are boosted even higher in the final ranking list. Note that BM25+tf+tf.BERT achieves the maximum number of relevant retrieved posts. It is important to note that the tf runs are not as consistent since ES does sharding and round-robins between different shard searching. Thus, did multiple runs to achieve the above scores.

Comparison with Other submissions: Our best unsubmitted run, BM25+tf+tf.BERT was compared with other submissions, among which MathDowsers is the system that achieved one of the best results in Task 1. Submissions by Approach0 and MathDowsers leveraged Tangent-S [176] and Tangent-L [177], which are Symbol Layout and Operator Tree-based systems giving more attention to the math formula in the text. Our system does not use a different representation for formula and text. and therefore seems to suffer for formula dependent topics. We believe representing math content as trees and separately from the text content could have benefited our scores.

In Figure 3.4 we see that our system is better for the topics that depend on text



(a) Number of topics for which each system had higher NCDG′ than the other system.

(b) Average NCDG′ over total topics depending on different types of topics.

Figure 3.4: Comparison of runs for the 77 topics in Task-1 based on dependence class of the topics. We clearly see that for the topics that depend on Text and Text+Formula our system performs better.

and text+formula. For Text dependent topics MathDowsers had 4 topics for which there NDCG′ score was higher than our best run, while our system performed better in 8 topics. For the 31 text+formula dependent topics our system had better NDCG′ score for 17 topics. MathDowsers achieves a higher ($\approx 96.4\%$ higher) NDCG′ score for the topics which are only formula dependent. In contrast, our system is better ($\approx 22.42\%$) at ranking posts containing both formula and text. This is attributed to the contextualized embeddings which our pretrained MLM can produce. It models equations with surrounding text and hence has better performance in text+formula topics. The difference ($\approx 77.2\%$) is even more when we compare text dependant topics.

3.4.4 Conclusion

Overall, our participation in the ARQMath Track helped in our understanding of how to improve multi-modal search (text+formula) by exploiting state-of-the-art text embedding and information retrieval models. In terms of effectiveness, our most effective run BM25+tf+tf.BERT was able to outperform the baselines and all submissions except Mathdowsers in terms of NDCG′. Our system achieved the best results for queries that depend on text and text+formula.

Future work would include the use of MSE dataset to get question-answer post pairs to train a better ranking model. It would also be helpful to investigate how important a formula is in a question post to retrieve relevant answers.

3.5 Visual Retrieval of Mathematical Formulas

3.5.1 Introduction

Mathematical Information Retrieval (MIR) concerns retrieving documents or document contents containing mathematical expressions. Formula retrieval is the task of retrieving relevant formulas for the given formula query. Mathematical formulas can be unique or rare in a collection, and providing the vector representation for formula results in more robust representations for unseen and unique formulas. Although many information retrieval and natural language processing (NLP) tasks benefit from dense vector representations of words, embedding models have not been studied much for mathematical formulae. Although some works show word embedding models do not work well with mathematics [178], other works have demonstrated that formula embedding is useful for MIR [127], which sparks more efforts in this direction. The best performing system on NTCIR-12 [146] formula browsing task uses an n-gram embedding model on formulas appearance and semantic [179]. This model, called Tangent-CFT, uses tuples to represent paths in Symbol Layout Trees (SLTs) for formula appearance by the location of symbols on writing lines (e.g., as given in LaTeX), along with Operator Trees (OPTs) representing the hierarchy of operations and arguments in a formula (i.e., the mathematical semantics). Considering the formula $x^2 = 0$, Figure 3.1(a) shows the SLT representation where edge labels are spatial relationship of symbols, (for instance 2 is located **a**bove x) whereas Figure 3.1(b) represents the OPT representation of this formula. This model then applies the fastText n-gram embedding model, averaging vector representations for tuples from OPTs and SLTs to obtain a final formula embedding. The embeddings capture partial/rough similarity between formulas well, which complements retrieval using concrete paths from OPTs/SLTs, which tends to identify highly similar formulas more reliably. By combining Tangent-CFT with the Approach0 model [158] using path-based retrieval in OPTs, state-of-the-art performance for the NTCIR-12 Wikipedia Formula Browsing Task (hereafter WFBT) is achieved. But Tangent-CFT fails to look at the whole formula and understand its structure as it is based on tuples and n-grams. An image-based embedding can provide a representation of how the formula is structured on the 2-D plane.

While previous MIR systems focused on tree-representation of the formula to

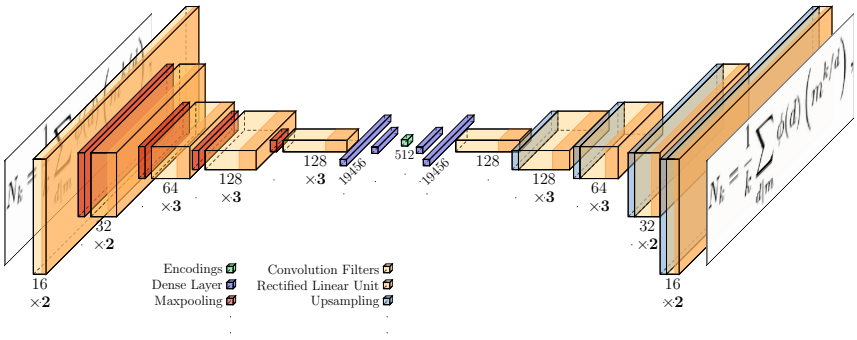


Figure 3.5: The architecture of our Convolutional Autoencoder (CAE). The encoder is on the left and the decoder is on the right. Each convolutional layer (orange) is followed immediately by a max-pooling layer (red). The number of convolutional filters (e.g., 16, 32) is displayed below each convolutional layer. The number of filter blocks (e.g., ×2, ×3) is displayed at the bottom. The final embedded vector of 512 dimensions is shown at the center of the diagram.

apply path-based retrieval (systems such as MCAT [180], Tangent-S [147]) or to learn a vector representation of formulas (systems such as Tangent-CFT), a few works have focused on visual retrieval of mathematical formulas. The most related system to this is Tangent-V which provides a domain-agnostic approach to visual retrieval [181], using edges from Line-of-Sight (LOS) graphs over connected components (i.e., 'black blobs') in an image. This has been applied for retrieving *handwritten* math in lecture videos from formula queries given in LaTeX (e.g., from course notes).

In this paper, we propose an approach that encodes math formulae as a whole into dense representations using an auto-encoder [182], comprised of deep convolutional neural networks (CNNs). The work done by [183] use a stack of CNNs as encoder but instead of reconstructing the image using a decoder, they use unsupervised proxy tasks to train the embeddings - latex-symbols as labels. The model based on proxy tasks only learns what symbols are present in the formula image and fails to capture the overall structure of the equation as a whole, which our model is able to do.

3.5.2 Model

We first establish a simple baseline model using text-based tf-idf representation. After this, we train our CAE network to learn the math formula image-based embeddings. Finally, we concatenate the text-based representation and the CAE embeddings to

get a final representation of the formula.

3.5.3 Subword TF-IDF Baseline

For tokenizing math formula we use byte-pair-encoding (BPE) [184] to break down the formula into frequently occurring subword units [24]. This tokenization is language independent and does not rely on a specific pre-processing pipeline. It is purely data-driven. We chose our vocabulary size to be 256, which is the maximum number of math symbols possible in each math-font family. Once we have tokenized the formula, we get the tf-idf representations of the formula. This gives us a robust representation of the math formula - a baseline, upon which we can build our model.

3.5.4 CAE

Our autoencoder for formula images is shown in Figure 3.5. The system first extracts features using multiple CNN layers and then flattens the 2-D grid into a 1-D vector. Then two fully connected (FC) layers encode the vector resulted from the deep convolutional encoder into a vector of 512 dimensions. The encoded features are then fed to another two FC layers followed by a deep convolutional encoder to reconstruct the formula image. The input and output images of this network have the same size ($w \times h \times c$), where w, h, and c are width, height, and the number of channels, respectively. In this paper, we use $c = 1$ because our images are black and white, which also reduces the number of filters. Our choice of activation function at every layer was Rectified Linear Unit (ReLU) because it avoids the problem of vanishing gradients. Here we explain each part of our neural network model:

Encoder A multi-layer CNN with max-pooling layers has been widely used as a standard architecture to extract visual features in images [185, 186]. The kernel size (3×3) and the number of filters in each layer were by the RED-Net model [187], which is originally motivated by VGG-Net [139]. Each block of CNN filters is followed by a max-pooling operation, which reduces the data dimensions from 600×60 ($36k$) to $38 \times 4 \times 128$ ($19,456$). The 2-D output from the encoder is flattened to a 1-D vector of size $19,456$ before being fed into a fully connected network with a ReLU activation. This 1D vector is then compressed to a latent representation of size 512.

[24] https://github.com/google/sentencepiece

Decoder On the dense-encoder side, the 512-D embedding is then again fed into a fully connected dense decoder network and reshaped into a 2-dimensional grid, which is input to the deep convolutional decoder. This reconstructs formula images. Each block has an upsampling or max unpooling layer followed by convolutional filters. The architecture is an exact mirror image of the encoder network.

3.5.5 Experiments

Data Preprocessing. We used the NTCIR-12 WFBT dataset [146], which provides more than 590,000 math formulae from English Wikipedia articles, out of which 328,828 are formulae having unique LaTeX strings. *pdflatex*[25] was used to generate the PDFs for all WFBT formulas, after which *pdftoppm*[26] was used to convert the PDFs to 300dpi raster images.

To create images we follow an approach similar to that in recent work [183]. But instead of 333×32 pixels we chose a bigger resolution of 600×60 to accommodate 98% of the dataset without cropping. Bigger expressions like matrices were cropped to fit in this frame. Cropping was the more logical choice as force-resizing the formula images would distort the math symbols. We render the formula in the center of the frame. We do the same for NTCIR-12 WFBT queries to get query images.

Training and Retrieval. We split our WFBT document collection into training, testing, and validation sets at ratios of 8:1:1. We make sure that only the test set has documents and queries labelled in NTCIR-12 WFBT, so that the labelled documents remain unseen by our model. The network is trained to minimize reconstruction errors, and so we use the mean-squared error for our loss function. We use early stopping to stop once validation loss no longer decreases. Figure 3.6 shows the reconstructions of some formulae from the test set. Once our model is trained, the decoder layers are removed. We then use the encoder to compute embeddings for the whole WFBT collection, including all the queries. In general, we observe that better the reconstruction the better are the retrieved results. We believe that this is because there are a lot of equations that have a specific structure and the autoencoder is able to learn the frequently occurring structures well.

For retrieval, formulas are ranked using the cosine similarity between the query formula embedding vector and each formula in the collection. We do this for our

[25]https://www.tug.org/applications/pdftex/
[26]https://linux.die.net/man/1/pdftoppm

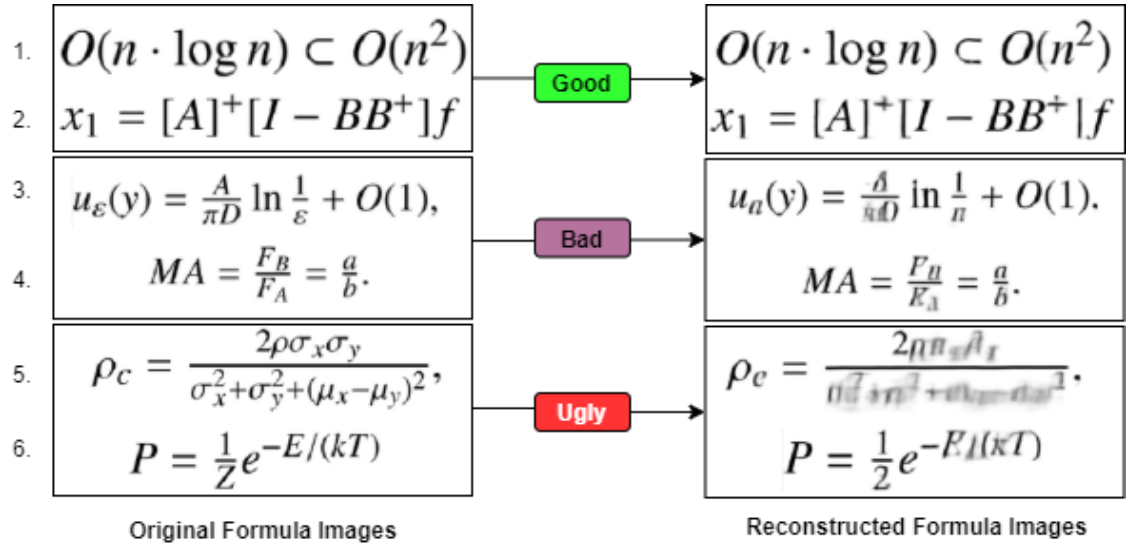


Figure 3.6: Reconstructed formula images from our CAE. In most cases, the input is constructed faithfully from the 512 dimension embedding vectors. Reconstruction quality suffers for sub-expressions that rarely occur at certain image locations.

baseline model - BPE (256-D), CAE(512-D) and finally a concatenation of both these vectors (768-D). For each query, we return the top 1000 formulae from the NTCIR-12 WFBT test collection. Because the embedding size is relatively small, it took approximately 0.52 second per query (CPU - Intel(R) Xeon(R) Silver 4116 CPU@2.10GHz). The complexity of the tree-based matching algorithms is polynomial in the number of symbols [147]. In our case it is $O(nd)$, in which n is the total number of documents in the collection, d is the number of dimensions.

3.5.6 Results and Discussion

NTCIR-12 WFBT provides 20 concrete math formula queries (i.e., formulas without wildcards). The task was evaluated using two assessors per hit, who rate each matched formulas included in the evaluation pool using a graded relevance score {0,1,2}, with 2 being the most relevant. The final relevance rating for each hit is the sum of the relevance scores by both assessors. Therefore, the final ratings for a formula range from 0 to 4 with 4 being the most relevant.

Evaluation Protocol. We use Binary preference-based measure (bpref) [188] for highly relevant documents (score of 4). MAP metrics compare systems using the

1**Type**	1**Model**	1 **bpref**	1**p@5**	1**NDCG**	1**MAP**
Tree	MCAT	0.63	**0.31**	0.71	0.61
	Tangent-S	0.62	0.28	0.70	0.66
Vector	Tangent-CFT	0.63	0.30	**0.84**	**0.71**
	BPE(baseline)	0.60	0.26	0.55	0.61
	CAE	0.42	0.18	0.26	0.40
	BPE+CAE	**0.64**	0.26	0.50	0.63

Table 3.3: Results for NTCIR-12 WFBT. We include state-of-the-art tree-based systems and embedding-based models for comparison over the 20 concrete queries.

same relevant results while reciprocal rank (RR) and Precision@k measures tend to produce lower scores for systems that do not participate in the original task, as unevaluated documents from the collection are treated as irrelevant.

NTCIR-12 Results. We compare our model with 2 types of MIR models (Table 3.3). The tree-based models include MCAT [180] and Tangent-S [181]. The embedding-based model is Tangent-CFT [179]. Our model achieves state-of-the-art in highly relevant *bpref* score. Although our results are not state-of-the-art in other measures, there are some subtle characteristics like the position of variables in an equation that the network can capture from the data. As our embeddings rely only on visual appearance at the pixel level, no information about the individual math symbols in the formula is available for our deep neural network. There are advantages of representing formulae using visual cues which are discussed in the following sections.

Comparison with Tangent-CFT Embeddings. The embeddings retain the symbol layout in retrieved formulae. An interesting finding here is that our embeddings give more precedence to the structure of the formula rather than the individual variables. The embeddings were able to retain simple linear formulae very effectively. For example, for query $ax^2 + bx + c = 0$ Tangent-CFT reports a bpref measure of 0.7407, while our embeddings achieve a bpref of 0.8889. It seems to be because the formulae retrieved have the same structure as the query. In this case, our formulae have different variables than the query, so Tangent-CFT embeddings did not perform as well. For instance, two formula $ad^2 + bd + c = 0$ and $an^2 + bn + q = 0$ were ranked higher in the ground truth as well as our system as compared to Tangent-CFT.

Another example is the formula query $O(mnlogm)$, where our model can retrieve two formulas $O(VElogV)$ and $O(KNlogM)$ that are fully relevant to the query while

2 **MathWiki-10**	2**MathWiki-12**	2**MathWiki-7**
$L(\lambda,\alpha,s)=\sum_{n=0}^{\infty}\frac{\exp(2\pi i\lambda n)}{(n+\alpha)^s}.$	$O(mn\log m)$	$0\to G^\wedge \overset{\pi^\wedge}{\to} X^\wedge \overset{\imath^\wedge}{\to} H^\wedge \to 0$
$L(\lambda,\alpha,s)=\sum_{n=0}^{\infty}\frac{\exp(2\pi i\lambda n)}{(n+\alpha)^s}.$	$O(mn\log m)$	$0\to G^\wedge \overset{\pi^\wedge}{\to} X^\wedge \overset{\imath^\wedge}{\to} H^\wedge \to 0$
$L(s,\chi)=\sum_{n=1}^{\infty}\frac{\chi(n)}{n^s}.$	$O(\log\log N)$	$0 \to \mathcal{A} \overset{\phi}{\to} \mathcal{B} \overset{\psi}{\to} \mathcal{C} \to 0$
$L(\lambda,\alpha,s)$	$O(N/\log N)$	$0 \longrightarrow \mathcal{A} \xrightarrow{\alpha} \mathcal{B} \xrightarrow{\beta} \mathcal{C} \longrightarrow 0$
$L(\chi,s)=\sum_{n=1}^{\infty}\frac{\chi(n)}{n^s}$	$O(c_{d,\epsilon}log(n))$	$0 \longrightarrow \mathbb{Z}\langle e\rangle \xrightarrow{\partial_1} \mathbb{Z}\langle v\rangle \longrightarrow 0$
$L(s,\chi)=\sum_{n=1}^{\infty}\frac{\chi(n)}{n^s}$	$O(nloglogn)$	$0\to H \overset{i}{\to} G \overset{\pi}{\to} G/H \to 0$

Table 3.4: Example Search Results for NTCIR-12 WFBT ranked by cosine similarity of embedding vectors. The first row is the query and the next five are the nearest neighbors of the query.

Tangent-CFT fails to retrieve them. This is again due to Tangent-CFT preferring common variables with the formula query. This may arise from Tangent-CFTs use of n-grams only; an embedded tree in the model uses symbol types rather than specific values, but the model still prefers identical symbols.

Figure 3.7 provides a query-wise bpref score comparison. Although the bpref score is better for our system as compared to Tangent-S, other metrics fail because our system doesn't account into the fuzziness or approximations in a formula which Tangent-CFT does really well. For example - x and a in the equations $x+y$ and $a+b$ are just variables which is understood by Tangent-S and can be considered similar, but our system will say they don't look the same so these equations are not similar. That is why our system performs bad in the other measures. Although math formulae retrieved using our model are visually similar to the queries, they may not be relevant in the context of mathematics, physics, or other domains, even if they share the same variables. Table 3.4 demonstrates the CAE model capturing visual structures when matching against the formulae in the queried collection. However, in the NTCIR-12 ground truth, only the first formula shown in the first row in Table 3.4 has the highest relevance score of 4. The rest have lower relevance scores of 1 or 0. Also, the CAE is not good at learning infrequent formula structures and symbols. It only reconstructs the math expressions effectively of expressions similar to what it has been trained on.

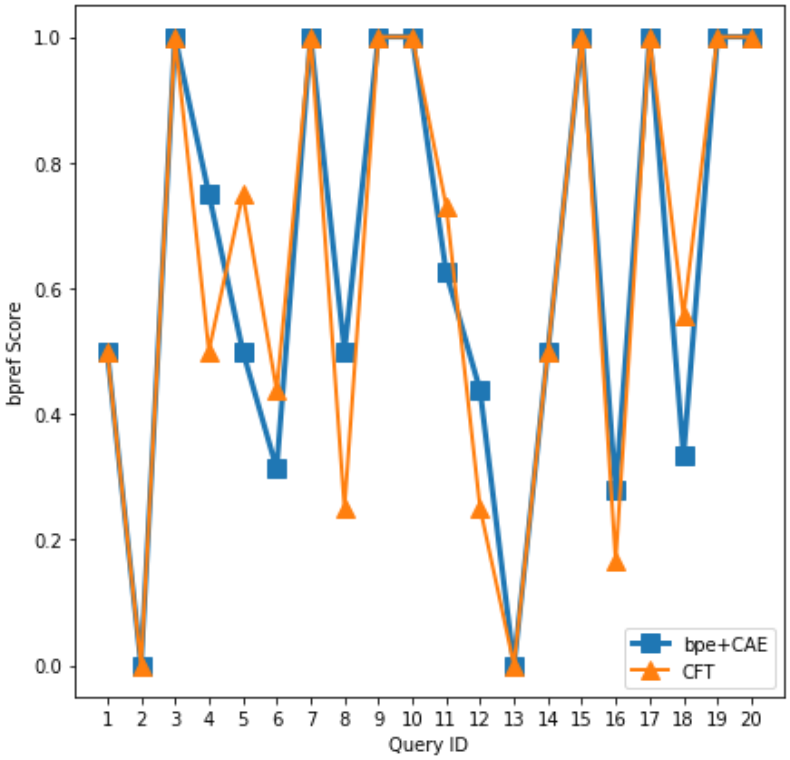


Figure 3.7: Plot of query-wise bpref score. It can be seen that for query with ID MathWiki-4,8,12 and 16 our system beats Tangent-CFT while for the MathWiki-5,6,11 and 18 Tangent-CFT is better. For the other 12 queries, the performance is the same.

3.5.7 Conclusion

We have presented a deep autoencoder that can convert math formula images into a fixed-length vector. We evaluated our CAE model using the NTCIR-12 Wikipedia Formula Browsing Task, and compared it with both tree-based models (MCAT and Tangent-S) and embedding-based model (Tangent-CFT). Formulae that look similar might not be relevant to the queries in the NTCIR MathWiki Task, but our embeddings combined with other embeddings like Tangent-CFT can be used to improve performance (e.g., to allow the visual structure to be better captured without consideration for specific symbols). Possible future improvements include a multi-modal embedding model which also incorporates dense symbol information instead of a static tf-idf representation.

3.6 Contributions & Implications

3.6.1 Integration with CiteSeerX

All the successful and practical technologies developed in this research work will eventually be integrated into CiteSeerX. The process includes identification, extraction, indexing and querying of math expressions in the 11 million CiteSeerX academic article library. We believe that this will push the envelope of multi-modal search and could in the future generalize to similar domains like - Chemistry equations.

3.6.2 Research Contributions

The proposed work will make the following contributions:

1. Creation of search engines for math and pseudocode search over Wikipedia and CiteSeerX. APIs will be explored for database users. Note that the CiteSeerX documents are publicly available, and the Wikipedia Foundation freely allows reuse of its contents with proper attribution.

2. New algorithms for indexing, ranking, and information retrieval will be created.

3. Software for algorithms and search engines will be make publicly available on GitHub. Such code will be useful in related research and will be applicable to other uses and projects.

4. Produce collections of isolated formulae and pseudo code, and related databases for others to use in their research.

Implications and Integration with CiteSeerX If the proposed work is successful, it can have socio-technical impacts on the society by improving access to mathematical information for experts and non-experts. This will lead to enhanced use and knowledge of mathematics in the society. As the discovery of math will be easier, it can help discover new and old mathematics for new and old problems. A math-aware search engine can also help educators and instructors find new resources to teach mathematics. Also, a robust math-aware search engine can attract students and others into STEM areas and increase our mathematical and code literacy.

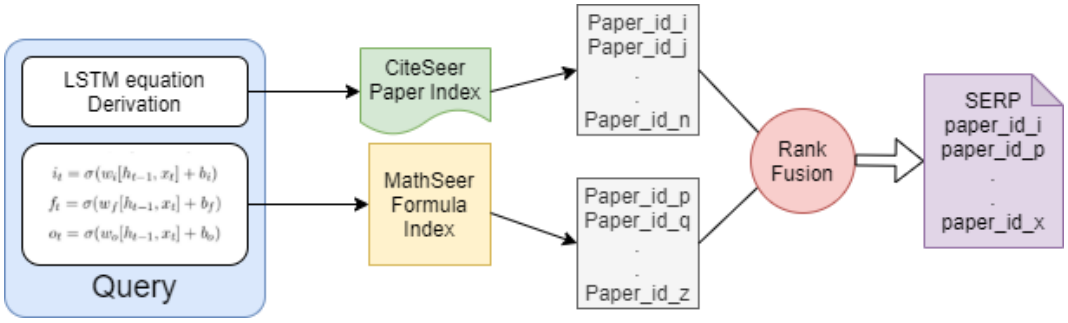


Figure 3.8: An example of fusing runs from Math search and text search. We use reciprocal rank fusion to do this. Paper_id correspond to unique paper id in CiteSeerX.

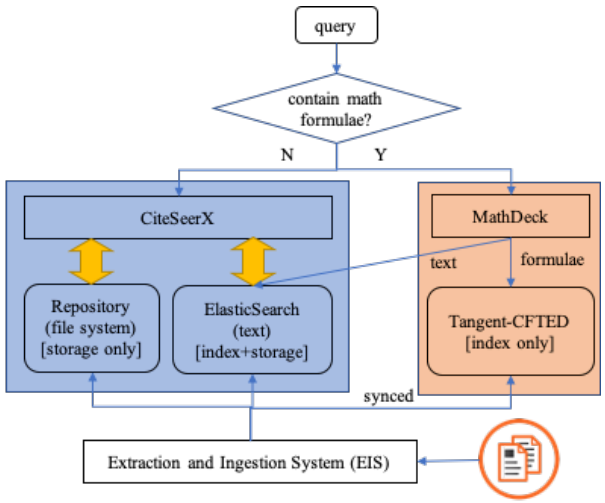


Figure 1: The top–level architecture of a math-aware CiteSeerX. The blue region is the CiteSeerX and the orange region is the MathSeer.

Figure 3.9: High level architecture of querying and search workflow

We plan to integrate the methods and techniques developed for Math with CiteSeerX providing the users a interface to search different modalities. The top-level architecture depicted in Figure 3.9 illustrates how an incoming query is processed depending on whether it is a complex query (containing math formulae) or not. The architecture also shows that CiteSeerX ingestion system should ensure that math entities in MathSeer and CiteSeerX are synced so any math formula in MathSeer has a corresponding paper in CiteSeerX (working as a foreign key). Also, in the Figure 3.8 we demonstrate how we plan to merge the rank lists and show them in the search engine result page. The text and math in the query are queried separately on ElasticSearch and math expression index. Both searches will return a ranked list of

paper-ids from CiteSeerX. These ranked lists will eventually be merged into one list which will be showed on the search engine result page.

Prototype Stage. We want to showcase CiteSeerX extraction pipeline working tightly with MathSeer so we will develop a prototype for a small dataset, more specifically articles from ACL[27]. This prototype will serve as a showcase of what can be done when we merge the two search platforms. We expect to encounter minor engineering challenges like integration and building services for standalone programs. Once we have build this prototype, the programs and architecture produced for it can be used to build and test the actual scalable search platform.

[27] `https://acl-arc.comp.nus.edu.sg/`

Chapter 4

Understanding Scholarly Information Need

Academic search engines have served the research community for years, yet there is little work done on understanding the taxonomy of query semantics. In this work, we present our findings of analyzing the query log of an academic search engine in the past four years. We study the distribution of query intents to understand the information requested by users. We classify query strings by topics using shallow and latent features captured using a customized word embedding model. To this end, we create a dataset that has scientific keywords and titles labeled with fields of study. This dataset is later used to train a classifier that discriminates query logs by topics. Our work will help to train better learning-based ranking functions that improve user experiences for an academic search engine. In addition, we anonymize our 78 million query logs and make them available to the research community for further exploration.

4.1 Introduction

Academic search engines with full text search capabilities like Google Scholar, Semantic Scholar, and Microsoft Academic Search hold special positions in academia. They provide the researchers quick and easy access to the plethora of research articles on the web. In spite of their usefulness, there is limited work in the area of academic search query understanding. The increasing online traffic for academic search makes it necessary to study the information needs in academic search, which is important

in guiding the development of ranking models. Query understanding models have been developed to probe users' information needs, including tasks such as query classification, intent understanding, segmentation, suggestion, and rewriting [189]. Query intent understanding for academic search engines aims at understanding how users search for a particular page or an article, such as by title, keyword, or author names. One goal of query topic classification is to assign a topic to a search query. In the context of an academic search engine, these topics are usually research-related.

Existing work has analyzed search engine logs to understand academic queries [190]. Previous work has classified queries into navigational and informational categories [191, 192]. This high-level categorization may not be sufficient to understand users' information need. This motivates us to examine the semantics of queries in a fine granularity by classifying them by intent and topics. Previous work has demonstrated that if a topic can be identified for a search query, the search results are significantly improved [189]. Research Subject classification using scientific abstracts has been well studied in a recent work [193]. This task is challenging due to vocabulary overlap between similar domains. However, topic classification on short text such as search queries can also be challenging due to the limited and ambiguous information.

In this work we focus on topic classification and intent understanding of search logs for an academic search engine, using CiteSeerX as a case study [24]. We introduce the concept of academic query intent understanding and research topic classification. We release a dataset containing millions of anonymized log records across 4 years. To the best of our knowledge, this is the first work that studies academic query logs at such a scale.

4.2 Data Acquisition and Preprocessing

CiteSeerX is a digital library search engine, currently indexing over 10 million open-access scientific documents [60]. The search engine uses Apache Solr as the indexer and provides full-text search. We analyze queries generated by the search engine between January 2017 and January 2021. The total number of raw search queries for this period is 78,124,884. The original query logs are generated by Tomcat (version 6.x) across three web servers. We select logs recording users' attempts to query using search interfaces. There are several possible interfaces the users may input a text

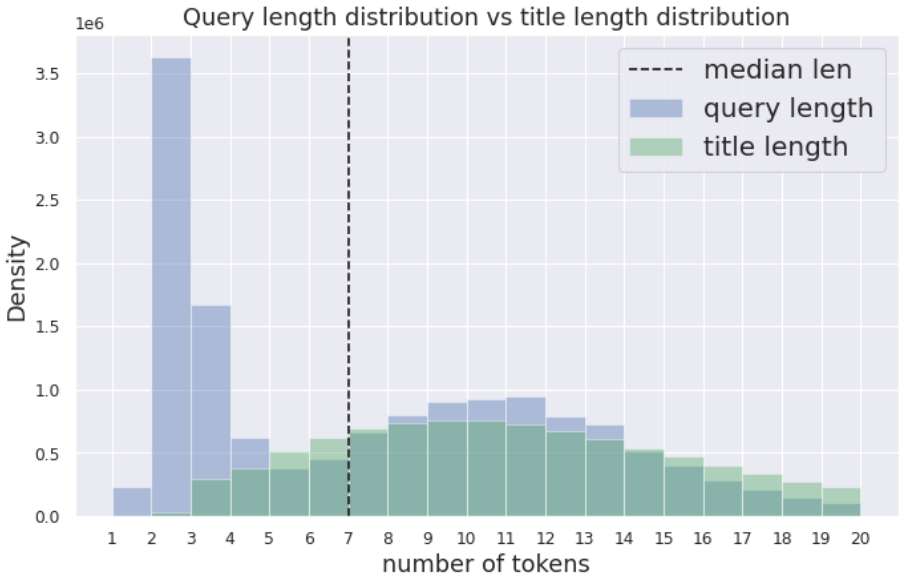


Figure 4.1: The length of queries peaks at length=2, which are bigrams and mostly author mentions. The title distribution from Microsoft Academic Graph (MAG) is very close to our query log distribution.

search. The search box for CiteSeerX[1] is available on the homepage, search engine result page (SERP) and a paper's summary page, or the advanced search page. We merge queries from all interfaces on all web servers.

For each search query log, we extract the following information. (1) Raw query string. (2) IP address of the request. (3) User-agent e.g., "Baiduspider 2.0" and "Chrome 45.0.2454". (4) User-agent Type e.g., "Spider", "PC", and, "Other". To anonymize the logs, we hash all IP addresses, ensuring the same hash for the same IP, which allows us to group requests by IP for further analysis.

We further remove logs matching the following patterns.

[nosep]Duplicate raw query logs (14,759,852 unique queries). Queries containing obscene words in a controlled list[2]. Queries with characters other than alphanumeric and special characters, e.g., Chinese characters. Queries containing more than 40 tokens.

Figure 4.1 shows the query length distribution. The peak appearing at bi-gram and tri-gram queries are predominantly contributed by keywords and author names.

[1] `http://citeseerx.ist.psu.edu`
[2] `https://www.cs.cmu.edu/~biglou/resources/`

4.3 Methods

4.3.1 Query Intent Classification

Identifying the intents of user queries is important for improving the ranking quality of search engines. For example, if a user searches a paper title, incorporating an exact title-match would greatly improve the ranking. Similarly, if the users are searching for author names, the ranker should include author-name disambiguation and recognition.

Academic search has 3 major intents. Users sometimes mix basic intents and query author names with keywords.

> [nosep]Concept/keyword search: Research concept e.g., “relational reinforcement learning". Title search: Exact title of a paper, e.g., “Acknowledgement Entity Recognition in CORD-19 Papers". Author search: Searching a specific author, e.g., “Chris Manning".

We use a supervised model to classify queries into three intents above. To build the ground truth, we obtain 48,472 titles from SciDocs [194], a benchmark dataset created for research subject classification, which includes an equal number of samples from diverse research subjects. We generate keyword samples by automatically extracting noun phrases from abstracts of papers in this dataset using a grammar-based chunking method. We classify author queries using a pre-trained named entity recognition (NER) model.

4.3.2 Query Topic Classification

Our goal is to understand what research topics have been searched and the distributions. We adopt a supervised machine learning model to classify the queries into research subjects. In SciDocs, each abstract is associated with a research subject label. We extract noun phrases from the abstracts and label each phrase with the research subject of the abstracts where they are extracted.

Table 4.1: Tasks and their respective datasets used in this paper. Abstracts from SciDocs were used to extract keywords which helped us augment data for each task.

Task	Dataset	Remarks
Title classification	SciDocs	19k samples for each class
Query Topic classification	SciDocs	8k noun phrases and titles for each research topic
Word representation learning (Scientific fastText)	MAG	500k samples for each research topic

4.4 Experiments and Results

4.4.1 Query Intent Classification

We use a 2-step filtering method for intent identification. In the first step, we identify the queries that contain author mentions and filter them out. The second step uses a classifier based on basic features [195] to identify if a string is a title or a keyword.

Step 1: Author Mention Identification

To identify author mentions in queries, we use the implementation of named-entity resolution by spaCy[3], which features a sophisticated word embedding strategy using subword features and "Bloom" embeddings, a deep convolutional neural network with residual connections, and a novel transition-based approach to named entity parsing. A preliminary manual examination on 100 random queries indicates the accuracy of the NER on our corpus is 93%. We exclude the queries which have author mentions in them. The remaining queries are passed to the title classifier.

Step 2: Title Classifier

The classifier extracts seven features from each query string, including document length, stopword count, punctuation count, number of words starting with uppercase letters, the minimum TF-IDF, the maximum TF-IDF, and the median TF-IDF of words. To calculate the TF-IDF of the queries and the train/test data, we sampled

[3] `https://spacy.io/universe/project/video-spacys-ner-model`

Table 4.2: Intent distribution and their samples form logs

Intent (%)	**Log Samples**
Author Search (13.3%)	Michael J. Schöning Mohammed Petiwala Youngkil Choi
Title Search (37%)	Automatic induction of FrameNet lexical units next-to-leading order perturbative qcd correction N-body spacetime constraints
Keyword Search (49.7%)	rapid event capture kinetics scalable compiler framework 4-bit reversible circuit

500k abstracts and titles from each of the 19 level-1 research topics defined in the Microsoft Academic Graph 4.1 so we have coverage of the scientific vocabulary across the research topics.

We use the SciDocs dataset to build the training data. We extract noun phrases from abstracts in this dataset and label them as keywords. The paper titles from this dataset are directly adopted as query samples labeled as "title". We randomly down-sample noun-phrases extracted to match the size of samples labeled as "title". Finally, we obtained a dataset containing a total of 96,944 titles and keywords. We use K-fold validation while training our classifiers. We use an 8:1:1 split for our train, validation, and test data. Hyperparameters include 100 trees in the forest and the Gini index to measure the quality of the split. Extracting basic features [195] and training a random forest classifier on it yields F1=0.96.

Query Intent Distribution

Using this classifier we classified 14 million unique CiteSeerX queries obtained in Section 4.2. The distribution of title, keyword, and author categories is shown in Table 4.2. Keywords dominate the search queries, which is consistent with the observation in Figure 4.1. It is interesting to see that author search constitutes about 14% of the search queries which indicates the importance of author name recognition and disambiguation in an academic search engine. Exact title matches take about 37% of all queries, which can be identified as a "navigational" query [191], where the users know what they want. Our results indicate that exact title search and author name

Table 4.3: Classification results for query topic classification.

Model	Precision	Recall	Macro Avg, F1
KNN	0.42	0.42	0.41
CNN	0.48	0.47	0.47
BiLSTM	**0.53**	**0.52**	**0.52**

search should be considered when designing a ranking function for an academic search engine. Table 4.2 shows some examples from the actual logs.

4.4.2 Query Topic Classification

Unlike query intent classification, in which the syntactical and lexical information is sufficient, classifying queries by topics requires the model to understand the semantics represented by latent features. We created a dataset consisting of 150k samples from the SciDocs dataset. We extracted noun phrases from each abstract and labeled them with research topics. Then we removed noun phrases that occurred more than once and in abstracts across multiple research topics, e.g., “state of the art", “the elements", and “engineers". We retained noun phrases that occurred in only one research topic, e.g., “polynomial sequences" from mathematics, and “the medieval style" from art. The extracted dataset includes titles and keywords from 19 research topics.

Training Scientific FastText

We reused the 500k abstracts and titles extracted for query intent classification to train a language model called SciFastText. A skipgram model such as FastText trained on general text usually cannot correctly produce dense vector representations for all scientific text because of the large number of domain-specific out of vocabulary tokens. To overcome this limit, it is necessary to retrain the word embedding model using scientific text. In fact, we compared a FastText trained on the Common Crawl data with SciFastText on the classification task and found that the F1 score of the latter is at least 2% higher. We do not use SciBERT as it has been trained only on Biology and Computer Science papers.

We compare three classifiers using the SciFastText embedding. The settings of each classifier are below. (1) **KNN**: The number of neighbors is set to 10. (2) **CNN**: The architecture contains 2 layers of 1-D convolutions. The number of filters is set to

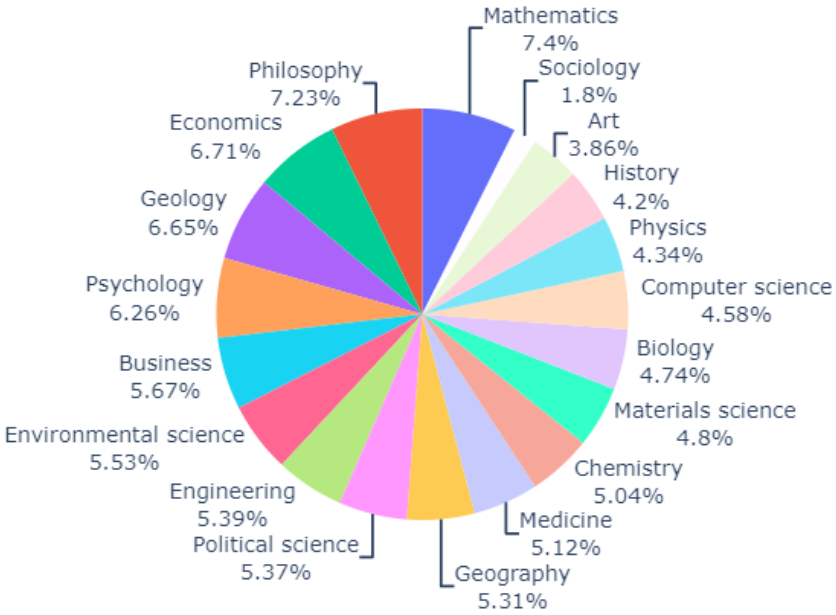


Figure 4.2: Research topic distribution of the query logs

64. The dropout rate is 0.2. The max sequence length is 20. The model adopted the batch normalization, the cross-entropy loss with the Adam optimizer. (3) **BiLSTM**: The architecture contains 2 layers of Bidirectional LSTM. The number of units in each layer is 256. The dropout rate is 0.2. The max sequence length is 20. The model adopted batch normalization, the cross-entropy loss with Adam optimizer.

We tracked our validation loss and used early stopping to make sure that the models do not overfit. BiLSTM outperforms the other two models, achieving a macro average F1 score of 0.52 (Table 4.3). We held 10% of the data as the test set. We found that identifying certain research topics seemed more difficult than the others. For example, Geology achieved an F1 of 0.72 but Sociology achieved an F1 of 0.23. We believe this was because of the lack of domain-specific scientific vocabulary used in these domains.

The query topic classifier performed well in terms of average precision (AP), which summarizes a precision-recall curve as the weighted average of precision achieved at each threshold, for several research topics, such as Geology, Chemistry, Material Science, Medicine, Biology, and Physics (Figure 4.3). We analysed the confusion matrix of the predictions for our test set and found that the classifier does not perform

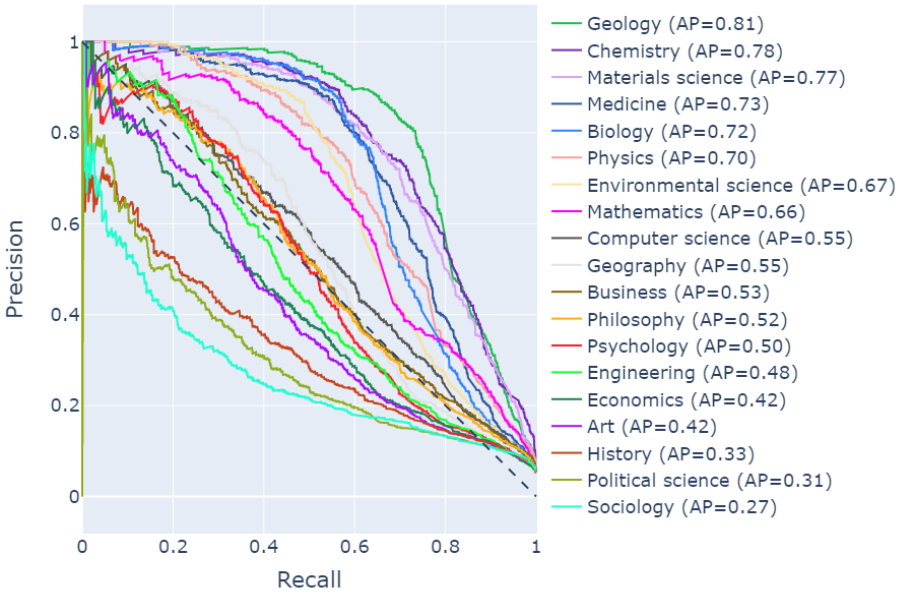


Figure 4.3: Precision-Recall curves for query topic classification sorted by AP for each research topic.

well for topics with significant overlap across queries in other topics. For example, Economics and Business, Art and History are often confused.

Using the trained model we predict the research topics of the query logs in CiteSeerX. We find that Mathematics is the most searched research topic (Figure 4.2). It should be noted that our classifier often confuses Computer Science and Mathematics short-text.

4.5 Conclusion

In this work, we present a preliminary effort of understanding queries of an academic search engine in two aspects: query intent and topics. Our results indicate it is important to incorporate author names and exact titles as features while designing a ranking function. We demonstrate the challenges with academic query topic classification and showcase preliminary results by our classifiers on the CiteSeerX data. This log dataset has been anonymized and shared with the research community for further exploration.

Chapter 5

COVIDSeer : Extending the CORD-19 Dataset

We develop an enhanced version of CORD-19 dataset released by the Allen Institute for AI. Tools in the SeerSuite project are used to exploit information in original articles not directly provided in the CORD-19 datasets. We add 728 new abstracts, 70,102 figures and 31,446 tables with captions that are not provided in the current data release. We also built a vertical search engine *COVIDSeer* based on the new dataset we created. COVIDSeer has a relatively simple architecture with features like keyword filtering, and similar paper recommendation. The goal was to provide a system and dataset that can help scientists better navigate through the literature concerning COVID-19. The enriched dataset can serve as a supplement to the existing dataset. The search engine, which offers keyphrase-enhanced search, will hopefully help biomedical and life science researchers, medical students, and the general public to more effectively explore coronavirus-related literature. The entire data set and the system will be made open source.

5.1 Introduction

5.1.1 Dataset

We use the COVID-19 Fatcat Snapshot released by IA to get the PDFs for the CORD-19 dataset. It contains 45,294 full-text PDF files. These papers are a subset of the CORD-19 dataset with mappings of the unique ID indentifier for CORD-19;

Table 5.1: COVIDSeer dataset compared with the 2020-04-10 release of the CORD-19 corpus.

Data Type	CORD-19	COVIDSeer
Abstracts	42,352	**43,080**
Keyphrases	No	**Yes**
Paper Recommendation	No	**Yes**
Figures w/ captions	-	**70,102**
Tables w/ caption	-	**31,446**
Total Papers	51,078	51,078

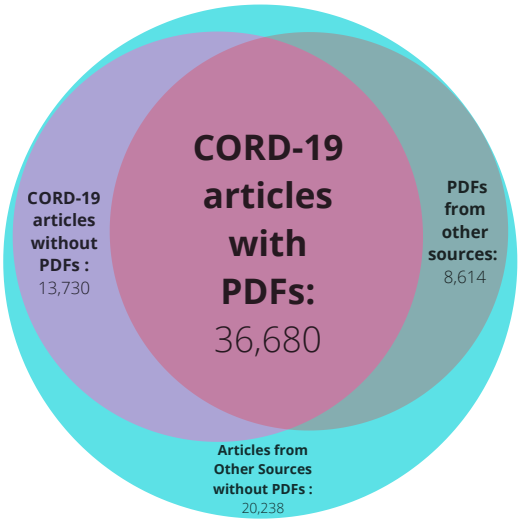


Figure 5.1: Overview of COVID-19 Fatcat Snapshot Dataset released by the Internet Archive. Other papers includes articles from sources other than Semantic Scholar

cord_uid to there own indentifiers. This makes it easier to map a corresponding PDF to it's CORD-19 metadata. All of these papers are open access and publicly available. More details about this dataset are shown in Figure 5.1. We see there are 36,680 PDFs which have a corresponding *cord_uid*. We maintain this unique identifier and its mappings across our dataset as well for easier adoption of other tools. IA also provides full-text and other metadata with the articles which we do not use. We very much rely on our extraction pipeline.

5.1.2 Information Extraction

Content extraction from scholarly PDFs allows authors to readily access particular entities, such as abstracts, results, tables, and/or figures. This allows easy access for someone who is not interested in all parts of the scholarly literature but rather just in particular sections related to their particular interests. We use PDFMEF [16], a multi-entity knowledge extraction framework for scholarly documents in a PDF format that extract multimodal information from papers. PDFMEF encapsulates state-of-the-art extraction tools for various types of content. It uses GROBID for document segmentation and metadata extraction and pdffigures for figure and table extraction. We can also configure PDFMEF to customize extracted content. Given the availability of full-text in the CORD-19 dataset, we focused on abstract, figure and table extraction. We illustrate this pipeline in Figure 5.2

Figure and Table Extraction - PDFMEF internally utilizes PDFFigures [18] to extract tables and figures. An independent evaluation suggests that it takes on average 0.2 seconds on each page and can achieve an F1-measure of 90% [16]. A compiled binary with set timeout of 20 seconds is called for execution. A timeout is set to avoid program freeze due to unusual file processing. The extracted figures and tables are collected in a directory with paper ID as prefix. From the 36,680 PDFs we have, we were able to successfully extract figures, tables, and there corresponding captions from only 30,274 PDFs. This created a total of 101,937 images (figures and tables) with captions.[1]

Abstract Extraction - GROBID extracts the bibliographical data corresponding to the header information (titles, authors and affiliation) with an accuracy of 77.79% per complete header instance. It takes a complete PDF as input to generate the TEI file in the output, and then the fields corresponding to a predefined and customized metadata schema are extracted into an XML file.

The dataset released on 04-10-2020 consisted of 51,045 research papers with 50,410 documents having an abstract. COVID-19 FATCAT Snapshot builds on top of that and adds more than 26,000 new abstracts from sources such as WHO, Wanfang corpus, CNKI corpus and FATCAT [2]. We only stick to the articles which have a *cord_uid* and belong to the CORD-19 corpus. IA has also released there

[1] In-depth view of the extracted data and cluster analysis : `https://github.com/jasonchhay/CovidSeer/blob/master/CovidSeer_supplementary.pdf`

[2] `https://fatcat.wiki/`

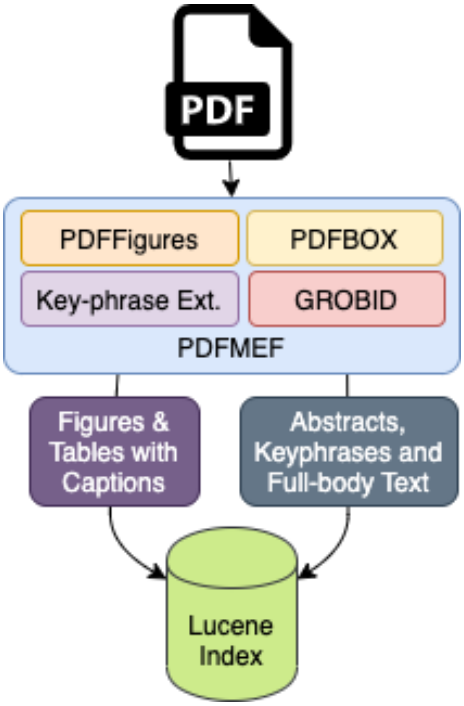


Figure 5.2: Extraction and indexing pipeline for COVIDSeer. Extracted text is indexed by Elasticsearch which uses Lucene.

version of metadata which we do not use for the data. We only use the PDFs with a corresponding *cord_uid*. AllenAI when building CORD-19 [14] used GROBID for abstract extraction. Our previous work shows that PDFBOX works better than GROBID [16] for text extraction [3] As a result of using PDFBox, we were able to extract more abstracts. PDFBOX was executed on 36,680 PDF documents and 728 missing abstracts were added to the collection.

Keyphrase extraction - Given the increasing breath of the COVID-19 literature, a high level topic description of a document could serve as a rich source of information. Therefore, a supervised keyphrase extraction model, also known as citation-enhanced keyphrase extraction (CeKE) model [196] built on a combination of novel features that captures information from citation contexts and existing features such as tf-idf,parts-of-speech tagging, and relative position of the text, was utilized for keyphrase extraction. There are two types of citation context. Citing contexts are texts around citations in the current article; cited contexts are texts around citations where the

[3]This study was based on GROBID 0.4 and PDFBox 1.8.6. However, for the latest releases no comparison exists.

current paper is cited in other articles. The latter requires full text of all articles in the citation graph, which is not available in the CORD-19 dataset. For the entries with full-text, we have extracted the keyphrases with the model using citing context. Otherwise, we use CeKE-TA that uses only the title and abstract. We extracted top-10 ranked keyphrases for each document. For the entries with full-text, keyphrases were extracted using CeKE-Citing. Otherwise CeKE-TA was used when only the title and abstract were available.

5.1.3 Search Engine Architecture

The information extracted is preprocessed using a standard Natural Language Processing pipeline similar to Covidex [197] and later indexed by Elasticsearch. By choosing Elasticsearch, we utilize a traditional monolithic ranking based architecture which is a combination of Boolean, TF/IDF, and the vector space models. ElasticSearch provides built-in analyzers which divides text into terms on word boundaries, based on Unicode Text Segmentation algorithm, and handles stemming, stopword removal, and tokenization internally. For text search, we use a combination of title, abstract, and available full-texts. These are indexed as separate fields into the Elasticsearch JSON documents. This breaking down the document in smaller atomic units ensures higher recall for the system [198].

The COVIDSeer website is deployed through the Django web framework because of its ease-of-use in serving web files, its Python back-end which easily supports Elasticsearch's Python library, and existing internal search engine frameworks that we built upon to remove development overhead. The user interface is developed through a hybrid of Django's template language and Vue.js. Vue is utilized to increase COVIDSeer's reactivity and browser load times. Pages are able to load much faster if the data is retrieved asynchronously through a REST API tied to the back-end instead of hanging the page while data is retrieved from Elasticsearch. Using our experiences from CiteSeerX [199], we wanted to see how well we could develop a similar search engine for COVID-19 research. With our rich experience in CiteSeerX [199], we were able to deploy the initial version COVIDSeer in 2 weeks. We also wanted to make other data available in a form that can be easily used by researchers. With this objective, we focused on two key areas: first, extracting more information from the dataset by means of abstract and keyphrase extraction and;

second, enhancing the users' search experience by means of faceted search and similar paper recommendations.

Faceted Search - Faceted search presents users with key-value metadata that can be used for query refinement [200]. Narrowing down traditional search results along multiple explicit dimensions allows the classification to be assessed in multiple ways. Therefore, we built and integrated a filtering mechanism for further accessing results of a query of interest. Available facets include authors, source, journal, publication year, presence/absence of full-text/abstracts. Sources are archival services or online repositories like bioRxiv, medRxiv, and others preprint services. This allows users to select filters from one or multiple categories and the intersection of all is presented in the search results.

Paper Recommendations - We also provide a list of top-10 similar papers for each paper in the dataset. We do this by first obtaining the top-10 candidate similar papers using tf-idf representation and cosine similarity of the abstracts and titles [74]. Second, we rerank these top-10 papers using SciBERT [201] embeddings of the abstracts and titles. To represent a paper, we average the contextual word-level embeddings for each abstract and title. Once we have these vectors we find the cosine similarity between the paper and the candidates returned by tf-idf similarity in the first step. It should be noted that our paper recommendation method is based on semantic similarity and recommended papers are within the CORD-19 corpus. A similar approach has been demonstrated in the work Citeomatic [202], where content-based recommendations were shown to give better results than citation graph-based approaches.

5.2 Discussion

COVIDSeer was launched online on March 30th, 2020. Since it was available online, we have received approximately 2,100 queries from 1,003 unique IP addresses. A systematic evaluation requires sufficient labeled and/or user-click data, which is is being collected. We target to study how keyphrases assist users to more efficiently navigate to desired papers. This work was done on a fixed snapshot of dataset released by CORD-19 and IA. With new weekly releases of CORD-19 dataset it is hard to keep our crawl repository and search engine updated, as crawling the actual PDFs for the articles will require sophisticated crawlers which we plan to build in the future. Nevertheless, we believe the enhancements and enrichment of the original dataset

proposed and demonstrated in this paper will be valuable to the research community.

5.3 Conclusion and Future Directions

Employing a amalgamation of information retrieval and machine learning techniques, we have enhanced the CORD-19 dataset through the addition of abstracts and principal keyphrases in the available literature. With the evolving nature of the CORD-19 dataset, we learned that building up a pipeline that updates the dataset used by the search engine is important. Furthermore, we use a 2 step model to recommend similar papers to a given paper with the aim of helping a researcher in his goal of finding relevant related work.

In addition to the information retrieval tasks, the proposed dataset can be coupled with machine learning to perform tasks such as Image Captioning, Table and Visual Question Answering, and Document Figure Classification.

Future work could be to implement author name disambiguation so as to correctly associate every author to their research paper. Another task would be creating a set of question templates for each of the found clusters and annotating figures and tables associated with them.

Chapter 6

ACL Anthology corpus

We present an up-to-date scholarly corpus from the ACL Anthology (ACL OCL), assisting open scientific research in the computational linguistics domain. Compared with previous ARC and AAN versions, ACL OCL includes structured full-texts with sections, references to figures, and links to a large knowledge resource (semantic scholar). ACL OCL contains 88k scientific papers, together with 210k figures extracted from the ACL Anthology up to September, 2022. To observe the developing trend of topics in the computational linguistics domain, we present topic classification and statistical analysis based on ACL OCL. Our dataset is open and available to download from huggingface[1].

6.1 Related Work

In this section, we introduce existing scholarly datasets. Scholarly datasets can be classified into two categories: task-specific and open research-aimed. The former[cite] mainly includes partial information from scientific papers (e.g., abstract, citation strings) and task-specific outputs (e.g., summaries, citation intent labels). The latter contains meta-information, full text of scientific papers, enabling researchers to further develop their task-specific data as well as analyze the characteristics of scientific papers or groups of them. The open research-aimed scholarly datasets serve as foundations for scientific document processing. Our work is in line with the open research-aimed dataset construction.

[1] `https://huggingface.co/datasets/ACL-OCL/acl-anthology-corpus`

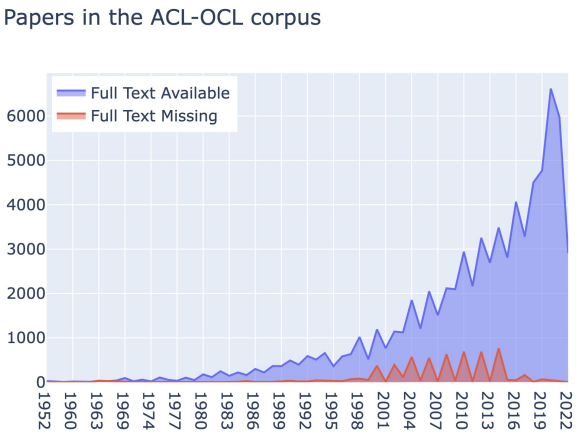


Figure 6.1: The growth in papers in the ACL Anthology over the years. We also see that the full-text extraction failed mostly between the years 2000 to 2015.

Data	#Doc	Full-text	Linked KG	Fig	Peer	Source	Domain
S2ORC	8.1M	structured	S2ORC$_{full}$	✓	partial	arXiv, ACL-anth, PMC	multi
unarXive	1.0M	string	MAG	×	partial	arXiv	multi
RefSeer	1.0M	string	CiteSeerX	×	partial	Word Wide Web	multi
CSL	396K	no	self	×	all	Chinese Core Journal	multi
ACL ARC	10.9K	string	self	×	all	ACL Anthology	CL
ACL AAN^2	25K	string	self	×	all	ACL Anthology	CL
ACL OCL	88k	structured	S2ORC$_{full}$	✓	all	ACL Anthology	CL

Table 6.1: Comparison between ACL OCL and existing full-text corpus. Structured full-text consists of information such as sections, paragraphs and etc. Peer means whether the scientific document is peer reviewed. ACL-anth is short for the ACL Anthology. CL means computational linguistics. Note that, S2ORC contains 42k papers from the ACL Anthology.

We compare our ACL OCL with existing open research-aimed corpora in Table 6.1. All corpora contain meta-information, which can be easily obtained from publishers, enabling analysis of citation/author networks. In contrast to corpora that develop their network with inside papers and thus limit the completeness of the network, the OCL is one of those linked to a large knowledge graph. Inspired by S2ORC, the OCL provides structured full texts, revealing the discourse structure (i.e., sections) of scientific documents. In addition, multi-modal features such as figures are extracted to enable research in document layout analysis or multi-modality. Most importantly, the ACL OCL target on the computational linguistic domain, which is in line with ACL ARC and ACL AAN. The up-to-date OCL corpus allows researchers to re-visit some domain-specific research questions such as hot topic detection or topic trend analysis starting from the most familiar CL domain.

6.2 Dataset Construction

Too detail focused – not enough on the important issues. Relegate engineering details to the appendix or website.

We start with crawling the PDFs and metadata from ACL Anthology. We then pass these PDFs through our full-text extraction process. We further enhance this data using the Semantic Scholar's API to fetch in-citations, out-citations and ids of other versions of the paper.

6.2.1 Data Acquisition

All the PDFs in ACL Anthology are open access and are available to crawl. So, we wrote custom crawlers to fetch all the PDF documents from ACL Anthology. We obey the 'robots.txt' of ACL Anthology and it allows everything on their webpage to be crawled. While we crawl all the PDFs, we also wanted to fetch the metadata from the website which will be more accurate than any PDF extraction methods. So, we crawled all the *.bib* files as well on the website. Overall we were able to fetch 80k documents including presentations, posters and conference proceedings some of which were more than 50 pages long. We get rid of some of these PDFs in our pre-processing step. (TODO mention the pre-processing steps somewhere)

6.2.2 Full-text extraction

For the full-text extraction from the PDFs we use GROBID. We also take inspiration from s2orc-doc2json[3] for our schema and enhance it further with missing information like affiliation and location information of the authors. s2orc-doc2json uses GROBID at its core but post-processes the extraction of GROBID into a more human comprehensible and familiar JSON schema. We are improving upon there post-processing step to include more information extracted by GROBID in our data.

6.2.3 Citation Network

To get the citation information for the papers in ACL we rely on Semantic Scholar's Open Academic Graph (S2AG) [203] [204] API. S2AG supports paper lookup via

[3] `https://github.com/allenai/s2orc-doc2json`

ACL ids. The metadata and other information provided helps us in fetching the citation information of the papers. This way we were able to get the citation counts, to and from citation links of all the ACL papers in our corpus. In total we have 669,650 connections between the 74k papers in . This opens up a whole new kind of analysis on this graph.

6.2.4 Linking with other corpus

TODO application of this S2AG API also helps us link ACL ids to the unique Semantic Scholar (S2) corpus ids. S2 clusters the different versions (e.g. arxiv, published version, corrected version) of a paper into one corpus paper id. This is great because now we have links to other versions of a paper in ACL including the arxiv versions which opens up more room for other analysis, for e.g. what changes are made to the arxiv version of a paper before it is published at an ACL venue etc. While the API provides the general metadata (abstract, title, author names, venue, year) it does not provide the full-text of the articles.

6.3 Dataset Analysis

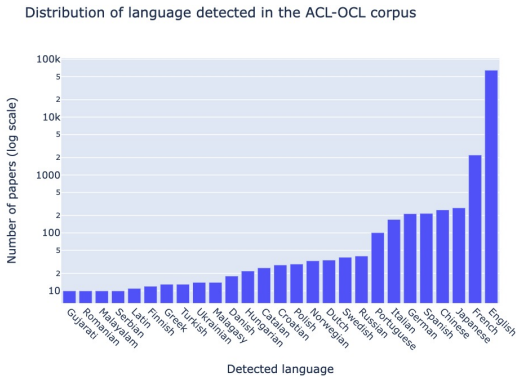


Figure 6.2: Language distribution in ACL-OCL

3. statistics: number of published papers per year, per main conference.

- schema

Citations	Title	Year	Predicted Topic
37,353	{BERT}: Pre-training of Deep Bidirectional Transformers for Language Understanding	2019	ML
23,467	{G}lo{V}e: Global Vectors for Word Representation	2014	LexSem
17,139	{B}leu: a Method for Automatic Evaluation of Machine Translation	2002	Summ
15,755	Learning Phrase Representations using {RNN} Encoder{–}Decoder for Statistical Machine Translation	2014	MT
14,182	{W}ord{N}et: A Lexical Database for {E}nglish	1992	LexSem
10,466	Convolutional Neural Networks for Sentence Classification	2014	ML
8,680	Thumbs up? Sentiment Classification using Machine Learning Techniques	2002	Sentiment
8,345	Deep Contextualized Word Representations	2018	LexSem
8,262	Building a Large Annotated Corpus of {E}nglish: The {P}enn {T}reebank	1993	Syntax
7,250	{ROUGE}: A Package for Automatic Evaluation of Summaries	2004	Summ

Table 6.2: Top-10 most cited papers in the ACL-OCL corpus of all time, with predicted topics from topic classification

In-degree	Title	Year
7,877	{BERT}: Pre-training of Deep Bidirectional Transformers for Language Understanding	2019
5,785	{B}leu: a Method for Automatic Evaluation of Machine Translation	2002
4,548	{G}lo{V}e: Global Vectors for Word Representation	2014
2,890	{M}oses: Open Source Toolkit for Statistical Machine Translation	2007
2,434	Deep Contextualized Word Representations	2018
2,348	Building a Large Annotated Corpus of {E}nglish: The {P}enn {T}reebank	1993
2,021	Neural Machine Translation of Rare Words with Subword Units	2016
1,947	A Systematic Comparison of Various Statistical Alignment Models	2003
1,702	Minimum Error Rate Training in Statistical Machine Translation	2003
1674	The Mathematics of Statistical Machine Translation: Parameter Estimation	1993

Table 6.3: Top-10 papers in the ACL-OCL network with the highest in-degree

6.3.1 Known issues with data quality

All the capabilities and limitation mentioned in [205] carry over to our dataset as well. The limitations of GROBID on PDFs can significantly impact the quality of the extracted text. Some of the common issues include incorrect author names and numbers, particularly when there are more than three tokens, as well as missing author emails and affiliations. The tool may also fail to identify cite spans and may incorrectly identify or miss sections altogether. Additionally, the tool may concatenate paragraphs, incorrectly associate footnote symbols with text, or include footnotes as text without proper section identification. Sometimes the extraction tool may also concatenate footnotes and tables or misidentify text as table captions, leading to missing tables or incorrect table formatting. As with other extractors, GROBID may also struggle with inline equation detection, sometimes converting equations into characters or failing to identify meaningless equations. Finally, the tool may incorrectly identify section names in appendices because the format for appendices are not standard.

6.4 Objective Topic Classification

As a widely used attribute for paper screening, topic information of scientific documents is very important. We specify topics to objective topics [206] which are used to denote research tasks (e.g., machine translation). However, topics of scientific documents are invisible on the ACL anthology website, although the authors provide the information when submitting their manuscripts for peer review. We investigate how to assign each CL paper with its most matching objective topic and provide a view of how can the topic information best benefit the CL community.

Given a scientific document d, with its textual information such as title, abstract and full-text, objective topic classification aims to assign a task topic label $l \in L$ to d. L is the topic label set taken from the submission topics (e.g., "Generation", "Question Answering") of the ACL conference[4]. The full topic label set is presented in Appendix 6.7.

Based on the amount of supervised information, we explore three classes of methods for topic classification, namely unsupervised methods, semi-supervised methods, and supervised methods.

6.4.1 NLI-based Un/Semi-supervised Methods

Without any scientific documents with ground-truth topic labels, an intuitive solution is to use zero-shot models for document classification. Yin et al. [207] fine-tuned BART [208] on natural language inference datasets, thus achieving zero-shot learning in many tasks including document classification. To identify the topic label of a document d, d and each candidate label l are taken as the premise and a hypothesis of an NLI task, and the fine-tuned BART-NLI model is asked to predict the probability $p(l|d)$ of l can be 'entailed' from d. The final topic label of d is the topic with the largest $p(l|d)$ value.

6.4.2 Keyword-based supervised Method

As salient information of documents, keywords are shown to be helpful for many tasks such as classification, clustering, summarization, and etc. Inspired by it, we design

[4]We remove four wide submission topics, including "Theme", "Student Research Workshop", "System Demonstrations", and "NLP Applications"

keyword-based supervised methods for topic classification. Keywords are extracted from each document of the training set, before selecting some topic-representative keywords for each topic. Given a test document d, the topic which contains the most topic-representative keywords in d is considered as its most matching topic. We explore different keyword extraction methods including TFIDF, Yake, which are simple and efficient unsupervised methods.

6.4.3 PLM-based Supervised Method

After being able to obtain over 2000 scientific documents with ground-truth topic labels, we explore the possibility to train a supervised model for topic classification. Given the success of pre-trained language models in NLP tasks with small-scale training data, we adopt a PLM-based classification framework. To be specific, the framework consists of a pre-trained language model for encoding the document, on top of which is a softmax classification layer for topic label prediction. In addition, we use pre-trained language models trained from scientific documents, to take advantage of their strong encoding power of domain-specific documents.

6.5 Experiments

6.5.1 Experimental settings

Data curation We crawl scientific papers of several online held CL conferences (e.g., ACL 2020, EMNLP 2020, ACL-IJCNLP 2021) between 2020 and 2022, together with their topics from their websites[5]. After aligning those papers with the data in the ACL OCL, we obtained 2545 documents in total. These documents together with their topics are used as our training or testing data respectively. Across all experiments, a 5-fold cross-validation was used, where we randomly select 2036 (80%) papers balanced in each topic as our training set, and the remaining 509 (20%) papers are used as the test set.

We use the standard precision, recall, and F1-measure as evaluation metrics.

[5]ACL 2020: https://virtual.acl2020.org/papers.html, EMNLP 2020: https://virtual.2020.emnlp.org/papers.html, ACL 2021: https://2021.aclweb.org/program/overview/

Source	#Doc	# Unique Topics
ACL 2020	705	21
EMNLP 2020	681	19
ACL 2021	470	21
EACL 2021	295	20
NAACL 2021	394	21
Total	2545	21

Table 6.4: Statistics of the topic classification corpus

Method	Mac. F1	Wei. F1	Acc.
BART-NLI-0shot	0.38	0.43	0.45
BART-NLI-tuned	0.44	0.50	0.51
TFIDF	0.51	0.54	0.53
Yake!	0.38	0.42	0.39
SciBERT	0.58	0.65	0.66
Specter	0.66	0.68	0.69

Table 6.5: Performances of different topic classification models

6.5.2 Topic Classification

We present the performances of different topic classification models in Table 6.6. We observe the following findings from Table 6.6: (1) both keyword- and PLM-based supervised methods significantly outperform unsupervised and semi-supervised methods, (2) the semi-supervised fine-tuned BART-NLI outperforms the zero-shot BART-NLI method. Both indicate the importance of using supervised data, even on a small scale. The best performance (0.68 in F1) achieved with the PLM-based supervised method is comparable to those reported by [56, 209] in similar topic classification tasks. While the performances are much lower than those (F1>0.8) in news domain [210], which indicates the challenges of NLP tasks in the scientific domain.

Method	Abstract	I+C	Diff.
SciBERT	0.66	0.67	0.01↑
SPECTER	0.64	0.68	0.04↑

Table 6.6: Performances (Accuracy) of different input texts, Abstract VS. I+C (Introduction+Conclusion).

We explore how different input text selection methods influence the task, namely Abstract and Introduction+Conclusion. Given the input length limit, we truncate input texts to keep 512 tokens. We adopt the I+C setting which has better performances.

6.5.3 Hot Topic Discovery

Figure 6.3: Plot of z-scores of each class, grouped by trend patterns. a) underrepresented historically, rapidly increasing b) declining interest before 2015, rapidly increasing afterwards c) relatively stable, with mild corrections in recent years d) peaked interest, rapidly declining in recent years

We provide a statistical analysis of objective topics in the CL domain in this section. We explore the trend of these topics from 2000 by analyzing their frequencies. Following previous work in social media analysis [211, 212], we use **bursty** to indicate that a topic is anomalously highly frequent over a time period. Here we adopt a metric z-score $z_{c,t}$ to evaluate the burstiness of a topic c in a year t, by measuring the difference between the real topic size and an estimated value. Refer to appendix 6.8 for calculation details of z-score.

Figure 6.3 presents the burstiness of all topics, which are grouped into four trend patterns.

6.6 Applications

1. As demonstrated in this paper, one potential application of the ACL anthology dataset is topic analysis of research themes. Using unsupervised learning techniques, it is possible to identify common research themes and topics across the papers in the dataset. Such an analysis can reveal dominant topics and keywords within the field of computer linguistics and can aid researchers in identifying new and emerging research directions as well.

2. Information extraction is another application of this corpus. By automatically extracting structured information from unstructured text data, it is possible to build domain-specific lexicons, thesauri, or knowledge graphs. For example, named entity recognition can be used to identify people, organizations, and locations mentioned in papers, which can then be used to build a knowledge graph of the field.

3. ACL OCL corpus can be used to train a free text generation model like BioGPT. Using such a model to generate text can aid researchers while writing research papers for a computational linguist venue and even brainstorm ideas if tuned properly. We are releasing a free text generation model trained on the ACL OCL corpus.[6].

4. By combining different types of data, such as figures, tables and captions, the ACL anthology dataset can be used in multi-modal research [213].

5. Building automated systems that can answer questions posed in natural language based on information in the ACL OCL corpus is another way this corpus can be used. Such a system can be used to answer questions related to the state-of-the-art in a particular subfield or about the findings of a particular research study [214].

6. An information retrieval system that allows users to search for relevant papers in the ACL anthology dataset based on keywords, topics, authors, or other criteria can be developed. This system can help researchers to find papers related to a particular research question or topic, and thus facilitate their research process.

[6] `https://huggingface.co/shaurya0512/distilgpt2-finetune-acl22`

A similar dataset but exclusive to COVID-19 related research, CORD-19 was used extensively to create search engines and discovery systems [215, 216].

7. Studies for linguistic disparity - Using natural language processing techniques to study linguistic disparities or biases in the ACL anthology dataset can reveal interesting patterns of language use or representation across different subfields or regions. Such studies can contribute to the field's awareness of linguistic disparities and biases, which may affect the diversity and inclusivity of the field. [217]

ID	**Abbr**	**Topic name**
1	CompSocial	Computational Social Science and Social Media
2	Dialogue	Dialogue and Interactive Systems
3	Discourse	Discourse and Pragmatics
4	Ethics	Ethics and NLP
5	NLG	Generation
6	IE	Information Extraction
7	IR	Information Retrieval and Text Mining
8	Interpret	Interpretability and Analysis of Models for NLP
9	VisRobo	Language Grounding to Vision, Robotics and Beyond
10	LingTheory	Linguistic Theories, Cognitive Modeling and Psycholinguistics
11	ML	Machine Learning for NLP
12	MT	Machine Translation and Multilinguality
13	WS	Phonology, Morphology and Word Segmentation
14	QA	Question Answering
15	Resource	Resources and Evaluation
16	LexSem	Semantics: Lexical Semantics
17	SenSem	Semantics: Sentence-level Semantics, Textual Inference
18	Sentiment	Sentiment Analysis, Stylistic Analysis, and Argument Mining
19	Speech	Speech and Multimodality
20	Summ	Summarization
21	Syntax	Syntax: Tagging, Chunking and Parsing

Table 6.7: Objective topics of papers in CL domain, together with defined abbreviations.

6.7 Topics in CL domain

We construct a taxonomy of objective topics in the computational linguistics domain, shown in Table 6.7, by taking and re-organizing the submission topics in major CL conferences (i.e., ACL, EMNLP, COLING). We surveyed their call for papers from 2000 until now, and include all topics in the taxonomy.

6.8 Hot Topic Discovery

We denote the number of published papers for a topic c in a year t to be $f_{c,t}$. Here we use a metric z-score $z_{c,t}$ to evaluate the burstiness of a topic c in a year t, by measuring the difference between the real topic size $f_{c,t}$ and an estimated value.

For a given topic c, and the time window t, the probability of f_c in t is modeled by the following Gaussian distribution,

$$P(f_{c,t}) \sim \mathcal{N}(N_t p_c, N_t p_c(1-p_c)), \tag{6.1}$$

where N_t is the number of papers published in t, and p_c is the expected probability of topic c in a random time window:

$$p_c = \frac{\sum_{t \in T} f_{c,t}}{\sum_{t \in T} N_t}, \tag{6.2}$$

where T is a long time period. z-score of topic c in time window t is calculated as follows,

$$z_{c,t} = \frac{f_{c,t} - E[e|d]}{\sigma[e|d]}. \tag{6.3}$$

Chapter 7

Open Domain Scientific Question Answering using GPT-4

In recent years, the rapid growth in scientific literature has made it increasingly challenging for researchers to stay updated and retrieve relevant information. To address this challenge, we propose a novel approach that combines the strengths of retrieval and grounded generation abilities of GPT-4. We present our open domain scientific question answering tool: S2QA. We also presents a preliminary study of GPT-4's capabilities in scientific question answering to generate accurate and informative answers. The proposed method comprises two primary components: (i) a retrieval-based mechanism that effectively extracts pertinent information from a large-scale scientific corpus, and (ii) a grounded generation process that utilizes GPT-4's capabilities to produce coherent and accurate answers based on the acquired information. We argue that the retrieval-based grounding not only improves the model's capacity to deliver precise answers, but also diminishes the likelihood of generating incorrect or hallucinated information. Additionally, we present an evaluation of our model on the PubmedQA task, demonstrating that grounded generation outperforms zero-shot capabilities of LLMs for scientific question answering.

7.1 Related Work

Scientific question answering using retrieval involves the development of methods to accurately extract answers from a vast corpus of scientific literature. BioAnswerFinder [218], uses a greedy iterative document retrieval technique, along with neural network-

based keyword selection and importance ranking to improve its baseline retrieval performance. This system evaluates seven retrieval approaches in total.Research has also been conducted on pre-training tasks and template-based question generation methods aimed at training neural retriever models in the biomedical domain [219]. Biomedical Question Answering (BQA) serves as a benchmark for natural language processing tasks, where models predict answers to given questions using resources like documents, images, knowledge bases, and question-answer pairs [220]. Scientific question answering using retrieval techniques employs various approaches, including semantic analysis, iterative document retrieval, and pre-training tasks. These systems are designed to accurately extract answers from vast bodies of scientific literature, ultimately aiding researchers in locating relevant information [221].

7.1.1 Pipeline

Talk about citations

7.2 Architecture of S2QA

Retrieval and re-ranking -
For retrieval of abstracts and title we are using the search endpoint of S2AG API [222]. We preprocess the query before sending a request. The ranking model of S2 search is a two stage model -

1. Candidate Selection: first stage consists of BM25 retrieval powered by Elastic-Search. This stage is the high recall step.

2. Re-ranking: These top-1000 documents from the above stage are re-ranked by a machine learning ranker. This stage is the high precision step.

   More details are here - `https://blog.allenai.org/building-a-better-search-engine-for-s`

Once we get the abstracts and title, we further re-rank them at our end using scientific embeddings. This makes sure that we have the more relevant papers ranked higher. We are using Specter 2 embeddings for this step. We retrieve the top-20 documents from the previous step and get the specter embeddings for each document and the query. We then use cosine similarity for each query-document pair and sort the documents.

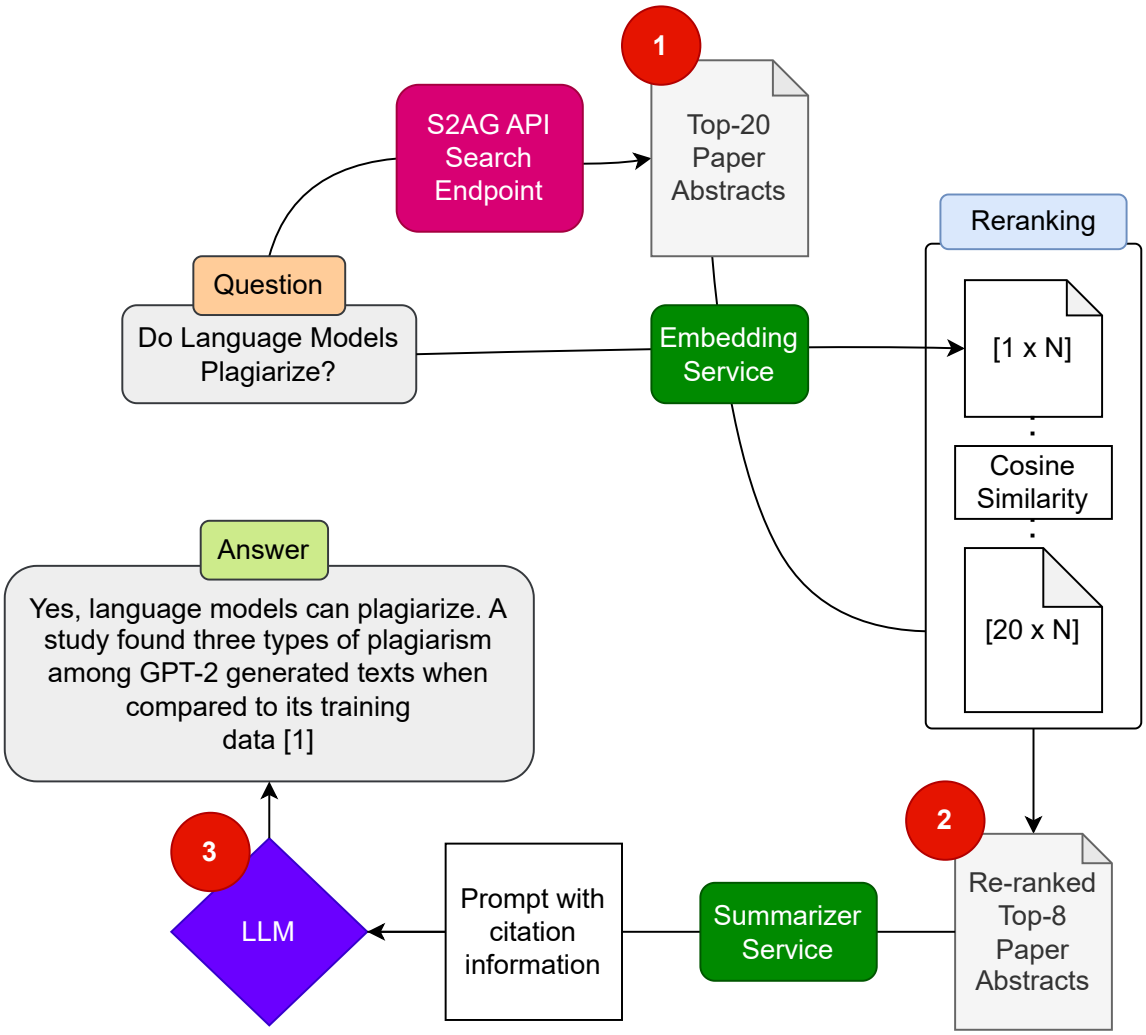


Figure 7.1: Architecture of S2QA. N is the dimension of the dense vector used to re-rank the documents.

If the abstract is too long(>256 tokens) we summarize the abstract to keep the system fast (more tokens more processing time) but we take a hit on our precision as we might miss out on information in the abstract.

7.2.1 Prompting

The top-8 abstracts are then turned into a prompt for the LLM, which in this case is GPT-4. We use a simple instruction template to prompt the LLM to answer:

```
[1] {title} and {abstract}
Source: {source_URL_1}
[2] {title} and {abstract}
Source:{source_URL_2}
. . .
[8] {title} and {abstract}
Source:{source_URL_8}

Instructions: Using the provided search results, write a comprehensive reply
to the given query. If you find a result relevant make sure to cite the result
using [[number](URL)] notation after the reference.

Query: {Question}
```

We must also ensure that the LLM is primed in such a way that it refrains from providing its own opinions and adheres to the instructions consistently throughout all responses. The following prompt defines the role of the LLM:

```
You are a helpful assistant to a researcher. You are given a prompt and a
list of references. You are asked to write a summary of the references if
they are related to the question. You should not include any personal
opinions or interpretations in your answer, but rather focus on objectively
presenting the information from the search results.
```

7.3 Experiments

We do a small scale study of our system as GPT-4 API is expensive, slow and not very reliable in terms of robustness (frequent 502 errors). In evaluating the performance of our system, we utilize the well-established PubMedQA [223] benchmark dataset as a means of assessment. The PubMedQA dataset requires the system to answer questions based on a given context with a simple "yes", "no" or "maybe". While the majority of prior research has focused on fine-tuning Language Model's (LLMs) capabilities using the PubMed collection, our approach diverges by employing an unsupervised method that leverages GPT-4. This technique involves prompting GPT-4 with context derived from retrieval. A comparison of accuracies is done for :

Table 7.1: Zero-shot classification accuracy comparison of our system with existing models on the PubmedQA-L test set. Please note that our system is not fine-tuned nor it is using the ground truth context in the PubMedQA dataset. For augmenting context we are using the S2AG API for retrieval of relevant documents.

Model	Context	Setting	**Accuracy %**
BioGPT [225]	GT	Fine-tuning	**78.2**
GPT-4 [224]	GT	Zero Shot	77.4
Flan-PaLM 8B [226]	GT	Three Shot	67.6
BioELECTRA uncased [227]	GT	Fine-tuning	64.02
GPT-4 + retrieval (Ours)	Retrieval	Zero Shot	62.0
PubMedBERT [228]	GT	Fine-tuning	55.8
PaLM 8B [226]	GT	Fine-tuning	34.0
GPT-4	None	Zero Shot	20.0

1. GPT-4 − GT: Asking the questions directly to GPT-4 without any ground truth(GT). This serves as a baseline. Here is the prompt for this task -

```
Please answer the following question with just yes/no/maybe.
Question: {pubmed_qa_question}
```

2. GPT-4 − GT + retrieval (ours): Replacing ground truth context with retrieval augmentation (grounded generation), which is indicative of the real-world scenario for our system.

3. GPT-4 + GT: This is a Zero-shot classification task, asking GPT-4 questions with the ground truth context [224] (this serves as a best case scenario for GPT-4).

We benchmarked our tool using the test set from the PubmedQA-Labeled dataset, which consists of 500 questions and their relevant context/abstracts. Our system outperformed the previous best system, which was specifically pre-trained and fine-tuned with Pubmed and related collections.

We argue that these capabilities will likely transfer to other domains as well. However, due to the lack of benchmark datasets to report numbers on, we cannot conduct a study for other domains at this time.

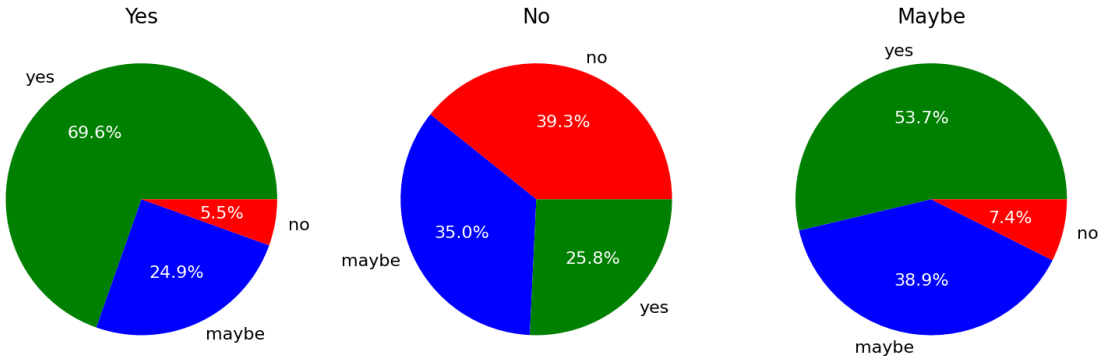


Figure 7.2: Distribution of the predicted answers for PubMedQA for each class - yes, no and maybe

7.3.1 Analysis of PubMedQA predictions

In the Figure 7.2 we see that answers from GPT-4 are more aligned towards **Yes** if the ground truth answer is **Maybe**. Our findings suggest that the GPT-4 model exhibits an inclination towards affirmative responses, particularly when the ground truth answer is equivocal in nature("maybe"). This inclination towards positive responses may potentially be attributed to the underlying nature of the language models that form the basis of the GPT-4 architecture. Further research is required to explore the mechanisms that contribute to this observed bias in the model's responses.

As we can see in the Table 7.1, GPT-4 without any context performs poorly in zero-shot classification. If GPT-4 doesn't have context it overwhelmingly says maybe. This explains the poor performance.

It is important to remember that our system is unsupervised question answering and we are comparing it with systems which have been fine-tuned for the PubMedQA benchmark. While BioBERT/BioGPT can only answer questions for Biology and related fields, our system can answer questions from any field of study.

7.4 Conclusion and future work

this paper presents a novel and effective approach for scientific QA utilizing GPT-4 and retrieval-based grounded generation. Our findings not only contribute to the understanding of GPT-4's potential in scientific QA but also highlight the importance of grounding the generation process for improved accuracy and reliability. We believe

that our work serves as a solid foundation for future research aiming to harness the power of large-scale language models for scientific information retrieval and knowledge discovery. Limitations - This is a real world system where a lot of abstracts might be missing from the retriever for examples. Improvements to the search and ranking capabilities of the S2AG search API will greatly improve the quality of the answers.

- GPT-4 predicts maybe for most questions because it is safer to say maybe than a yes or no. But given just the option to say yes or no it might change its answers, this is something we have not explored in this work.

- We believe that these capabilities of our system go beyond just bio-logy data and it can perform effectively in other domains as well.

- It should be noted that all our answers are zero shot and no training data is used here to be fair to all fields of study

- Scientific search as a ChatGPT plugin.

Bibliography

[1] Stalnaker, D. and R. Zanibbi (2015) "Math expression retrieval using an inverted index over symbol pairs," in *DRR*.

[2] Lyons, I. M. and S. L. Beilock (2011) "Mathematics anxiety: separating the math from the anxiety," *Cerebral cortex*, **22**(9), pp. 2102–2110.

[3] Zanibbi, R. and A. Orakwue (2015) "Math search for the masses: Multimodal search interfaces and appearance-based retrieval," in *Conferences on Intelligent Computer Mathematics*, Springer, pp. 18–36.

[4] Caragea, C., J. Wu, A. Ciobanu, K. Williams, J. Fernández-Ramírez, H.-H. Chen, Z. Wu, and L. Giles (2014) "Citeseer x: A scholarly big dataset," in *European Conference on Information Retrieval*, Springer, pp. 311–322.

[5] Schubotz, M., A. Youssef, V. Markl, and H. S. Cohl (2015) "Challenges of mathematical information retrievalin the ntcir-11 math wikipedia task," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, ACM, pp. 951–954.

[6] Wang, Y., L. Gao, S. Wang, Z. Tang, X. Liu, and K. Yuan (2015) "WikiMirs 3.0: a hybrid MIR system based on the context, structure and importance of formulae in a document," in *Proceedings of the 15th ACM/IEEE-CS joint conference on digital libraries*, ACM, pp. 173–182.

[7] Council, N. R. et al. (2014) *Developing a 21st century global library for mathematics research*, National Academies Press.

[8] Bird, S., R. Dale, B. Dorr, B. Gibson, M. Joseph, M.-Y. Kan, D. Lee, B. Powley, D. Radev, and Y. F. Tan (2008) "The ACL Anthology Reference Corpus: A Reference Dataset for Bibliographic Research in Computational Linguistics," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, European Language Resources Association (ELRA), Marrakech, Morocco.
URL `http://www.lrec-conf.org/proceedings/lrec2008/pdf/445_paper.pdf`

[9] Radev, D. R., P. Muthukrishnan, V. Qazvinian, and A. Abu-Jbara (2013) "The ACL anthology network corpus," *Language Resources and Evaluation*, pp. 1–26.
URL http://dx.doi.org/10.1007/s10579-012-9211-2

[10] Hall, D., D. Jurafsky, and C. D. Manning (2008) "Studying the History of Ideas Using Topic Models," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Honolulu, Hawaii, pp. 363–371.
URL https://aclanthology.org/D08-1038

[11] Gollapalli, S. D. and X. Li (2015) "EMNLP versus ACL: Analyzing NLP research over time," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Lisbon, Portugal, pp. 2002–2006.
URL https://aclanthology.org/D15-1235

[12] McKeown, K., H. Daume III, S. Chaturvedi, J. Paparrizos, K. Thadani, P. Barrio, O. Biran, S. Bothe, M. Collins, K. R. Fleischmann, L. Gravano, R. Jha, B. King, K. McInerney, T. Moon, A. Neelakantan, D. O'Seaghdha, D. Radev, C. Templeton, and S. Teufel (2016) "Predicting the impact of scientific concepts using full-text features," *Journal of the Association for Information Science and Technology*, **67**(11), pp. 2684–2696, https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.23612.
URL https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.23612

[13] Guidotti, E. and D. Ardia (2020) "COVID-19 data hub," .

[14] Wang, L. L., K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. M. Kinney, Z. Liu, W. Merrill, P. Mooney, D. A. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. M. Raymond, D. S. Weld, O. Etzioni, and S. Kohlmeier (2020) "CORD-19: The Covid-19 Open Research Dataset," *ArXiv*, **abs/2004.10706**.

[15] Teregowda, P. B., I. G. Councill, J. P. F. Ramírez, M. Khabsa, S. Zheng, and C. L. Giles (2010) "SeerSuite: Developing a Scalable and Reliable Application Framework for Building Digital Libraries by Crawling the Web." *WebApps*, **10**, pp. 14–14.

[16] Wu, J., J. Killian, H. Yang, K. Williams, S. R. Choudhury, S. Tuarob, C. Caragea, and C. L. Giles (2015) "PDFMEF: A Multi-Entity Knowl-

edge Extraction Framework for Scholarly Documents and Semantic Search," Association for Computing Machinery, New York, NY, USA.

[17] Lopez, P. (2009) "GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications," in *ECDL*.

[18] Clark, C. and S. Divvala (2015) "Looking Beyond Text: Extracting Figures, Tables, and Captions from Computer Science Paper," .

[19] Chen, H.-H., J. Wu, and C. L. Giles (2017) "Compiling Keyphrase Candidates for Scientific Literature Based on Wikipedia." .

[20] Zhao, W. X., K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, and J. rong Wen (2023) "A Survey of Large Language Models," *ArXiv*, **abs/2303.18223**.

[21] Peng, B., M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Lidén, Z. Yu, W. Chen, and J. Gao (2023) "Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback," *ArXiv*, **abs/2302.12813**.

[22] de Almeida, V. P., S. Bhowmik, G. F. Lima, M. Endler, and K. Rothermel (2020) "DSCEP: An Infrastructure for Decentralized Semantic Complex Event Processing," in *IEEE International Conference on Big Data, Big Data 2020, Atlanta, GA, USA, December 10-13, 2020* (X. Wu, C. Jermaine, L. Xiong, X. Hu, O. Kotevska, S. Lu, W. Xu, S. Aluru, C. Zhai, E. Al-Masri, Z. Chen, and J. Saltz, eds.), IEEE, pp. 391–398.
URL `https://doi.org/10.1109/BigData50022.2020.9377877`

[23] de Sompel, H. V., M. L. Nelson, C. Lagoze, and S. Warner (2004) "Resource Harvesting within the OAI-PMH Framework," *D Lib Mag.*, **10**(12).
URL `https://doi.org/10.1045/december2004-vandesompel`

[24] Giles, C. L., K. D. Bollacker, and S. Lawrence (1998) "CiteSeer: An Automatic Citation Indexing System," in *Proceedings of the 3rd ACM International Conference on Digital Libraries*, pp. 89–98.

[25] Vine, R. (2006) "Google Scholar," *Journal of the Medical Library Association*, **94**(1), pp. 97–99.
URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1324783/`

[26] Sinha, A., Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. P. Hsu, and K. Wang (2015) "An Overview of Microsoft Academic Service (MAS) and Applications," in *Proceedings of the 24th International Conference on World Wide Web*, WWW

'15 Companion, Republic and Canton of Geneva, Switzerland, pp. 243–246.
URL http://dx.doi.org/10.1145/2740908.2742839

[27] Wang, K., Z. Shen, C. Huang, C.-H. Wu, D. Eide, Y. Dong, J. Qian, A. Kanakia, A. Chen, and R. Rogahn (2019) "A Review of Microsoft Academic Services for Science of Science Studies," *Frontiers in Big Data*, **2**, p. 45.
URL https://www.frontiersin.org/article/10.3389/fdata.2019.00045

[28] Erera, S., M. Shmueli-Scheuer, G. Feigenblat, O. P. Nakash, O. Boni, H. Roitman, D. Cohen, B. Weiner, Y. Mass, O. Rivlin, G. Lev, A. Jerbi, J. Herzig, Y. Hou, C. Jochim, M. Gleize, F. Bonin, and D. Konopnicki (2019) "A Summarization System for Scientific Documents," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019 - System Demonstrations* (S. Padó and R. Huang, eds.), Association for Computational Linguistics, pp. 211–216.
URL https://doi.org/10.18653/v1/D19-3036

[29] Ginsparg, P. (2011) "ArXiv at 20," *Nature*, **476**(7359), pp. 145–147.
URL https://doi.org/10.1038/476145a

[30] Goeckenjan, G., H. Sitter, M. Thomas, D. Branscheid, M. Flentje, F. Griesinger, N. Niederle, M. Stuschke, T. Blum, K. Deppermann, et al. (2011) "PubMed Results," *Pneumologie*, **65**(8), pp. e51–e75.

[31] Manca, S. (2018) "ResearchGate and Academia.edu as networked sociotechnical systems for scholarly communication: A literature review." *Research in Learning Technology*, **26**.

[32] Tang, J., J. Zhang, L. Yao, J. Li, L. Zhang, and Z. 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* (Y. Li, B. Liu, and S. Sarawagi, eds.), ACM, pp. 990–998.
URL https://doi.org/10.1145/1401890.1402008

[33] Falagas, M. E., E. I. Pitsouni, G. A. Malietzis, and G. Pappas (2008) "Comparison of PubMed, Scopus, Web of Science, and Google Scholar: strengths and weaknesses," *The FASEB Journal*, **22**(2), pp. 338–342, https://faseb.onlinelibrary.wiley.com/doi/pdf/10.1096/fj.07-9492LSF.
URL https://faseb.onlinelibrary.wiley.com/doi/abs/10.1096/fj.07-9492LSF

[34] Entlich, R., J. Olsen, L. Garson, M. Lesk, L. Normore, and S. Weibel (1997) "Making a Digital Library: The Contents of the CORE Project," *ACM Trans. Inf. Syst.*, **15**(2), p. 103–123.
URL https://doi-org.proxy.lib.odu.edu/10.1145/248625.248627

[35] Lim, E.-P. and Y. Lu (1999) "Harp: A Distributed Query System for Legacy Public Libraries and Structured Databases," *ACM Trans. Inf. Syst.*, **17**(3), p. 294–319.
URL https://doi.org/10.1145/314516.314521

[36] Lagoze, C., W. Arms, S. Gan, D. Hillmann, C. Ingram, D. Krafft, R. Marisa, J. Phipps, J. Saylor, C. Terrizzi, W. Hoehn, D. Millman, J. Allan, S. Guzman-Lara, and T. Kalt (2002) "Core Services in the Architecture of the National Science Digital Library (NSDL)," in *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '02, Association for Computing Machinery, New York, NY, USA, p. 201–209.
URL https://doi.org/10.1145/544220.544264

[37] Teregowda, P. B., B. Urgaonkar, and C. L. Giles (2010) "CiteSeerx: a cloud perspective," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, HotCloud'10, Berkeley, CA, USA, pp. 9–9.
URL http://dl.acm.org/citation.cfm?id=1863103.1863112

[38] Wu, J., P. Teregowda, K. Williams, M. Khabsa, D. Jordan, E. Treece, Z. Wu, and C. L. Giles (2014) "Migrating a Digital Library to a Private Cloud," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pp. 97–106.

[39] Wu, Z., J. Wu, M. Khabsa, K. Williams, H. Chen, W. Huang, S. Tuarob, S. R. Choudhury, A. Ororbia, P. Mitra, and C. L. 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*, pp. 117–126.
URL https://doi.org/10.1109/JCDL.2014.6970157

[40] Tang, J., J. Zhang, L. Yao, J. Li, L. Zhang, and Z. 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*, KDD '08, Association for Computing Machinery, New York, NY, USA, p. 990–998.
URL https://doi.org/10.1145/1401890.1402008

[41] Xia, F., W. Wang, T. M. Bekele, and H. Liu (2017) "Big Scholarly Data: A Survey," *IEEE Transactions on Big Data*, **3**(1), pp. 18–35.

[42] Wu, J., K. Kim, and C. L. Giles (2019) "CiteSeerX: 20 years of service to scholarly big data," in *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse, AIDR 2019, Pittsburgh, PA, USA, May 13-15, 2019* (H. Wang and K. Webster, eds.), ACM, pp. 1:1–1:4.
URL https://doi.org/10.1145/3359115.3359119

[43] Caragea, C., J. Wu, S. D. Gollapalli, and C. L. 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.*, pp. 3997–4002.
URL http://www.aaai.org/ocs/index.php/IAAI/IAAI16/paper/view/12343

[44] Han, H., C. L. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. A. Fox (2003) "Automatic document metadata extraction using support vector machines," in *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '03, pp. 37–48.
URL http://dl.acm.org/citation.cfm?id=827140.827146

[45] Williams, K. and C. L. Giles (2013) "Near Duplicate Detection in an Academic Digital Library," in *Proceedings of the 2013 ACM Symposium on Document Engineering*, DocEng '13, ACM, New York, NY, USA, pp. 91–94.
URL http://doi.acm.org/10.1145/2494266.2494312

[46] Treeratpituk, P. and C. L. Giles (2009) "Disambiguating authors in academic publications using random forests," in *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '09, ACM, New York, NY, USA, pp. 39–48.
URL http://doi.acm.org/10.1145/1555400.1555408

[47] Florescu, C. and C. Caragea (2017) "PositionRank: An Unsupervised Approach to Keyphrase Extraction from Scholarly Documents," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pp. 1105–1115.
URL https://doi.org/10.18653/v1/P17-1102

[48] Huang, W., Z. Wu, C. Liang, P. Mitra, and C. Giles (2015) "A Neural Probabilistic Model for Context Based Citation Recommendation," *Proceedings of the AAAI Conference on Artificial Intelligence*, **29**(1).
URL https://ojs.aaai.org/index.php/AAAI/article/view/9528

[49] Chen, H.-H., L. Gou, X. Zhang, and C. L. Giles (2011) "CollabSeer: A Search Engine for Collaboration Discovery," in *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, JCDL '11,

Association for Computing Machinery, New York, NY, USA, p. 231–240.
URL https://doi.org/10.1145/1998076.1998121

[50] Chen, H., Y. Yang, W. Lu, and J. Chen (2020) "Exploring multiple diversification strategies for academic citation contexts recommendation," *Electron. Libr.*, **38**(4), pp. 821–842.
URL https://doi.org/10.1108/EL-02-2020-0046

[51] Khabsa, M. and C. L. Giles (2014) "The number of scholarly documents on the public web," *PLoS ONE*, **9**(5), p. e93949.

[52] Heisey, S. (2020), "SolrPerformanceProblems," https://cwiki.apache.org/confluence/display/SOLR/SolrPerformanceProblems.
URL https://cwiki.apache.org/confluence/display/SOLR/SolrPerformanceProblems

[53] Wu, J., B. Kandimalla, S. Rohatgi, A. Sefid, J. Mao, and C. L. Giles (2018) "CiteSeerX-2018: A Cleansed Multidisciplinary Scholarly Big Dataset," in *IEEE International Conference on Big Data, Big Data 2018, Seattle, WA, USA, December 10-13, 2018*, pp. 5465–5467.
URL https://doi.org/10.1109/BigData.2018.8622114

[54] Kandimalla, B., S. Rohatgi, J. Wu, and C. L. Giles (2021) "Large Scale Subject Category Classification of Scholarly Papers With Deep Attentive Neural Networks," *Frontiers in Research Metrics and Analytics*, **5**, p. 31.
URL https://www.frontiersin.org/article/10.3389/frma.2020.600382

[55] Wu, J., P. Teregowda, M. Khabsa, S. Carman, D. Jordan, J. San Pedro Wandelmer, X. Lu, P. Mitra, and C. L. Giles (2012) "Web Crawler Middleware for Search Engine Digital Libraries: A Case Study for CiteSeerX," in *Proceedings of the Twelfth International Workshop on Web Information and Data Management*, WIDM '12, ACM, New York, NY, USA, pp. 57–64.
URL http://doi.acm.org/10.1145/2389936.2389949

[56] Lo, K., L. L. Wang, M. Neumann, R. Kinney, and D. Weld (2020) "S2ORC: The Semantic Scholar Open Research Corpus," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, pp. 4969–4983.
URL https://www.aclweb.org/anthology/2020.acl-main.447

[57] Wu, J., J. Killian, H. Yang, K. Williams, S. R. Choudhury, S. Tuarob, C. Caragea, and C. L. 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*, K-CAP

2015, Association for Computing Machinery, New York, NY, USA.
URL https://doi.org/10.1145/2815833.2815834

[58] Clark, C. and S. K. 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*, pp. 143–152.
URL http://doi.acm.org/10.1145/2910896.2910904

[59] Lopez, P. (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*, ECDL'09, Springer-Verlag, Berlin, Heidelberg, pp. 473–474.
URL http://dl.acm.org/citation.cfm?id=1812799.1812875

[60] Wu, J., K. Williams, H. Chen, M. Khabsa, C. Caragea, A. Ororbia, D. Jordan, and C. L. Giles (2014) "CiteSeerX: AI in a Digital Library Search Engine," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 2930–2937.

[61] Manku, G. S., A. Jain, and A. Das Sarma (2007) "Detecting Near-Duplicates for Web Crawling," in *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, Association for Computing Machinery, New York, NY, USA, p. 141–150.
URL https://doi.org/10.1145/1242572.1242592

[62] Andoni, A. and P. Indyk (2008) "Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions," *Commun. ACM*, **51**(1), p. 117–122.
URL https://doi.org/10.1145/1327452.1327494

[63] Bloom, B. H. (1970) "Space/Time Trade-Offs in Hash Coding with Allowable Errors," *Commun. ACM*, **13**(7), p. 422–426.
URL https://doi.org/10.1145/362686.362692

[64] Singh, A., M. D'Arcy, A. Cohan, D. Downey, and S. Feldman (2022) "SciRepEval: A Multi-Format Benchmark for Scientific Document Representations," *ArXiv*, **abs/2211.13308**.

[65] Guo, J., Y. Fan, Q. Ai, and W. B. Croft (2016) "A Deep Relevance Matching Model for Ad-hoc Retrieval," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ACM.

[66] Xiong, C., Z. Dai, J. Callan, Z. Liu, and R. Power (2017) "End-to-End Neural Ad-hoc Ranking with Kernel Pooling," *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[67] Dai, Z., C. Xiong, J. Callan, and Z. Liu (2018) "Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search," *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

[68] Mitra, B., F. Diaz, and N. Craswell (2017) "Learning to match using local and distributed representations of text for web search," in *Proceedings of the 26th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, pp. 1291–1299.

[69] Mitra, B. and N. Craswell (2019) "An Updated Duet Model for Passage Re-ranking," *ArXiv*, **abs/1903.07666**.

[70] Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018) "Deep Contextualized Word Representations," in *North American Chapter of the Association for Computational Linguistics*.

[71] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019) "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *ArXiv*, **abs/1810.04805**.

[72] Zhao, L. (2012) "Modeling and solving term mismatch for full-text retrieval," *SIGIR Forum*, **46**, pp. 117–118.

[73] Mitra, B., N. Craswell, et al. (2018) "An introduction to neural information retrieval," *Foundations and Trends® in Information Retrieval*, **13**(1), pp. 1–126.

[74] Nogueira, R. and K. Cho (2019) "Passage Re-ranking with BERT," *arXiv preprint arXiv:1901.04085*.

[75] Dai, Z. and J. Callan (2019) "Deeper Text Understanding for IR with Contextual Neural Language Modeling," *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[76] Yilmaz, Z. A., W. Yang, H. Zhang, and J. J. Lin (2019) "Cross-Domain Modeling of Sentence-Level Evidence for Document Retrieval," in *Conference on Empirical Methods in Natural Language Processing*.

[77] MacAvaney, S., A. Yates, A. Cohan, and N. Goharian (2019) "CEDR: Contextualized Embeddings for Document Ranking," *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[78] Hofstätter, S. and A. Hanbury (2019) "Let's measure run time! Extending the IR replicability infrastructure to include performance aspects," in *OSIRRC@SIGIR*.

[79] Kohavi, R., A. Deng, B. Frasca, T. Walker, Y. Xu, and N. Pohlmann (2013) "Online controlled experiments at large scale," *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*.

[80] Nogueira, R., W. Yang, J. J. Lin, and K. Cho (2019) "Document Expansion by Query Prediction," *ArXiv*, **abs/1904.08375**.

[81] Dai, Z. and J. Callan (2019) "Context-Aware Sentence/Passage Term Importance Estimation For First Stage Retrieval," *ArXiv*, **abs/1910.10687**.

[82] Jayawardana, Y., A. C. Nwala, G. Jayawardena, J. Wu, S. Jayarathna, M. L. Nelson, and C. Lee Giles (2020) "Modeling Updates of Scholarly Webpages Using Archived Data," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 1868–1877.

[83] Mi^{2}utka, J. (2008) "Indexing mathematical content using full text search engine," .

[84] Pattaniyil, N. and R. Zanibbi (2014) "Combining TF-IDF Text Retrieval with an Inverted Index over Symbol Pairs in Math Expressions: The Tangent Math Search Engine at NTCIR 2014." in *NTCIR*.

[85] Matthewman, S. (2011) *Technology and social theory*, Macmillan International Higher Education.

[86] Aycock, A. (1995) ""Technologies of the self:" Foucault and Internet discourse," *Journal of Computer-Mediated Communication*, **1**(2), p. JCMC121.

[87] Buldt, B., B. Löwe, and T. Müller (2008) "Towards a new epistemology of mathematics," *Erkenntnis*, **68**(3), pp. 309–329.

[88] Rieger, O. Y. (2009) "Search engine use behavior of students and faculty: User perceptions and implications for future research," *First Monday*, **14**(12).

[89] Cranshaw, J. and A. Kittur (2011) "The polymath project: lessons from a successful online collaboration in mathematics," in *Proceedings of the SIGCHI conference on human factors in computing systems*, ACM, pp. 1865–1874.

[90] Kohlhase, A. (2014) "Math Web Search Interfaces and the Generation Gap of Mathematicians," in *International Congress on Mathematical Software*, Springer, pp. 586–593.

[91] Gowers, T. and M. Nielsen (2009) "Massively collaborative mathematics," *Nature*, **461**(7266), p. 879.

[92] Smithies, S., K. Novins, and J. Arvo "A handwriting-based equation editor," .

[93] Pollanen, M., T. Wisniewski, and X. Yu "Xpress: a novice interface for the real-time communication of mathematical expressions," Citeseer.

[94] Zhao, J., M.-Y. Kan, and Y. L. Theng (2008) "Math information retrieval: user requirements and prototype implementation," in *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, ACM, pp. 187–196.

[95] Pineau, D. C. (2016) "Math-Aware Search Engines: Physics Applications and Overview," *arXiv preprint arXiv:1609.03457*.

[96] Kohlhase, A. (2014) "Search interfaces for mathematicians," in *Intelligent Computer Mathematics*, Springer, pp. 153–168.

[97] Zanibbi, R., K. Davila, A. Kane, and F. W. Tompa (2016) "Multi-stage math formula search: Using appearance-based similarity metrics at scale," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ACM, pp. 145–154.

[98] Zanibbi, R. and D. Blostein (2012) "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition (IJDAR)*, **15**(4), pp. 331–357.

[99] Sasarak, C., K. Hart, R. Pospesel, D. Stalnaker, L. Hu, R. Livolsi, S. Zhu, and R. Zanibbi "min: A multimodal web interface for math search," .

[100] Chan, K.-F. and D.-Y. Yeung (2000) "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, **3**(1), pp. 3–15.

[101] Zanibbi, R., K. Novins, J. Arvo, and K. Zanibbi "Aiding manipulation of handwritten mathematical expressions through style-preserving morphs," .

[102] Hu, L. and R. Zanibbi (2011) "HMM-based recognition of online handwritten mathematical symbols using segmental k-means initialization and a modified pen-up/down feature," in *2011 International Conference on Document Analysis and Recognition*, IEEE, pp. 457–462.

[103] Hu, L., K. Hart, R. Pospesel, and R. Zanibbi (2012) "Baseline extraction-driven parsing of handwritten mathematical expressions," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, IEEE, pp. 326–330.

[104] Zhang, J., J. Du, and L. Dai (2018) "Multi-scale attention with dense encoder for handwritten mathematical expression recognition," in *2018 24th International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 2245–2250.

[105] Zhang, J., J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai (2017) "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognition*, **71**, pp. 196–206.

[106] Mitra, B. and N. Craswell (2015) "Query auto-completion for rare prefixes," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, ACM, pp. 1755–1758.

[107] Zhang, Y., M. M. Rahman, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara, A. Angert, E. Banner, V. Khetan, et al. (2016) "Neural information retrieval: A literature review," *arXiv preprint arXiv:1611.06792*.

[108] Ganguly, D., D. Roy, M. Mitra, and G. J. Jones (2015) "Word embedding based generalized language model for information retrieval," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, ACM, pp. 795–798.

[109] Ye, X., H. Shen, X. Ma, R. Bunescu, and C. Liu (2016) "From word embeddings to document similarities for improved information retrieval in software engineering," in *Proceedings of the 38th international conference on software engineering*, ACM, pp. 404–415.

[110] Nalisnick, E., B. Mitra, N. Craswell, and R. Caruana (2016) "Improving document ranking with dual word embeddings," in *Proceedings of the 25th International Conference Companion on World Wide Web*, International World Wide Web Conferences Steering Committee, pp. 83–84.

[111] Kuzi, S., A. Shtok, and O. Kurland (2016) "Query expansion using word embeddings," in *Proceedings of the 25th ACM international on conference on information and knowledge management*, ACM, pp. 1929–1932.

[112] Mitra, B., E. Nalisnick, N. Craswell, and R. Caruana (2016) "A dual embedding space model for document ranking," *arXiv preprint arXiv:1602.01137*.

[113] Cai, F., M. De Rijke, et al. (2016) "A survey of query auto completion in information retrieval," *Foundations and Trends® in Information Retrieval*, **10**(4), pp. 273–363.

[114] Di Santo, G., R. McCreadie, C. Macdonald, and I. Ounis (2015) "Comparing approaches for query autocompletion," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp. 775–778.

[115] Krishnan, U., A. Moffat, and J. Zobel (2017) "A Taxonomy of Query Auto Completion Modes," in *Proceedings of the 22nd Australasian Document Computing Symposium*, ACM, p. 6.

[116] Bar-Yossef, Z. and N. Kraus (2011) "Context-sensitive query auto-completion," in *Proceedings of the 20th international conference on World wide web*, ACM, pp. 107–116.

[117] Schmidt, A., J. Hoffart, D. Milchevski, and G. Weikum (2016) "Context-sensitive auto-completion for searching with entities and categories," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ACM, pp. 1097–1100.

[118] Chien, S. and N. Immorlica (2005) "Semantic similarity between search engine queries using temporal correlation," in *Proceedings of the 14th international conference on World Wide Web*, ACM, pp. 2–11.

[119] Kharitonov, E., C. Macdonald, P. Serdyukov, and I. Ounis (2013) "User model-based metrics for offline query suggestion evaluation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 633–642.

[120] Guo, J., G. Xu, X. Cheng, and H. Li (2009) "Named entity recognition in query," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 267–274.

[121] Shokouhi, M. (2013) "Learning to personalize query auto-completion," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 103–112.

[122] Perozzi, B., R. Al-Rfou, and S. Skiena (2014) "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, pp. 701–710.

[123] Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2017) "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, **5**, pp. 135–146.

[124] Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013) "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119.

[125] Thanda, A., A. Agarwal, K. Singla, A. Prakash, and A. Gupta (2016) "A Document Retrieval System for Math Queries." in *NTCIR*.

[126] Le, Q. and T. Mikolov (2014) "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, pp. 1188–1196.

[127] Gao, L., Z. Jiang, Y. Yin, K. Yuan, Z. Yan, and Z. Tang (2017) "Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: can mathematical formulae be embedded like a natural language?" *arXiv preprint arXiv:1707.05154*.

[128] Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013) "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*.

[129] Krstovski, K. and D. M. Blei (2018) "Equation embeddings," *arXiv preprint arXiv:1803.09123*.

[130] Zhou, W., H. Li, and Q. Tian (2017) "Recent advance in content-based image retrieval: A literature survey," *arXiv preprint arXiv:1706.06064*.

[131] Siagian, C. and L. Itti (2007) "Rapid biologically-inspired scene classification using features shared with visual attention," *IEEE transactions on pattern analysis and machine intelligence*, **29**(2), pp. 300–312.

[132] Lowe, D. G. (2004) "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, **60**(2), pp. 91–110.

[133] Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012) "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105.

[134] Babenko, A. and V. Lempitsky (2015) "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE international conference on computer vision*, pp. 1269–1277.

[135] Gordo, A., J. Almazán, J. Revaud, and D. Larlus (2016) "Deep image retrieval: Learning global representations for image search," in *European conference on computer vision*, Springer, pp. 241–257.

[136] Radenović, F., G. Tolias, and O. Chum (2016) "CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples," in *European conference on computer vision*, Springer, pp. 3–20.

[137] Yue-Hei Ng, J., F. Yang, and L. S. Davis (2015) "Exploiting local features from deep networks for image retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 53–61.

[138] Razavian, A. S., J. Sullivan, S. Carlsson, and A. Maki (2016) "Visual instance retrieval with deep convolutional networks," *ITE Transactions on Media Technology and Applications*, **4**(3), pp. 251–258.

[139] Simonyan, K. and A. Zisserman (2014) "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*.

[140] Zanibbi, R. and B. Yuan (2011) "Keyword and image-based retrieval of mathematical expressions," in *Document Recognition and Retrieval XVIII*, vol. 7874, International Society for Optics and Photonics, p. 78740I.

[141] Marinai, S., B. Miotti, and G. Soda (2010) "Mathematical symbol indexing for digital libraries," in *Italian Research Conference on Digital Libraries*, Springer, pp. 113–124.

[142] Kohonen, T. (1982) "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, **43**(1), pp. 59–69.

[143] Davila Castellanos, K. (2017) "Symbolic and Visual Retrieval of Mathematical Notation using Formula Graph Symbol Pair Matching and Structural Alignment," .

[144] Kumar, P. P., A. Agarwal, and C. Bhagvati (2012) "A structure based approach for mathematical expression retrieval," in *International Workshop on Multi-disciplinary Trends in Artificial Intelligence*, Springer, pp. 23–34.

[145] Chaudhuri, S. and R. Kaushik (2009) "Extending autocompletion to tolerate errors," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, ACM, pp. 707–718.

[146] Zanibbi, R., A. Aizawa, M. Kohlhase, I. Ounis, G. Topic, and K. Davila (2016) "NTCIR-12 MathIR Task Overview," in *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016*.
URL `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/OVERVIEW/01-NTCIR12-OV-MathIR-ZanibbiR.pdf`

[147] Davila, K. and R. Zanibbi (2017) "Layout and semantics: Combining representations for mathematical formula search," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, pp. 1165–1168.

[148] Wu, J., K. M. Williams, H.-H. Chen, M. Khabsa, C. Caragea, S. Tuarob, A. G. Ororbia, D. Jordan, P. Mitra, and C. L. Giles (2015) "Citeseerx: Ai in a digital library search engine," *AI Magazine*, **36**(3), pp. 35–48.

[149] Wu, J., B. Kandimalla, S. Rohatgi, A. Sefid, J. Mao, and C. L. Giles (2018) "CiteSeerX-2018: A Cleansed Multidisciplinary Scholarly Big Dataset," *2018 IEEE International Conference on Big Data (Big Data)*, pp. 5465–5467.

[150] Yu, B., X. Tian, and W. Luo (2014) "Extracting mathematical components directly from PDF documents for mathematical expression recognition and retrieval," in *International Conference in Swarm Intelligence*, Springer, pp. 170–179.

[151] Baker, J. B., A. P. Sexton, and V. Sorge (2012) "MaxTract: Converting PDF to mbox\LaTeX, MathML and Text," in *International Conference on Intelligent Computer Mathematics*, Springer, pp. 422–426.

[152] Kohlhase, M. (2016) "Formats for Topics and Result Submissions for the MathIR Task at NTCIR-12," .

[153] Craswell, N. (2009) *Bpref*, Springer US, Boston, MA, pp. 266–267.
URL `https://doi.org/10.1007/978-0-387-39940-9_489`

[154] Järvelin, K. and J. Kekäläinen (2002) "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems (TOIS)*, **20**(4), pp. 422–446.

[155] Mansouri, B., A. Agarwal, D. Oard, and R. Zanibbi (2020) "Finding Old Answers to New Math Questions: The ARQMath Lab at CLEF 2020," in *European Conference on Information Retrieval*, Springer, pp. 564–571.

[156] Munavalli, R. and R. Miner (2006) "Mathfind: a math-aware search engine," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 735–735.

[157] Kohlhase, M., S. Anca, and C. Jucovschi "MathWebSearch 0.4, a semantic search engine for mathematics," .

[158] Zhong, W. and R. Zanibbi (2019) "Structural Similarity Search for Formulas Using Leaf-Root Paths in Operator Subtrees," in *European Conference on Information Retrieval*, Springer, pp. 116–129.

[159] Kristianto, G. Y., G. Topic, and A. Aizawa (2016) "MCAT Math Retrieval System for NTCIR-12 MathIR Task." in *NTCIR*.

[160] Gao, L., K. Yuan, Y. Wang, Z. Jiang, and Z. Tang (2016) "The Math Retrieval System of ICST for NTCIR-12 MathIR Task." in *NTCIR*.

[161] Mansouri, B., A. Agarwal, D. Oard, and R. Zanibbi (2020) "Finding Old Answers to New Math Questions: The ARQMath Lab at CLEF 2020," in *Advances in Information Retrieval* (J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, eds.), Springer International Publishing, Cham, pp. 564–571.

[162] Yang, W., H. Zhang, and J. Lin (2019) "Simple applications of BERT for ad hoc document retrieval," *arXiv preprint arXiv:1903.10972*.

[163] Yang, P., H. Fang, and J. Lin (2017) "Anserini: Enabling the use of Lucene for information retrieval research," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1253–1256.

[164] Cormack, G. V., C. L. Clarke, and S. Buettcher (2009) "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 758–759.

[165] Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2018) "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*.

[166] Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017) "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008.

[167] Dai, Z. and J. Callan (2019) "Deeper text understanding for IR with contextual neural language modeling," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 985–988.

[168] Yilmaz, Z. A., S. Wang, W. Yang, H. Zhang, and J. Lin (2019) "Applying BERT to document retrieval with birch," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pp. 19–24.

[169] Nogueira, R., W. Yang, K. Cho, and J. Lin (2019) "Multi-stage document ranking with BERT," *arXiv preprint arXiv:1910.14424*.

[170] Craswell, N., B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees (2020) "Overview of the trec 2019 deep learning track," *arXiv preprint arXiv:2003.07820*.

[171] Bajaj, P., D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen, et al. (2016) "Ms marco: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*.

[172] Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov (2019) "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*.

[173] Gururangan, S., A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith (2020) "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks," *arXiv preprint arXiv:2004.10964*.

[174] Sun, C., X. Qiu, Y. Xu, and X. Huang (2019) "How to fine-tune bert for text classification?" in *China National Conference on Chinese Computational Linguistics*, Springer, pp. 194–206.

[175] Sakai, T. and N. Kando (2008) "On information retrieval metrics designed for evaluation with incomplete relevance assessments," *Information Retrieval*, **11**(5), pp. 447–470.

[176] Zhong, W., S. Rohatgi, J. Wu, C. L. Giles, and R. Zanibbi (2020) "Accelerating Substructure Similarity Search for Formula Retrieval," in *Advances in Information Retrieval* (J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, eds.), Springer International Publishing, Cham, pp. 714–727.

[177] Fraser, D., A. Kane, and F. W. Tompa (2018) "Choosing Math Features for BM25 Ranking with Tangent-L," in *Proceedings of the ACM Symposium on Document Engineering 2018*, pp. 1–10.

[178] Greiner-Petter, A., T. Ruas, M. Schubotz, A. Aizawa, W. Grosky, and B. Gipp (2019) "Why Machines Cannot Learn Mathematics, Yet," *arXiv preprint arXiv:1905.08359*.

[179] Mansouri, B., S. Rohatgi, D. W. Oard, J. Wu, C. L. Giles, and R. Zanibbi (2019) "Tangent-CFT: An Embedding Model for Mathematical Formulas," in *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, ACM, pp. 11–18.

[180] Kristianto, G. Y. and A. Aizawa "MCAT Math Retrieval System for NTCIR-12 MathIR Task," .

[181] Davila, K., R. Joshi, S. Setlur, V. Govindaraju, and R. Zanibbi (2019) "Tangent-V: Math Formula Image Search Using Line-of-Sight Graphs," in *European Conference on Information Retrieval*, Springer, pp. 681–695.

[182] Krizhevsky, A. and G. E. Hinton (2011) "Using very deep autoencoders for content-based image retrieval," in *ESANN*.

[183] Pfahler, L., J. Schill, and K. Morik (2019) "The Search for Equations–Learning to Identify Similarities between Mathematical Expressions," *Machine Learning and Knowledge Discovery in Databases, ECML, PKDD*.

[184] Sennrich, R., B. Haddow, and A. Birch (2015) "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*.

[185] Masci, J., U. Meier, D. Cireşan, and J. Schmidhuber (2011) "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*, Springer, pp. 52–59.

[186] Ghifary, M., W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li (2016) "Deep reconstruction-classification networks for unsupervised domain adaptation," in *European Conference on Computer Vision*, Springer, pp. 597–613.

[187] Mao, X., C. Shen, and Y.-B. Yang (2016) "Image Restoration Using Very Deep Convolutional Encoder-Decoder Networks with Symmetric Skip Connections," in *Advances in Neural Information Processing Systems 29* (D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, eds.), Curran Associates, Inc., pp. 2802–2810.

[188] Buckley, C. and E. M. Voorhees (2004) "Retrieval evaluation with incomplete information," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, pp. 25–32.

[189] Chang, Y. and H. Deng (2020) *Query Understanding for Search Engines*, The Information Retrieval Series, Springer International Publishing AG.

[190] Kacprzak, E., L. M. Koesten, L.-D. Ibáñez, E. Simperl, and J. Tennison (2017) "A query log analysis of dataset search," in *International Conference on Web Engineering*, Springer, pp. 429–436.

[191] Khabsa, M., Z. Wu, and C. L. Giles (2016) "Towards better understanding of academic search," in *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, IEEE, pp. 111–114.

[192] Herskovic, J. R., L. Y. Tanaka, W. Hersh, and E. V. Bernstam (2007) "A Day in the Life of PubMed: Analysis of a Typical Day's Query Log," *J Am Med Inform Assoc*, **14**(2), pp. 212–220.

[193] Kandimalla, B., S. Rohatgi, J. Wu, and C. L. Giles (2021) "Large Scale Subject Category Classification of Scholarly Papers With Deep Attentive Neural Networks," *Front. Res. Metr. Anal.*, **5**, p. 31.

[194] Cohan, A., S. Feldman, I. Beltagy, D. Downey, and D. S. Weld (2020) "SPECTER: Document-level Representation Learning using Citation-informed Transformers," in *ACL*.

[195] Sefid, A., J. Wu, C. G. Allen, J. Zhao, L. Liu, C. Caragea, P. Mitra, and C. L. Giles (2019) "Cleaning noisy and heterogeneous metadata for record linking across scholarly big datasets," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9601–9606.

[196] Caragea, C., F. A. Bulgarov, A. Godea, and S. Das Gollapalli (2014) "Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, Doha, Qatar, pp. 1435–1446.
URL https://www.aclweb.org/anthology/D14-1150

[197] Zhang, E., N. Gupta, R. Nogueira, K. Cho, and J. Lin (2020), "Rapidly Deploying a Neural Search Engine for the COVID-19 Open Research Dataset: Preliminary Thoughts and Lessons Learned," 2004.05125.

[198] Lin, J. (2009) "Is searching full text more effective than searching abstracts?" *BMC bioinformatics*, **10**(1), p. 46.

[199] Wu, J., K. Kim, and C. L. Giles (2019) "CiteSeerX: 20 years of service to scholarly big data," *Proceedings of the Conference on Artificial Intelligence for Data Discovery and Reuse*.

[200] Koren, J., Y. Zhang, and X. Liu (2008) "Personalized Interactive Faceted Search," Association for Computing Machinery, New York, NY, USA.

[201] Beltagy, I., K. Lo, and A. Cohan (2019) "SciBERT: A Pretrained Language Model for Scientific Text," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3606–3611.

[202] Bhagavatula, C., S. Feldman, R. Power, and W. Ammar (2018) "Content-Based Citation Recommendation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 238–251.

[203] Wade, A. D. (2022) "The Semantic Scholar Academic Graph (S2AG)," in *Companion Proceedings of the Web Conference 2022*, WWW '22, Association for Computing Machinery, New York, NY, USA, p. 739.
URL https://doi.org/10.1145/3487553.3527147

[204] Kinney, R., C. Anastasiades, R. Authur, I. Beltagy, J. Bragg, A. Buraczynski, I. Cachola, S. Candra, Y. Chandrasekhar, A. Cohan, et al. (2023) "The Semantic Scholar Open Data Platform," *arXiv preprint arXiv:2301.10140*.

[205] Meuschke, N., A. Jagdale, T. Spinde, J. Mitrović, and B. Gipp (2023) "A Benchmark of PDF Information Extraction Tools Using a Multi-task and Multi-domain Evaluation Framework for Academic Documents," in *Information for a Better World: Normality, Virtuality, Physicality, Inclusivity: 18th International Conference, iConference 2023, Virtual Event, March 13–17, 2023, Proceedings, Part II*, Springer, pp. 383–405.

[206] Prabhakaran, V., W. L. Hamilton, D. McFarland, and D. Jurafsky (2016) "Predicting the Rise and Fall of Scientific Topics from Trends in their Rhetorical Framing," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Berlin, Germany, pp. 1170–1180.
URL https://aclanthology.org/P16-1111

[207] Yin, W., J. Hay, and D. Roth (2019) "Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, Hong Kong, China, pp. 3914–3923.
URL https://aclanthology.org/D19-1404

[208] Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer (2020) "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Online, pp. 7871–7880.
URL https://aclanthology.org/2020.acl-main.703

[209] Li, Y., Y. Zhang, Z. Zhao, L. Shen, W. Liu, W. Mao, and H. Zhang (2022) "CSL: A Large-scale Chinese Scientific Literature Dataset," in *Proceedings of the 29th International Conference on Computational Linguistics*, International Committee on Computational Linguistics, Gyeongju, Republic of Korea, pp. 3917–3923.
URL https://aclanthology.org/2022.coling-1.344

[210] Wang, Z., P. Wang, L. Huang, X. Sun, and H. Wang (2022) "Incorporating Hierarchy into Text Encoder: a Contrastive Learning Approach for

Hierarchical Text Classification," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Dublin, Ireland, pp. 7109–7119.
URL https://aclanthology.org/2022.acl-long.491

[211] Li, C., A. Sun, and A. Datta (2012) "Twevent: Segment-Based Event Detection from Tweets," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, Association for Computing Machinery, New York, NY, USA, p. 155–164.
URL https://doi.org/10.1145/2396761.2396785

[212] Qin, Y., Y. Zhang, M. Zhang, and D. Zheng (2018) "Frame-Based Representation for Event Detection on Twitter," *IEICE Transactions on Information and Systems*, **E101.D**(4), pp. 1180–1188.

[213] Hsu, T.-Y., C. L. Giles, and T.-H. Huang (2021) "SciCap: Generating Captions for Scientific Figures," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Association for Computational Linguistics, Punta Cana, Dominican Republic, pp. 3258–3264.
URL https://aclanthology.org/2021.findings-emnlp.277

[214] Lu, P., S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan (2022) "Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering," in *The 36th Conference on Neural Information Processing Systems (NeurIPS)*.

[215] Zhang, E., N. Gupta, R. Tang, X. Han, R. Pradeep, K. Lu, Y. Zhang, R. Nogueira, K. Cho, H. Fang, et al. (2020) "Covidex: Neural ranking models and keyword search infrastructure for the covid-19 open research dataset," *arXiv preprint arXiv:2007.07846*.

[216] Rohatgi, S., Z. Karishma, J. Chhay, S. R. R. Keesara, J. Wu, C. Caragea, and C. L. Giles (2020) "COVIDSeer: Extending the CORD-19 Dataset," *Proceedings of the ACM Symposium on Document Engineering 2020*.

[217] Ranathunga, S. and N. de Silva (2022) "Some languages are more equal than others: Probing deeper into the linguistic disparity in the nlp world," *arXiv preprint arXiv:2210.08523*.

[218] Özyurt, I. B. and J. S. Grethe (2019) "Iterative Document Retrieval via Deep Learning Approaches for Biomedical Question Answering," *2019 15th International Conference on eScience (eScience)*, pp. 533–538.

[219] Luo, M., A. Mitra, T. Gokhale, and C. Baral (2022) "Improving Biomedical Information Retrieval with Neural Retrievers," in *AAAI Conference on Artificial Intelligence*.

[220] Jin, Q., Z. Yuan, G. Xiong, Q. Yu, C. Tan, M. Chen, S. Huang, X. Liu, and S. Yu (2021) "Biomedical Question Answering: A Comprehensive Review," *ArXiv*, **abs/2102.05281**.

[221] Gao, J., C. Xiong, P. Bennett, and N. Craswell (2022) "Neural Approaches to Conversational Information Retrieval," *Neural Approaches to Conversational Information Retrieval*.

[222] Kinney, R. M., C. Anastasiades, R. Authur, I. Beltagy, J. Bragg, A. Buraczynski, I. Cachola, S. Candra, Y. Chandrasekhar, A. Cohan, M. Crawford, D. Downey, J. Dunkelberger, O. Etzioni, R. Evans, S. Feldman, J. Gorney, D. W. Graham, F. Hu, R. Huff, D. King, S. Kohlmeier, B. Kuehl, M. Langan, D. Lin, H. Liu, K. Lo, J. Lochner, K. MacMillan, T. Murray, C. Newell, S. Rao, S. Rohatgi, P. L. Sayre, Z. Shen, A. Singh, L. Soldaini, S. Subramanian, A. Tanaka, A. D. Wade, L. M. Wagner, L. L. Wang, C. Wilhelm, C. Wu, J. Yang, A. Zamarron, M. van Zuylen, and D. S. Weld (2023) "The Semantic Scholar Open Data Platform," *ArXiv*, **abs/2301.10140**.

[223] Jin, Q., B. Dhingra, Z. Liu, W. W. Cohen, and X. Lu (2019) "PubMedQA: A Dataset for Biomedical Research Question Answering," in *Conference on Empirical Methods in Natural Language Processing*.

[224] Nori, H., N. King, S. M. McKinney, D. Carignan, and E. Horvitz (2023) "Capabilities of GPT-4 on Medical Challenge Problems," *ArXiv*, **abs/2303.13375**.

[225] Luo, R., L. Sun, Y. Xia, T. Qin, S. Zhang, H. Poon, and T.-Y. Liu (2022) "BioGPT: Generative Pre-trained Transformer for Biomedical Text Generation and Mining," *Briefings in bioinformatics*.

[226] Singhal, K., S. Azizi, T. Tu, S. Mahdavi, J. L. K. Wei, H. W. Chung, N. Scales, A. K. Tanwani, H. J. Cole-Lewis, S. J. Pfohl, P. A. Payne, M. G. Seneviratne, P. Gamble, C. Kelly, N. Scharli, A. Chowdhery, P. D. Mansfield, B. A. y Arcas, D. R. Webster, G. S. Corrado, Y. Matias, K. H.-L. Chou, J. Gottweis, N. Tomaev, Y. Liu, A. Rajkomar, J. K. Barral, C. Semturs, A. Karthikesalingam, and V. Natarajan (2022) "Large Language Models Encode Clinical Knowledge," *ArXiv*, **abs/2212.13138**.

[227] Kanakarajan, K. R., B. Kundumani, and M. Sankarasubbu (2021) "BioELECTRA:Pretrained Biomedical text Encoder using Discriminators," in *Workshop on Biomedical Natural Language Processing*.

[228] Gu, Y., R. Tinn, H. Cheng, M. R. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon (2020) "Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, **3**, pp. 1 – 23.

Vita

Shaurya Rohatgi

Education

The Pennsylvania State University *Aug. 2017 – Present*
Ph.D. in Information Sciences and Technology
ABV - Indian Institute of Information Technology and Management *Sep. 2009 – Jun. 2014*
Integrated M.Tech. in Information Technology

Experiences

The Pennsylvania State University *Aug. 2017 – Present*
Research Assistant
Allen Institute of Artificial Intelligence *May. 2021 – Present*
Research Intern/Collaborator

Publications/Pre-prints

[1] Behrooz Mansouri, **Shaurya Rohatgi**, Douglas W Oard, Jian Wu, C Lee Giles, Richard Zanibbi. Tangent-CFT: An embedding model for mathematical formulas. *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*, 2019

[2] Kunho Kim, **Shaurya Rohatgi**, C Lee Giles. Hybrid deep pairwise classification for author name disambiguation. *Proceedings of the 28th ACM international conference on information and knowledge management*, 2369-2372, 2019

[3] **Shaurya Rohatgi**, Zeba Karishma, Jason Chhay, Sai Raghav Reddy Keesara, Jian Wu, Cornelia Caragea, C Lee Giles. Covidseer: Extending the CORD-19 dataset. *Proceedings of the ACM Symposium on Document Engineering 2020*, 1-4, 2020

[4] **Shaurya Rohatgi**, Jian Wu, C Lee Giles. PSU at CLEF-2020 ARQMath Track: Unsupervised Re-ranking using Pretraining. *CEUR Workshop Proceedings*, 2020

[5] **Shaurya Rohatgi**, C Lee Giles, Jian Wu. What Were People Searching For? A Query Log Analysis of An Academic Search Engine. *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 342-343, 2021

[6] **Shaurya Rohatgi**, Jian Wu, C Lee Giles. Ranked List Fusion and Re-ranking with Pre-trained Transformers for ARQMath Lab. *CEUR Workshop Proceedings*, 2021

[7] Jian Wu, **Shaurya Rohatgi**, Sai Raghav Reddy Keesara, Jason Chhay, Kevin Kuo, Arjun Manoj Menon, Sean Parsons, Bhuvan Urgaonkar, C Lee Giles. Building an Accessible, Usable, Scalable, and Sustainable Service for Scholarly Big Data. *2021 IEEE International Conference on Big Data (Big Data)*, 141-152, 2021

[8] **Shaurya Rohatgi**, Doug Downey, Daniel King, Sergey Feldman. S2AMP: a high-coverage dataset of scholarly mentorship inferred from publications. *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, 1-5, 2022