Integrated Digital Library System for Long Documents and their Elements

Satvik Chekuri
Virginia Tech
Blacksburg, Virginia, USA
satvikchekuri@vt.edu

Prashant Chandrasekar
University of Mary Washington
Fredericksburg, Virginia, USA
pchandra@umw.edu

Bipasha Banerjee
Virginia Tech
Blacksburg, Virginia, USA
bipashabanerjee@vt.edu

Sung Hee Park
Virginia Tech
Blacksburg, Virginia, USA
shpark@vt.edu

Nila Masrourisaadat
Virginia Tech
Blacksburg, Virginia, USA
nilamasrouri@vt.edu

Aman Ahuja
Virginia Tech
Blacksburg, Virginia, USA
aahuja@vt.edu

William A. Ingram
waingram@vt.edu
Virginia Tech
Blacksburg, Virginia, USA

Edward A. Fox
Virginia Tech
Blacksburg, Virginia, USA
fox@vt.edu

ABSTRACT

We describe a next-generation integrated Digital Library (DL) system that addresses the numerous goals associated with long documents such as Electronic Theses and Dissertations (ETDs). Our extensible workflow-centric design supports a variety of users/personas (e.g., researchers, curators, and experimenters) who can benefit from improved access to ETDs and the content buried therein. Our approach leverages natural language processing, deep learning, information retrieval, and software engineering methods. The services cover ingesting, storing, curating, analyzing, detecting, extracting, classifying, summarizing, topic modeling, browsing, searching, retrieving, recommending, visualizing/reporting, and interacting with ETDs and derivative text/image-based elements/objects. Workflows connect the services and their APIs, along with UI-based access. We believe our approach can guide others to combine tailored user support, research, and education by way of extensible DLs.

CCS CONCEPTS

• **Information systems → Digital libraries and archives**; *Information retrieval*; Document representation; Retrieval tasks and goals.

KEYWORDS

Digital Library, Information System, Information Retrieval, Deep Learning, NLP

1 INTRODUCTION

The digital library (DL) community faces many challenges. We propose a new approach to developing DLs that benefits diverse users. This paper describes some of those challenges and benefits, explaining our system architecture, research investigations, implementation, and results. Research questions include:

(1) How can a DL support a diverse set of user personas, including curators, experimenters, and a variety of general end-users – with diverse interests in accessing content?
(2) How can a DL better support access to long and complex documents, going beyond the traditional paradigms that only consider metadata or full-text (documents as a whole)?
(3) How can user interfaces (UIs) and services be connected by integrating: ingesting, storing, curating, analyzing, detecting, extracting, classifying, summarizing, topic modeling, browsing, searching, retrieving, recommending, visualizing/reporting, and interacting with content – that leverages natural language processing (NLP), deep learning, information retrieval (IR), and software engineering methods?
(4) How can a DL be made extensible, so it is easy to enhance to support additional personas, changing user experience (UX) requirements, improved methods/technologies, and diverse hardware/cloud environments?
(5) How can academics advance their research while at the same time help students by way of problem/project based learning and mentoring by subject-matter experts (SMEs)?

The problems related to handling long documents (e.g., electronic theses and dissertations – ETDs [53]) provide an appropriate context in which to answer these questions; see Table 1. Libraries have long provided metadata based support (based largely on title, keywords, and abstract) for books and other long works. Surprisingly, the IR and DL communities have done little to improve that support, in spite of decades of improvements in the handling of short works (e.g., web pages, papers, and articles). Full-text search and passage retrieval [41, 64] provide some assistance, but help little to address context, which for shorter works has been managed by way of fields; those have little use when documents have complex structures. Further, XML-based techniques have had little appeal for users [25, 33, 42, 54]. Special methods have been developed for document elements like figures [39], tables [2, 47, 58], equations [63], and references [17, 59] – but there has been little integration of those methods in support of DLs handling long documents.

Books are ubiquitous in scholarship. They are used by learners up through the college years, and are the most important type of work in many disciplines (e.g., humanities). Highly visible, books have long had identifiers (ISBNs and DOIs) and are easy to cite.

Table 1: System Workflows and Services for Stakeholders (User Personas)

	Workflows		Services		
Curator	ETD Curation	Usage Statistics	Upload ETDs; Provide metadata	Browse, validate, update ETDs	ETDs Stats
Researcher	ETD/Chapter Search	ETD/Chapter Categorization; Topics	ETD Object Detection	Indexing; Ranking	ETD Summarization
	ETD/Chapter Recommendation		Recommendation	ETD/Chapter Categorization	ETD Topic Modeling
Experimenter	Search for ETDs/Chapters	Get recommendations	Run experiment with selected options	UI service to display the experiment result	Log and display results
	View documents	User management	Log results with timestamps and metrics	Run experiment with selected options	

They are highly regarded for those seeking promotion or advancement, including graduate students completing their research and degree programs by filing an ETD, which usually is freely available to the world, facilitating open access knowledge sharing without economic barriers [71]. Thus the set of personas associated with ETDs is broad, including student authors, student learners and researchers, faculty, those in the global research community, university curators (e.g., librarians and archivists), and graduate administrators. Added to that set are those in the IR and DL communities engaged in research on long documents, who are just beginning on the long required process of devising new approaches and evaluating them experimentally.

To support all these personas and their needs, an integrated DL system could provide a framework, if an extensible architecture could be employed. Given the current technologies involved in the IR and DL fields, there are many dimensions requiring extensibility. One involves handling the growing number of user personas, and ever changing UX and UI requirements. Regarding use cases, some seek to: learn by way of Literature Review chapters, identify what to study among the References cited, find a reproducible approach detailed in a Methodology section, choose metrics and Evaluation techniques, identify open problems mentioned as Future Work, check on the effects of sponsoring research by extracting from Acknowledgments, or find Datasets to work with. Regarding use, many expect support for searching, browsing, recommending, visualizing, and summarizing – leveraging classification and/or topic modeling systems. Another dimension involves the many types of document elements found in works like ETDs, some of which are rare, making it hard to apply deep learning methods. Yet another involves the diversity in the works, varying across: disciplines, national/regional/local/school/department settings, styles and structures, scanning/OCRing techniques, digital publishing methods, terminology and language use, and use of multimedia/hypermedia. Regarding implementations, there are constant changes in software engineering and hardware/cloud environments and approaches.

Given the need to implement such a system, and the limited resources available to libraries and DL researchers, it is fortunate that problem/project based learning provides a setting for graduate students to assist, and help build this system [19, 26, 37, 57, 66].

2 USAGE SCENARIO/USER PERSONAS

A commonly used tool for user experience design is the development of user personas. A user persona represents a class of users, based on common goals, motivations, and behaviors. In developing these personas, we borrow from the 5S framework for DLs [24]. We base our user personas on the societies construct of 5S. For this work, we have created three user personas: *curators*, *researchers*, and *experimenters*. We describe the needs, workflows, and system requirements for each persona in the paragraphs that follow.

The *curator* persona represents those responsible for collecting, managing, and preserving digital collections, and ensuring long-term access to digital resources. To support their needs, the DL should provide a suite of collection management tools to allow the curator to organize collections and create / edit metadata. Curators create, upload, and edit digital items and metadata in batches, as well as one at a time. Curators also track usage and performance, so the system should provide analytical tools to measure and report system usage patterns and other metrics. In this regard, curators overlap with those administering graduate programs and those overseeing academic research, i.e., who engage in analysis and reporting; curators can assist them in those activities. Moreover, curators should ensure the long-term preservation of digital assets by monitoring the integrity of digital assets to make sure that none have become altered or corrupted, so that the archive remains trustworthy and accessible. The curator should set access restrictions based on user roles and permissions to ensure the privacy and security of sensitive information. The curator also needs to be able to quickly find items in the DL, so the system needs to facilitate advanced search and browse functionality. The curator needs a DL system that is interoperable with other parts of the curation workflow, so the system should adhere to open standards for interoperability. Finally, the curator persona would benefit from a system that integrates machine learning or AI in support of curatorial tasks, such as automated metadata generation, format conversion, and machine-generated summaries, keywords, or related works, which could help end users to discover and understand the content.

The *researcher* persona represents the learners, students, faculty, and community of researchers who use the DL. Most researchers have a background in a specific field of study. They may be working at a university or research institute, or as an independent scholar. Researchers use DLs to support their research workflows, which include searching, browsing, reading, and downloading information resources, as well as collecting and analyzing data, and synthesizing findings. They may use tools within the DL system to annotate items, visualize data, and share data or findings with other researchers. To serve the needs of researchers, the DL should facilitate advanced search and browse capabilities, e.g., faceted search, results filtering, and NLP. The DL should adhere to open standards for interoperability and integrate seamlessly with other tools in the research workflow. The DL should provide a personalized experience, allowing researchers to customize UIs, save search queries and result sets, and get personalized recommendations. Likewise, the DL should provide access controls ensuring secure protection of sensitive information. A DL built for researchers might also include features for data visualization and collaboration among researchers.

We separate the *experimenter* persona from other researchers due to their unique needs, which partially overlap with that of *developer*. Experimenters use the data contained in the DL for computationally intensive work, such as text and data mining or statistical analysis. To support the needs of experimenters, the DL system will need to be designed to support large data sets and high-performance computing resources. The DL should support open data interoperability standards to integrate with experimenters' workflows, collaboration, and reproducibility. Experimenters also require a system that provides them tools for organizing and managing data and workflows, including version control. Additionally, the DL should provide experimenters with access controls that ensure secure protection of their data, their workflows, and any other sensitive information. Thus, experimenters include many in the DL R&D world, including data scientists, algorithm developers, system builders, and a broad range of innovators.

3 FRAMEWORK

Figure 1 illustrates the architecture of our system and its key dataflows. Services correspond to user tasks that address user goals, which are based on customer discovery [81].

3.1 Architectural Elements

3.1.1 User-facing Services. There is support through UIs for each of the 3 personas, including searching and recommending over a collection of ETDs and related digital objects. Our approach to achieve such is to build containerized micro services. Examples are:

- A front-end service which lets users enter queries, and aims to show relevant documents and digital objects.
- A search service to index and search documents, as well as log user interactions.
- A recommendation service that builds and leverages models which consume user logs and ETD data.

3.1.2 Underlying Services. We use ElasticSearch[6, 15, 40] to index and search over the ETD metadata and chapters. We created a REST API abstraction to interact with ElasticSearch using a Python Flask [16] server. The search service uses the Python requests [60] library to query the curator team's GET ETD API to receive ETD metadata and the chapter objects related to the ETD. We use the sentence-transformers [21] Python module with the *distilroberta* [74] language model to create dense vectors for abstract and title text, and index them into ElasticSearch. We utilized a variety of tools and libraries to build our system. To aid those interested in reproducing our efforts, we explain the versions and details [22].

3.1.3 Search Specifics. Specific details of our work with ElasticSearch (ES) include: (1) Proper configuration of elasticsearch.yml for logging, security, etc.; (2) Using docker-compose.yml to containerize the ES cluster so as to easily run locally; (3) Index templates; (4) SQL queries to query data from a relational database; (5) Code to massage the queried data into JSON format that is compliant with the ES schema; (6) Code to index the data from previous steps; (7) Code to allow updating of indexed documents with new data; (8) Querying ES over digital objects and get appropriate results; (9) Dockerfile that containerizes the code and ES; (10) Uploading PyTorch machine learning models to ES from v8.x and using it for text embeddings and vector search; and (11) Employing the kNN API available from ES v8.x.

3.2 Content and Representation

Out of a collection of roughly 1/2 million ETDs collected from around the USA, we identified a subset of 57,129. From this we selected a 5K subset suitable for students to process during a semester long course (CS5604 in Fall 2022). The content and representation layer includes a repository with database tables (including for metadata) and a file system for larger entities. There are APIs for accessing and communicating with the repository.

3.2.1 Database. We use a PostgreSQL server with 9 database tables. One is for the abstract ETD, and another for each version of its metadata. There also is a table for an abstract derived object. Tables associated with objects include: metadata, summarization, classification, and topic set. Since there can be multiple schemes for topics and classification, there are tables to specify each of those. An ETD identifier is given as a 7 digit integer.

3.2.2 File System. There is an hierarchical directory structure, with upper level for the collections and lower level for the different ETDs in a collection. With our sample collection, it suffices to have a 3 digit number for each source collection, and a 4 digit number for the different ETDs in a collection. Each ETD has subdirectories for the different types of associated or derived content.

3.2.3 APIs. Our system also provides APIs to read/write/update/delete records in the database and entries in the file system. This allowed the rest of the system implementation to proceed based on API calls, thus decoupling it from repository operation. Student teams all agreed on the specifications for the APIs, which are listed in Table 2. The APIs can be grouped as follows:

- **Curator Webpage**: Help the curator to access the ETDs and perform basic operations like create, read, and update.
- **Accessing File System**: Help access digital objects in the file system. The upload API response is the absolute path.
- **Accessing SQL Database**: Help other teams to save their results in the database and access existing ETD data.
- **Logging**: We track the users' search queries and the results they click on. User ID and ETD ID pairs can be found in ElasticSearch logs, and help with recommendation. We expose

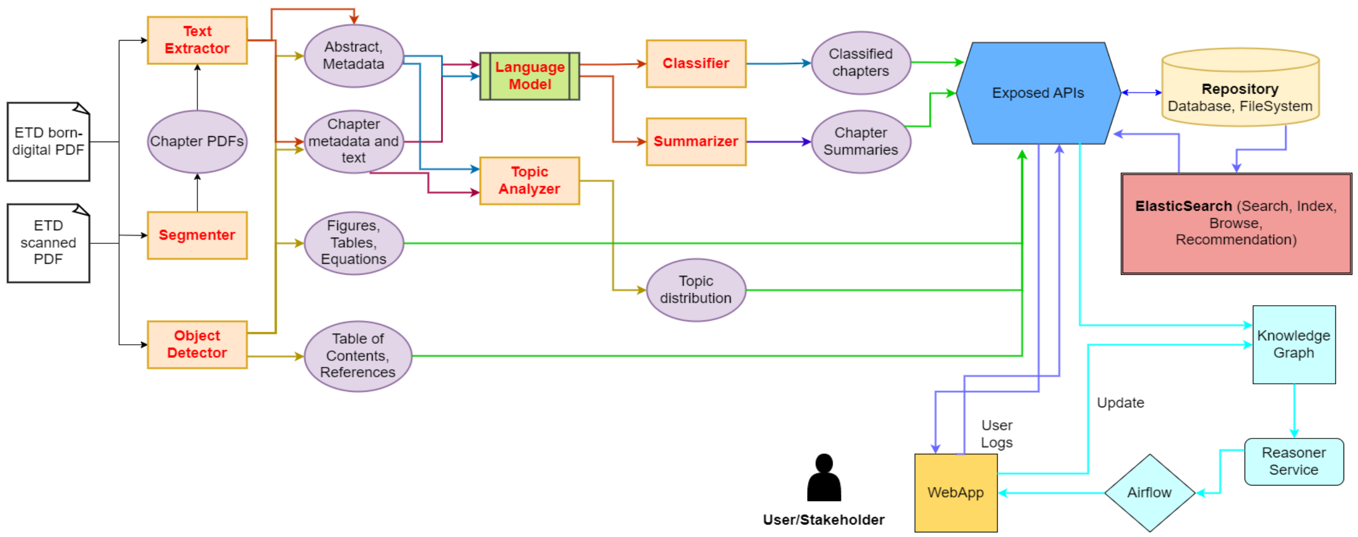


Figure 1: Architecture of an extensible IR system. Oval = input/output data (indexed for search). Square = service.

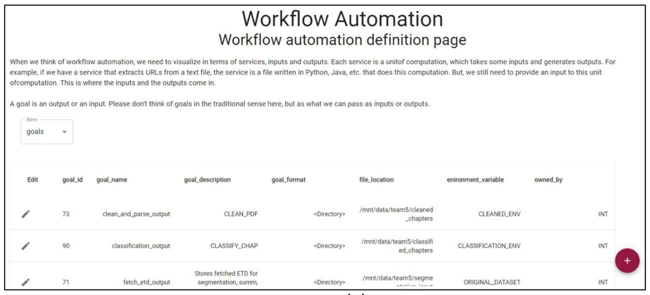
Table 2: API List. Key for Personas: C= Curator, E=Experimenter, R=Researcher

Purpose	Method	Description	Persona(s)
Get Statistics Report of School Contribution Amount	GET	Provides statistical reports of the stored ETD collections; intended mainly for front-end page use.	C
Get ETD and its Object	GET	Returns all ETDs along with their objects. The “start” and “limit” parameters define the pagination function.	E R
Get ETD PDF by etd_id	GET	Returns the ETD PDF by etd_id.	E R
Get ETD by etd_id	GET	Returns an ETD corresponding to an etd_id.	E R
Get Objects by etd_id	GET	Returns all object details that belong to the ETD.	E R
Save Summarization	POST	Takes a JSON object with two fields and values, and inserts into the summarization table with the associated object_id.	E
Save Classification	POST	Given a model and set of labels, inserts into classification tables. Updates `object_classification` table. label name = `class_name`	E
Get Object Relations Present in the Knowledge Graph by etd_id	GET	Returns object relations present in the knowledge graph.	R
Get Object File by object_id	GET	If the type of the object is an image, this API returns the image file.	R
Save Object Topic	POST	Takes a JSON body with a set of topic terms and the probability of each term. Saves it in the relevant topic-related table.	R
Get ETD Page Image	GET	Returns the specific page of the ETD.	R
Add New ETD	POST	Adds a new ETD to the database and uploads the file to the file system.	C R
Save Detected Object	POST	Saves a PDF/image file into the file system, and inserts a row into “object” metadata table with file system path and object type. Returns object_id.	E R
Save Detected Objects	POST	This batch-upload version of the “Save Detected Object” API takes a ZIP file and a list of objects in JSON format.	E R
Save ETD Page Images	POST	Saves the generated page images for a particular ETD.	E
Save Topic Modelling Results	POST	Saves the generated topic modeling results.	R

the logged data to the recommendation model through an API. The recommendation system calls the API endpoint with a user ID and receives the user’s history of search queries with the search results they clicked on.

3.3 Workflows

While system operations can proceed with code calling APIs, a higher level of abstraction results from running a series of services by calling a workflow that connects them. We use Apache Airflow to execute workflows that are specified as directed acyclic graphs



Workflow Automation

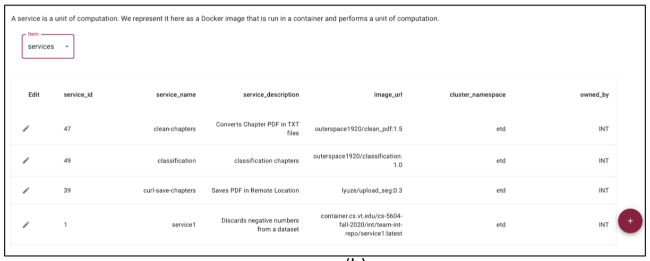
Workflow automation definition page

When we think of workflow automation, we need to visualize in terms of services, inputs and outputs. Each service is a unitof computation, which takes some inputs and generates outputs. For example, if we have a service that extracts URLs from a text file, the service is a file written in Python, Java, etc. that does this computation. But, we still need to provide an input to this unit ofcomputation. This is where the inputs and the outputs come in.

A goal is an output or an input. Please don't think of goals in the traditional sense here, but as what we can pass as inputs or outputs.

Edit	goal_id	goal_name	goal_description	goal_format	file_location	enironment_variable	owned_by
	73	clean_and_parse_output	CLEAN_PDF	<Directory>	/mnt/data/team5/cleaned_chapters	CLEANED_ENV	INT
	90	classification_output	CLASSIFY_CHAP	<Directory>	/mnt/data/team5/classified_chapters	CLASSIFICATION_ENV	INT
	71	fetch_etd_output	Stores fetched ETD for segmentation, summ,	<Directory>	/mnt/data/team5/segme	ORIGINAL_DATASET	INT

(a)



A service is a unit of computation. We represent it here as a Docker image that is run in a container and performs a unit of computation.

Edit	service_id	service_name	service_description	image_url	cluster_namespace	owned_by
	47	clean-chapters	Converts Chapter PDF in TXT files	outerspace1920/clean_pdf:1.5	etd	INT
	49	classification	classification chapters	outerspace1920/classification:1.0	etd	INT
	39	curl-save-chapters	Saves PDF in Remote Location	lyuze/upload_seg:0.3	etd	INT
	1	service1	Discards negative numbers from a dataset	container.cs.vt.edu/cs-5604-fall-2020/int/team-int-repo/service1:latest	etd	INT

(b)

Figure 2: Workflow Automation UI. (a) Goals, (b) Services.

of services. To further simplify the implementation, we provide an extensible infrastructure (see bottom right of Figure 1) so goals can be interpreted by a reasoner, which refers to a knowledge graph that associates goals, workflows, and services. To allow no-code specification of the workflows we include a UI with screens to describe and connect the goals (Figure 2 (a)) and services (Figure 2 (b)). Thus, a UX researcher (UX-R) can specify the design of a workflow-centric DL system, according to an in-depth description of the methodology for building an extensible information system [23].

Examples of information goals (in the domain of long document IR, with ETDs) for researcher persona are: (1) extracting full-text from PDF, (2) extracting chapters, (3) extracting tables and figures, (4) classifying chapters into custom categories, etc.

Experimenter personas want: (1) to access training data for their models, (2) a platform to train a machine-learning model to perform object detection and topic modeling, (3) to run experiments using these models and compare their results for optimal use, (4) to offload the results of a trained model to a persistent storage, etc.

The current DL includes workflow implementations for long document Segmentation, Classification, Summarization, Object detection, Indexing, among others.

4 CORE SERVICES FOR CURATORS

From the content provider's perspective of the system, we have five major components in the system design: the cloud server, the PostgreSQL database, the file system, the Flask web application for the curator, and the APIs for the experimenters.

Our local Kubernetes Docker cluster orchestrates the containers with PostgreSQL, Flask, and other necessary packages. The file system is mounted from the database container, which has information about the file system location of each ETD. The Flask application

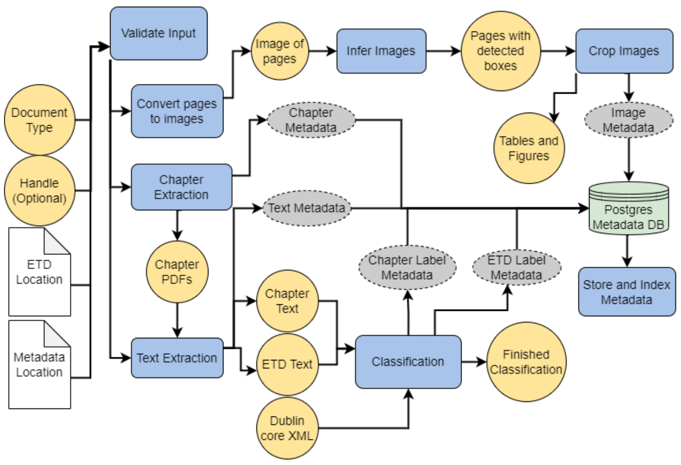


Figure 3: Workflow for ETD processing

runs on a container which not only hosts the web application that the curator uses, but also provides the APIs so other users can interact with the database and file system.

The Docker containers simultaneously support multiple personas, with whom they interact at different stages. Read and write performance is not affected when the other users are reading or inserting data like object images, chapter summaries, etc.

The web UI uses Flask, where curators access ETDs using the Flask framework in Python. It includes extended APIs for CRUD operations running on the container cluster. Figure 3 shows the workflow that curators use to add and process ETDs.

5 CORE SERVICES FOR RESEARCHERS

5.1 Segmentation, Classification, and Summarization

A student or a researcher wants to access a particular section of interest from an ETD. To do so, they will need more information about each of the chapters to determine whether the content is of interest.

Once a user uploads an ETD, the segmentation pipeline predicts the chapter boundaries and creates chapter PDFs. Users have used the segmentation pipeline to segment 5000 documents using the model described in Section 7.2.1. Segmented chapter PDFs of these 5000 documents are stored in the database and can be used by other services like summarization and classification. A PDF parsing and cleaning pipeline follows the segmentation pipeline. PDFPlumber [67] is used to extract text from the PDFs. To ensure that we only provide text to the downstream services, both tables and figures are removed from the extracted text.

Our segmentation work is based on [52]. The deep learning pipeline for segmentation uses both image and text features as sequence elements. A VGG model was used to extract image features, whereas Glove and FastText were used for text embedding. An LSTM was trained with extracted features from both text and images. Training and validation were done on 1459 ETDs (80/20 split). The testing was done on 150 ETDs. The test set consists of documents from both arXiv and non-arXiv sources. The model used

Table 3: Evaluation of classification models. Optimizer: Adam E; Learning Rate: 5e-5; Batch Size: 8; Epochs: 8

Model Name	Accuracy (%)	F-1 Score (%)	Chapter/ Summary
BERT-base	64.75	56.28	Chapter
SciBERT	80.77	77.5	Chapter
BERT-base	63.01	54.41	Summary
SciBERT	70.89	67	Summary

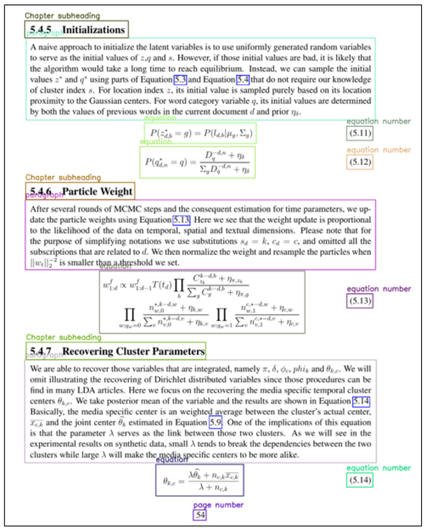
for segmentation predicts one of six labels. We look at the 'Chapter Start' label to detect chapter boundaries and segment the ETD into chapter PDFs. The 'Chapter Start' label has a F1 score of 72% with 83% Precision, and 77% Recall.

The classification service uses the extracted chapter text as its input. Our classification task was done using language models. Chapter classification was done using both full text and the chapter summary as the input. Some of the models used for the experiments are (1) Language model-based classifiers – BERT, SciBERT, and Longformer; and (2) Machine learning-based classifiers – SVM and Random Forest. 3742 chapter text was used to fine-tune the language models. For the classification task, a dataset comprising 27 different classes was chosen. The classification evaluation test set consisted of 749 chapters. The models are available to use by the experimenter as described in Section 6.1. The performance is depicted in Table 3. We conclude that SciBERT on chapter text has better overall accuracy and F1 scores for predicting classification labels. Thus, this model is used to generate classification labels on 5000 ETDs; these are stored in the database.

Similar to the classification service, the summarization pipeline takes the extracted chapter text to generate chapter summaries. We used various summarization models in the experimental setup of the task. Our summarization pipeline supports both abstractive (BigBird [84] pre-trained on the BookSum dataset) and extractive summarization (TextRank, LexRank, and LSA) models. In Section 6.1 we describe our results of implementing and testing these models. We ran the summarization pipeline on 5000 documents using TextRank. The service produces a list of chapter summaries, one for each chapter, which the DL subsequently stores in the database. The chapter summaries are available for use in tasks such as classification, search, and recommendation. Furthermore, the summarization service for experimenters supports all of the summarization models. Figure 5 (c) displays the document view page of an ETD with chapter summaries and classification labels.

5.2 Object Detection

The object detection module takes as input a PDF version of the document, and produces a parsed version in a structured XML format. It uses object detection models such as YOLO [79] that have been trained on a dataset consisting of ETD-specific elements [3, 4]. The PDF document is first split into individual page images, each of which is then fed to the object detection model for extracting elements on the corresponding pages. More specifically, after creating the page images from the ETDs using the PDF2image Python library, we train YOLOv7 [80] and Faster R-CNN (Detectron2 [83]) to extract objects such as the metadata, figures, tables, chapters,


Chapter subheading
paragraph
equation
equation number
equation
equation number
Chapter subheading
paragraph
equation
equation number
Chapter subheading
paragraph
equation
equation number
page number

5.4.5 Initializations

A naive approach to initialize the latent variables is to use uniformly generated random variables to serve as the initial values of z,q and s. However, if those initial values are bad, it is likely that the algorithm would take a long time to reach equilibrium. Instead, we can sample the initial values z^* and q^* using parts of Equation 5.3 and Equation 5.4 that do not require our knowledge of cluster index s. For location index z, its initial value is sampled purely based on its location proximity to the Gaussian centers. For word category variable q, its initial values are determined by both the values of previous words in the current document d and prior η_β.

$$P(z^*_{d,b} = g) = P(l_{d,b}|\mu_g, \Sigma_g) \quad (5.11)$$

$$P(q^*_{d,n} = q) = \frac{D_q^{-d,n} + \eta_\beta}{\Sigma_g D_q^{-d,n} + \eta_\beta} \quad (5.12)$$

5.4.6 Particle Weight

After several rounds of MCMC steps and the consequent estimation for time parameters, we update the particle weights using Equation 5.13. Here we see that the weight update is proportional to the likelihood of the data on temporal, spatial and textual dimensions. Please note that for the purpose of simplifying notations we use substitutions $s_d = k$, $c_d = c$, and omitted all the subscriptions that are related to d. We then normalize the weight and resample the particles when $||w_t||_2^{-2}$ is smaller than a threshold we set.

$$w^f_{1:d} \propto w^f_{1:d-1} T(t_d) \prod_b \frac{C^{k-d,b}_{z_b} + \eta_{\pi,z_b}}{\sum_g C^{k-d,b}_g + \eta_{\pi,g}} \prod_{w:q_w=0} \frac{n^{*,k-d,w}_{w,0} + \eta_{k,w}}{\sum_v n^{*,k-d,v}_{v,0} + \eta_{k,v}} \prod_{w:q_w=1} \frac{n^{c,*-d,w}_{w,1} + \eta_{c,w}}{\sum_v n^{c,*-d,v}_{v,1} + \eta_{c,v}} \quad (5.13)$$

5.4.7 Recovering Cluster Parameters

We are able to recover those variables that are integrated, namely π, δ, ϕ_c, phi_k and $\theta_{k,c}$. We will omit illustrating the recovering of Dirichlet distributed variables since those procedures can be find in many LDA articles. Here we focus on the recovering the media specific temporal cluster centers $\theta_{k,c}$. We take posterior mean of the variable and the results are shown in Equation 5.14. Basically, the media specific center is an weighted average between the cluster's actual center, $\overline{x_{c,k}}$ and the joint center $\widehat{\theta_k}$ estimated in Equation 5.9. One of the implications of this equation is that the parameter λ serves as the link between those two clusters. As we will see in the experimental results on synthetic data, small λ tends to break the dependencies between the two clusters while large λ will make the media specific centers to be more alike.

$$\theta_{k,c} = \frac{\lambda\widehat{\theta_k} + n_{c,k}\overline{x_{c,k}}}{\lambda + n_{c,k}} \quad (5.14)$$

54

Figure 4: Object detection example

titles, and paragraphs – yielding both image-based and text-based objects. The resultant XML document consists of different elements, which are broadly divided into three categories:

- **Front Matter:** includes metadata elements, e.g., document title, author name(s), degree, university, etc. Other elements give an overview of the document, such as the abstract and list/table of contents.
- **Document Body:** is the main part of the ETD, consisting of a list of chapters. Each contains respective sections, with their paragraphs, figures (and captions), tables (and captions), equations (with numbers), algorithms, and footnotes.
- **Back Matter:** includes references and appendices.

Figure 4 shows examples of detected objects (bounding boxes and object categories) from an ETD page. The content (i.e., text) from text-based objects, such as paragraphs, captions footnotes, etc. is further extracted using text extraction tools such as PyMuPDF[49] or OCR, before being populated in the XML.

5.3 Topic Modeling of ETDs and Chapters

Our framework allows users to select one of the topic modeling techniques, and the number of topics, and to get the resulting topics. This service includes three major components:

- **Documents per Topic Distribution**: This module helps users find the most popular topics in the document collection. Given a threshold value and a topic, this component calculates the number of documents in the database for which the given topic's probability exceeds a threshold.
- **Topic List**: For every topic, this module shows the top 10 words that are representative of that topic; the set thus serves as a type of label.
- **Similar Topics**: Some users work in interdisciplinary fields. In such instances, it is often desirable to show a list of related topics to the user. This is done based on similarities between different rows of the topic-word matrix.

In this study, we utilized the Python library OCTIS [73] to train several models, including Latent Dirichlet Allocation (LDA) [13], Neural LDA [69], Product of Latent Dirichlet Allocation (ProdLDA) [69], and Correlated Topic Model (CTM) [11], on a dataset of ETDs.

5.4 Search and Recommendation

The **search** service is responsible for indexing and searching over all the ETD metadata, figures, chapters, and other digital objects. A researcher wants to search and browse over a collection of ETDs and their elements, across various universities and disciplines, and be able to view a list of ETDs ranked for relevance to the user query. The service runs as a container and interacts with Front-end, ElasticSearch, and Database containers, as shown in Figure 1. The service queries the database using the APIs provided by the curator team, and indexes the ETDs and other digital objects into ElasticSearch. Flask [16], a micro-web framework, is used to create an API endpoint to receive queries from the front-end. We then create a search query and use an ElasticSearch client to search over the ElasticSearch index. Conventional methods like inverted index and ML based models like kNN are considered and implemented to improve the search. We use different metadata fields like author, university, major, etc. to sort and filter the search results. By default we sort the documents based on the estimated relevance score provided by ElasticSearch [14], as was done by a prior class team [45]. The search service logs the user queries and demographics, and makes it available for the recommendation engine to provide user-based recommendations.

We index the following items into ElasticSearch: (1) ETD metadata which has text fields like author, abstract, title, etc. (2) Chapter summaries, classification labels, and other chapter-related objects.

Figure 5 (b) and 5 (d) display screenshots of the UI for our DL system for search. We implement the following search methods to search over ETD metadata and chapters.

- ElasticSearch's default method, which uses keyword matching to search through the documents.
- kNN search, which finds the k nearest vectors to a query vector, as measured by a similarity metric. The indexed documents consist of a field of type dense vector.
- A hybrid search method that performs kNN and keyword-based search independently, and then returns top results based on the combined score (e.g., $0.9 * match\_score + 0.1 * knn\_score$).

The **recommendation** module recommends similar ETDs as ETDs of potential interest to the user. A researcher expects a personalized experience from a DL system, and the recommender model provides that by displaying top-n documents. This is shown in Figure 5 (a) based on user interests and interactions. The idea is to use the user click history as a form of feedback to our recommendation system. Click history refers to the log of the ETDs that the user has clicked on. We use user click events to log user data, and generate a dataset of users and their associated ETDs. This dataset is then used to generate recommendations by training a machine learning model for the same. For a registered user, we provide a fine-grained recommendation by leveraging that user's interactions and preferences.

The data that is used to build a dataset to train the model includes: (1) User Interaction, (2) Clicks on ETD links, and (3) User Search History (i.e., extracting keywords from each user search query and mapping them to a topic). We resolve the cold start problem by asking the user upfront about the topics they are interested in. These topics can be the very same topics produced as a result of topic modeling.

We based our model on the Deep Learning Recommendation Model (DLRM) [56]. It is a hybrid implementation of content and collaborative filtering. The DLRM model forms the global model. Recommendations at a user level need to change fairly quickly based on user behavior and usage patterns. It would be difficult to fine-tune the global model for each user. Hence, we follow [1] to train a simple logistic regression model for each user. This model provides a click probability per ETD for each user and is then added to that of the global model to give us the best of both worlds. We achieve global knowledge of recommendations through the larger model and user-level personalization through the smaller model.

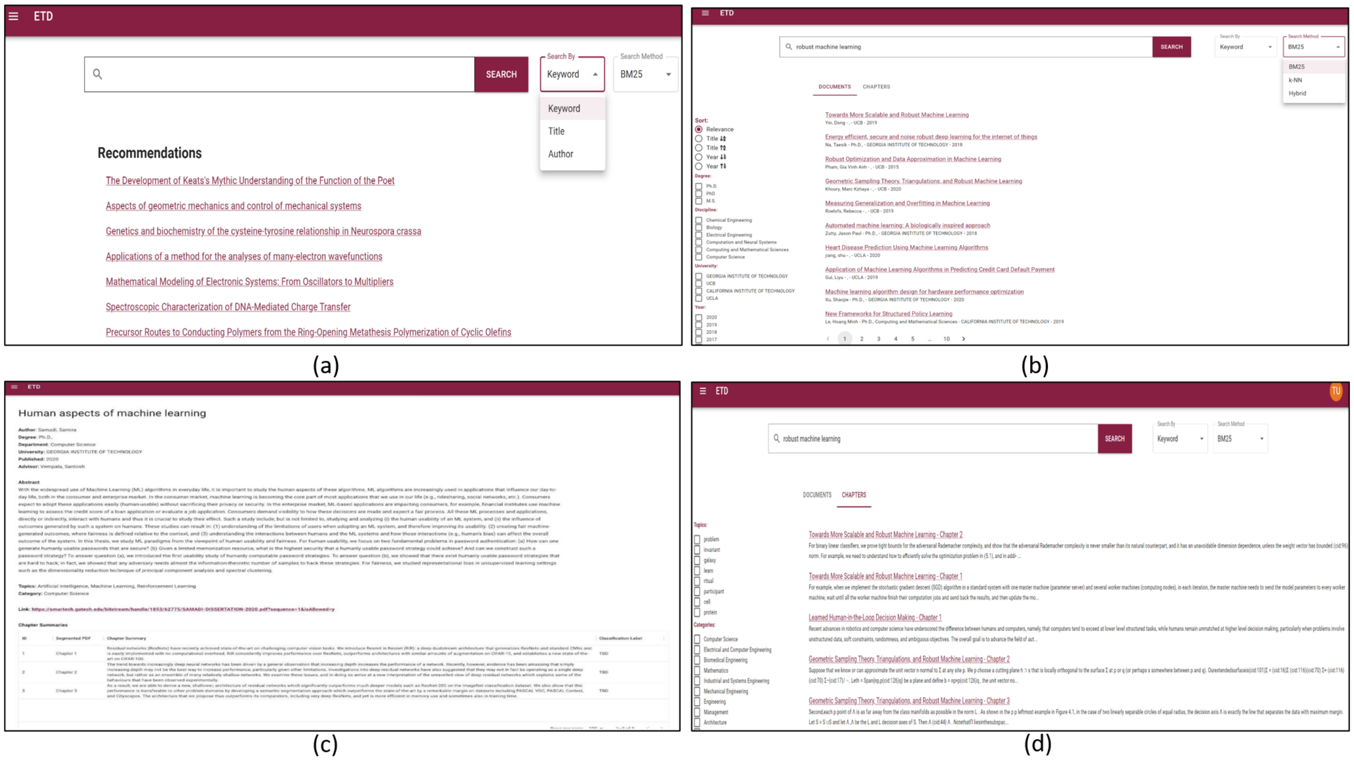
The integrated recommendation model is exposed as the recommendation service through the infer and train API. The recommendation service interacts with the front-end and ElasticSearch index. As more people use and interact with the system, the data distribution of the interaction logs changes. In order to deliver relevant recommendations to the user, the model is updated with changes in the data. To facilitate easy updates to the model, we provide a train API that can be called at a set frequency by the ElasticSearch service.

6 CORE SERVICES FOR EXPERIMENTERS

6.1 Segmentation, Classification, and Summarization

The experimenter page as shown in Figure 6 (a) is created to help developers and researchers plug in different models and check the performance. For the experimenter UI interface, segmentation, classification, and summarization services are performed in succession. The user has the ability to upload the ETD. This page enables the user to select the desired summarization and classification models from a dropdown menu. For the classification task, fine-tuned SciBERT, BERT, and Longformer models are available to the user. For summarization, the models currently available are TextRank, BART, and BigBird. Once a user has made their selection and submitted the form, an API call triggers the workflow to run each of the services. Then the ETD is processed as follows.

(1) PDFs are segmented into chapters.
(2) Chapter PDFs are stored in the database.
(3) Text is extracted and cleaned from the PDFs and stored as text files.
(4) Extracted text is fed into the summarization and classification pipelines with the desired models as arguments.
(5) Chapter classification labels and summaries are retrieved from respective pipelines and displayed to the user in the UI.



ETD

SEARCH

Search By: Keyword

Search Method: BM25

Keyword
Title
Author

Recommendations

The Development of Keats's Mythic Understanding of the Function of the Poet
Aspects of geometric mechanics and control of mechanical systems
Genetics and biochemistry of the cysteine-tyrosine relationship in Neurospora crassa
Applications of a method for the analyses of many-electron wavefunctions
Mathematical Modeling of Electronic Systems: From Oscillators to Multipliers
Spectroscopic Characterization of DNA-Mediated Charge Transfer
Precursor Routes to Conducting Polymers from the Ring-Opening Metathesis Polymerization of Cyclic Olefins

ETD

robust machine learning

SEARCH

Search By: Keyword

Search Method: BM25

BM25
k-NN
Hybrid

DOCUMENTS CHAPTERS

Sort:
[✓ Relevance]
Title ⇅
Title ⇅
Year ⇅
Year ⇅

Degree:
Ph.D.
PhD
M.S.

Discipline:
Chemical Engineering
Biology
Electrical Engineering
Computation and Neural Systems
Computing and Mathematical Sciences
Computer Science

University:
GEORGIA INSTITUTE OF TECHNOLOGY
UCB
CALIFORNIA INSTITUTE OF TECHNOLOGY
UCLA

Year:
2020
2019
2018
2017

Towards More Scalable and Robust Machine Learning
Yin, Dong - , - UCB - 2019
Energy efficient, secure and noise robust deep learning for the internet of things
Na, Taesik - Ph.D., - GEORGIA INSTITUTE OF TECHNOLOGY - 2018
Robust Optimization and Data Approximation in Machine Learning
Pham, Gia Vinh Anh - , - UCB - 2015
Geometric Sampling Theory, Triangulations, and Robust Machine Learning
Khoury, Marc Kzhaya - , - UCB - 2020
Measuring Generalization and Overfitting in Machine Learning
Roelofs, Rebecca - , - UCB - 2019
Automated machine learning: A biologically inspired approach
Zutty, Jason Paul - Ph.D., - GEORGIA INSTITUTE OF TECHNOLOGY - 2018
Heart Disease Prediction Using Machine Learning Algorithms
jiang, shu - , - UCLA - 2020
Application of Machine Learning Algorithms in Predicting Credit Card Default Payment
Gui, Liyu - , - UCLA - 2019
Machine learning algorithm design for hardware performance optimization
Xu, Shaojie - Ph.D., - GEORGIA INSTITUTE OF TECHNOLOGY - 2020
New Frameworks for Structured Policy Learning
Le, Hoang Minh - Ph.D., Computing and Mathematical Sciences - CALIFORNIA INSTITUTE OF TECHNOLOGY - 2019

1 2 3 4 5 ... 10

ETD

Human aspects of machine learning

Author: Samadi, Samira
Degree: Ph.D.,
Department: Computer Science
University: GEORGIA INSTITUTE OF TECHNOLOGY
Published: 2020
Advisor: Vempala, Santosh

Abstract
With the widespread use of Machine Learning (ML) algorithms in everyday life, it is important to study the human aspects of these algorithms. ML algorithms are increasingly used in applications that influence our day-to-day life, both in the consumer and enterprise market. In the consumer market, machine learning is becoming the core part of most applications that we use in our life (e.g., ridesharing, social networks, etc.). Consumers expect to adopt these applications easily (human-usable) without sacrificing their privacy or security. In the enterprise market, ML-based applications are impacting consumers, for example, financial institutes use machine learning to assess the credit score of a loan application or evaluate a job application. Consumers demand visibility to how these decisions are made and expect a fair process. All these ML processes and applications, directly or indirectly, interact with humans and thus it is crucial to study their effect. Such a study include, but is not limited to, studying and analyzing (i) the human usability of an ML system, and (ii) the influence of outcomes generated by such a system on humans. These studies can result in: (1) understanding of the limitations of users when adopting an ML system, and therefore improving its usability. (2) creating fair machine-generated outcomes, where fairness is defined relative to the context, and (3) understanding the interactions between humans and the ML systems and how those interactions (e.g., human's bias) can affect the overall outcome of the system. In this thesis, we study ML paradigms from the viewpoint of human usability and fairness. For human usability, we focus on two fundamental problems in password authentication: (a) How can one generate humanly usable passwords that are secure? (b) Given a limited memorization resource, what is the highest security that a humanly usable password strategy could achieve? And can we construct such a password strategy? To answer question (a), we introduced the first usability study of humanly computable password strategies. To answer question (b), we showed that there exist humanly usable password strategies that are hard to hack; in fact, we showed that any adversary needs almost the information-theoretic number of samples to hack these strategies. For fairness, we studied representational bias in unsupervised learning settings such as the dimensionality reduction technique of principal component analysis and spectral clustering.

Topics: Artificial Intelligence, Machine Learning, Reinforcement Learning
Category: Computer Science

Link: https://smartech.gatech.edu/bitstream/handle/1853/62775/SAMADI-DISSERTATION-2020.pdf?sequence=1&isAllowed=y

Chapter Summaries

ID	Segmented PDF	Chapter Summary	Classification Label
1	Chapter 1	Residual networks (ResNets) have recently achieved state-of-the-art on challenging computer vision tasks. We introduce Resnet in Resnet (RiR): a deep dualstream architecture that generalizes ResNets and standard CNNs and is easily implemented with no computational overhead. RiR consistently improves performance over ResNets, outperforms architectures with similar amounts of augmentation on CIFAR-10, and establishes a new state-of-the-art on CIFAR-100.	TBD
2	Chapter 2	The trend towards increasingly deep neural networks has been driven by a general observation that increasing depth increases the performance of a network. Recently, however, evidence has been amassing that simply increasing depth may not be the best way to increase performance, particularly given other limitations. Investigations into deep residual networks have also suggested that they may not in fact be operating as a single deep network, but rather as an ensemble of many relatively shallow networks. We examine these issues, and in doing so arrive at a new interpretation of the unravelled view of deep residual networks which explains some of the behaviours that have been observed experimentally.	TBD
3	Chapter 3	As a result, we are able to derive a new, shallower, architecture of residual networks which significantly outperforms much deeper models such as ResNet-200 on the ImageNet classification dataset. We also show that this performance is transferable to other problem domains by developing a semantic segmentation approach which outperforms the state-of-the-art by a remarkable margin on datasets including PASCAL VOC, PASCAL Context, and Cityscapes. The architecture that we propose thus outperforms its comparators, including very deep ResNets, and yet is more efficient in memory use and sometimes also in training time.	TBD

ETD

robust machine learning

SEARCH

Search By: Keyword

Search Method: BM25

DOCUMENTS CHAPTERS

Topics:
problem
invariant
galaxy
learn
ritual
participant
cell
protein

Categories:
Computer Science
Electrical and Computer Engineering
Biomedical Engineering
Mathematics
Industrial and Systems Engineering
Mechanical Engineering
Engineering
Management
Architecture

Towards More Scalable and Robust Machine Learning - Chapter 2
For binary linear classifiers, we prove tight bounds for the adversarial Rademacher complexity, and show that the adversarial Rademacher complexity is never smaller than its natural counterpart, and it has an unavoidable dimension dependence, unless the weight vector has bounded (cid:96) norm. For example, we need to understand how to efficiently solve the optimization problem in (5.1), and in addi- ...

Towards More Scalable and Robust Machine Learning - Chapter 1
For example, when we implement the stochastic gradient descent (SGD) algorithm in a standard system with one master machine (parameter server) and several worker machines (computing nodes), in each iteration, the master machine needs to send the model parameters to every worker machine, wait until all the worker machine finish their computation jobs and send back the results, and then update the mo...

Learned Human-in-the-Loop Decision Making - Chapter 1
Recent advances in robotics and computer science have underscored the difference between humans and computers, namely, that computers tend to exceed at lower level structured tasks, while humans remain unmatched at higher level decision making, particularly when problems involve unstructured data, soft constraints, randomness, and ambiguous objectives. The overall goal is to advance the field of aut...

Geometric Sampling Theory, Triangulations, and Robust Machine Learning - Chapter 2
Suppose that we know or can approximate the unit vector n normal to Σ at any site p. We p choose a cutting plane h ⊃ s that is locally orthogonal to the surface Σ at p or q (or perhaps s somewhere between p and q). Ourextendedsurfaceis(cid:101)Σ = (cid:16)Σ (cid:116)(cid:70) Σ+ (cid:116) (cid:70) Σ−(cid:17)/ ∼. Leth = Span(np,p(cid:126)q) be a plane and define b = np×p(cid:126)q , the unit vector no...

Geometric Sampling Theory, Triangulations, and Robust Machine Learning - Chapter 3
Second,each p point of Λ is as far away from the class manifolds as possible in the norm L . As shown in the p p leftmost example in Figure 4.1, in the case of two linearly separable circles of equal radius, the decision axis Λ is exactly the line that separates the data with maximum margin. Let S = S ∪S and let Λ ,Λ be the L and L decision axes of S. Then Λ (cid:44) Λ . NotethatΠ liesinthesubspac...

Figure 5: (a) Home page with personalized recommendations; (b) Search page; (c) Document view page; (d) Chapter search page.

6.2 Object Detection

The Experimenter page offers the ability to detect and display objects and topics from ETDs using various models. It includes a model selector supporting both Detectron2 and YOLOv7 with customization options for model weights and hyper-parameters. Accessible via the login sidebar menu under "Object Detection," the experimenter can choose a model and be redirected to the ETD view page. The detection results are displayed in HTML format generated from the object detection's XML output. To showcase the detection results of different models on an ETD, a Flask application was developed to produce an HTML page from the object detection model. The user selects between Detectron2 and YOLOv7, then uploads the desired ETD PDF for detection. The HTML page displays the detected chapters, figures, and linked captions extracted from the ETD in an organized manner.

6.3 Topic Modeling of ETDs and Chapters

Our Experimenter page for topic modeling currently offers three types of services. The ETD Embedding API accepts the model name, number of topics, and ETD ID as input. It returns a topic vector that represents the probabilities of the document being associated with each of the k topics generated. The ETD Related Documents API, given the model name, number of topics, and ETD ID, outputs the most relevant documents, with the default value for the optional parameter "top k" being 5 (related documents). The ETD Related Topics API, with the same input parameters as the previous API, returns the most related topics, with a default value for the "top k" parameter of 5 (related topics).

6.4 Search

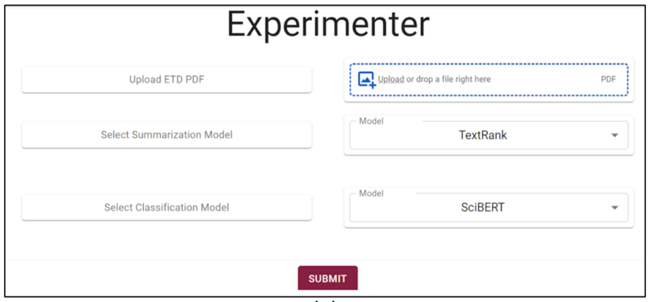
Apart from the default behavior exposed to the users to search over the existing ETDs, we provide an interface as shown in Figure 6 (b) for the experimenter to create and run search-related experiments. These experiments will allow the experimenter to index custom vectors for each ETD. Once such data is created, the experimenter can perform a hybrid search (kNN + keyword) on the indexed documents and check the scores for the documents returned. The flow for creating and running an experiment is shown in Figure 7. These experiments are user specific, and the created experiments' metadata is stored in an ElasticSearch index.

We also provide four default experiments accessible to all users. These involve text embeddings created from the abstract and title of 1000 ETDs. The models used to generate embeddings for the default experiments are: (1) all-distilroberta-v1 [74], (2) all-mpnet-base-v2 [77], (3) all-MiniLM-L12-v2 [76], and (4) LaBSE [75].

7 BACKGROUND AND RELATED WORK

7.1 Workflow-centric Information Systesm

The evolution of workflow management systems (WMSs) has been a natural consequence of advances in computer technology, an increase in digital sensors, and as a by-product, an increase in the volume of observational data and any data collected through



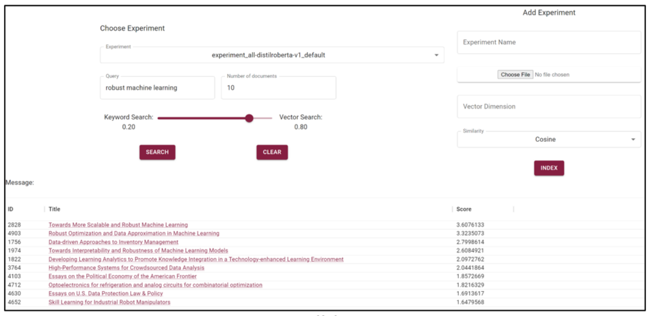
Experimenter

Upload ETD PDF — Upload or drop a file right here PDF

Select Summarization Model — Model: TextRank

Select Classification Model — Model: SciBERT

(a)



Choose Experiment

Experiment: experiment_all-distilroberta-v1_default

Query: robust machine learning

Number of documents: 10

Keyword Search: 0.20 — Vector Search: 0.80

Add Experiment

Experiment Name

Choose File No file chosen

Vector Dimension

Similarity: Cosine

Message:

ID	Title	Score
2828	Towards More Scalable and Robust Machine Learning	3.6076133
4903	Robust Optimization and Data Approximation in Machine Learning	3.3235073
1756	Data-driven Approaches to Inventory Management	2.7998614
1974	Towards Interpretability and Robustness of Machine Learning Models	2.6084921
1822	Developing Learning Analytics to Promote Knowledge Integration in a Technology-enhanced Learning Environment	2.0972762
3764	High-Performance Systems for Crowdsourced Data Analysis	2.0441864
4103	Essays on the Political Economy of the American Frontier	1.8572669
4712	Optoelectronics for refrigeration and analog circuits for combinatorial optimization	1.8216329
4630	Essays on U.S. Data Protection Law & Policy	1.6913617
4652	Skill Learning for Industrial Robot Manipulators	1.6479568

(b)

Figure 6: Experimenter UI: (a) Classification and Summarization; (b) Search.

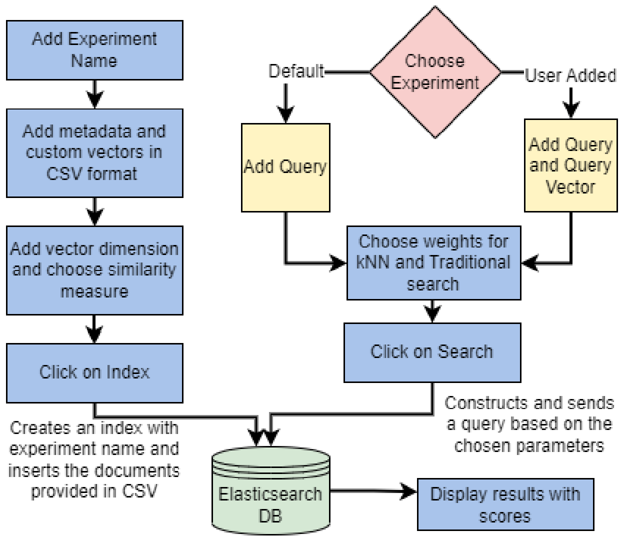

Add Experiment Name
Add metadata and custom vectors in CSV format
Add vector dimension and choose similarity measure
Click on Index
Creates an index with experiment name and inserts the documents provided in CSV
Choose Experiment
Default
User Added
Add Query
Add Query and Query Vector
Choose weights for kNN and Traditional search
Click on Search
Constructs and sends a query based on the chosen parameters
Elasticsearch DB
Display results with scores

Figure 7: Experimenter interactions with ElasticSearch

automation. Jim Gray, in the Fourth Paradigm, recommends that the scientific community foster digital data libraries for both data and literature [34].

Liew et al. state that WMSs: (a) support collaborative research, (b) construct workflows without concern about resources and workflow execution, (c) automate steps for reproducibility and simulation-based studies, (d) integrate resources (data and processing power) from heterogeneous sources, and (e) optimize workflow execution [30, 32, 46, 72].

Many WMSs exist, each with varying characteristics. Pegasus is a popular WMS [18]. It uses Wings for workflow composition and provenance tracking [27]. Like Pegasus, Apache Taverna is an open-source WMS [82]. Similarly, there are many more powerful WMSs such as KNIME, Kepler, Galaxy, etc. [10, 12, 51]. One of the main aims of these WMSs is to help the users conduct their experiments without having knowledge of workflow execution and optimization. However, these interfaces are not particularly helpful for an audience of SMEs who might have little to no information on the tasks and files required. That knowledge barrier is not addressed by the WMSs mentioned above.

7.2 DL Services Background

The traditional approach to very long document IR is to simply treat them like other documents, and just help find an entire long document. This is what is done by WWW search engines, library catalogs, and scholarly oriented systems [5, 28, 31]. One of the early approaches to improve access to the content of very long documents aimed to extract relevant passages [41]. Another approach is to apply text mining, initially to aid with discovery [36, 85]. Recently, there have been several studies of the use of deep learning related to paper-length documents, including with transformers and attention models [9, 35, 44, 78, 84]. Our models apply those and our own findings, and scale them up for very long documents.

7.2.1 Segmentation, Classification, and Summarization: Long documents are often organized into chapters and sections. Automatic **segmentation** of ETDs is used to identify their chapters. These chapters are then used for chapter-level classification, summarization, and subsequent chapter searches and recommendations. The variation in format, writing structure, and styles make it hard to automatically detect chapter boundaries, which was the result when using PyMuPDF [49] and GROBID [50].

In this work, we use a segmentation model [52] that leverages LaTeX documents to train a segmentation model to predict the chapter boundaries of ETDs. LaTeX sources (some available on arXiv [1]) contain chapter and section tags that help in boundary detection. The model uses both page images and textual information to detect chapter boundaries.

University repositories hosting ETDs come with metadata information. Department or discipline information is often included in the metadata. Research has become interdisciplinary in nature and the document-level **classification** labels often are not indicative of such collaboration. We perform chapter-level classification, and this work is performing multi-label classification. In [38], the author leveraged 28 ProQuest subject categories to multi-label classify the chapters. Language models have also been used for the classification task. BERT [20] and SciBERT [8] are some popular language models that uses transformers to perform various sequence tasks including classification. To obtain more tailored results, we can fine-tune base models [7]. We use language models for our chapter classification work.

[1]https://arxiv.org/

Text **summarization** helps with succinctly conveying the information in a large body of text. Automatic text summarization has become especially popular in recent years with the growth in numbers of digital documents. Text summaries not only reduce reading time but also improve the effectiveness of search and retrieval. Extractive summarization techniques use sentences from the input text in the generated summary. Some popular models are TextRank, LexRank, and LSA. On the other hand, abstractive summaries resemble human summaization more closely. Most popular summarization techniques like the 'Abstractive Text Summarization using Sequence-to-sequence RNNs' [55] and 'Pointer Generator' [65]. Language models have also been used for the summarization task, like BART [43], T5 [61], and Big Bird [84] While most popular language models only work with shorter sequences, the Longformer Encoder Decoder (LED) [9] incorporates a longer sequence length to help summarize longer documents.

7.2.2 Object Detection: We evaluate our object detection pipeline using two models – Faster R-CNN and YOLOv7. Faster R-CNN [62] is the third iteration of the R-CNN architecture [29]. Its evolution from the VGG-16 backbone included Region of Interest (RoI) Pooling, a Region Proposal Network (RPN), and Facebook AI Research's Detectron2 [83]. YOLOv7 has the highest accuracy [80]. It outperforms both transformer-based object detectors and convolution-based object detectors in real-time.

7.2.3 Topic Modeling: Topic modeling of ETDs aims to extract thematic collections of words that represent topics. Early works such as LDA [13] were based on a probabilistic formulation of topics. Recent works include neural topic models such as NeuralLDA [69], ProdLDA [69], and CTM [11]. The representations learned from topic models can be used for downstream tasks that rely on document representations, such as finding similar documents (document recommendation) and finding similar topics.

7.2.4 Search, Browsing, and Recommendation: To aid browsing, as well as searching, which are the core DL services, search engines such as Solr and ElasticSearch (ES) which are based on Lucene are commonly employed. Elasticsearch [6, 15, 40] can power extremely fast searches that will be beneficial for the search engine we are trying to build around the ETDs. Also, with recent support for embeddings in the ES v8.x search engine, ES has provided a platform to implement ML-based search methods.

User-level recommendation is one of the essential DL services. It provides user's with similar document recommendations based on their interest's and search history. Recommendation models are divided into two types: (1) content-based filtering which consists of search-based methods [68] and cluster-based methods. (2) collaborative filtering which consists of bag-of-items, traditional user-based methods, item-based filtering [48], and Sequential Transformers.

Steck et al. [70] describe multiple examples of how these recommendations are brought to life. In the traditional bag of items approach, the order in which the items gained association with the user is not maintained nor given the same significance with respect to the recommendation engine. This assumption does not hold with DL recommendation engines, as recently viewed/read content holds more significance and thus more weight during recommendation.

In item-to-item collaborative filtering [48], the algorithm matches every purchased and rated items of the user to similar items, then combines those similar items into a list of recommendations. Naumov et al. [56] propose a deep learning model for recommendations which is a hybrid filtering method that ranks based on the Click-Through-Rate to show recommendations to the user.

Aberdeen et al. [1] introduce an idea to implement personalized recommendations accurately and at scale, and to incorporate user feedback continuously. We see that in a real-world use case, user behaviour varies vastly and it is important to provide recommendations catering to the user, and to avoid generalizations.

8 CONCLUSION AND FUTURE WORK

This paper describes the framework, components, and key personas of our integrated DL, built to support mining and exploration of long documents and their elements. Personas, through different scenarios, can satisfy their information goals through our UI and workflow/service-centric architecture.

Services, built on state-of-the-art models, identify/generate syntactic/semantic elements. 57,129 ETDs' metadata and abstract were extracted, stored, indexed, and made available for search and browsing. We have used 5K of the 57K to generate/detect elements. Through object detection, we have identified approximately 550K image-based objects. Through segmentation, we have identified approximately 22.8K chapters from 5K ETDs. Chapter texts support classification and summarization. Through topic modeling, we have assigned topics for the 5K ETDs and 22.8K chapters, which in turn were used to provide a list of "top-10" similar ETDs/chapters based on topics. The generated elements for the 5K ETDs were indexed and made available for searching, browsing, and recommendation.

We will scale up our efforts from 5K to much larger numbers. We will improve our models for existing services, while developing new services, e.g., searching for figure and table captions. Through additional analysis and services, and related evaluations and user studies, we will better support the information goals of our diverse stakeholders.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants OAC-1835660, CMMI-1638207, and IIS-1619028; by IMLS grant LG-37-19-0078-19; and by Virginia Tech through CS and University Libraries. We also acknowledge the students in the Fall 2022 CS5604 class [19, 26, 37, 57, 66], and additional mentors who assisted that class (Sara Ahmadi and Dhanush Dinesh).

REFERENCES

[1] Douglas Aberdeen, Ondrey Pacovsky, and Andrew Slater. 2010. The Learning Behind Gmail Priority Inbox. In *LCCC: NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds.*
[2] E. Agichtein and V. Ganti. 2004. Mining reference tables for automatic text segmentation. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 20–29.
[3] Aman Ahuja, Alan Devera, and Edward Alan Fox. 2022. Parsing Electronic Theses and Dissertations Using Object Detection. In *Proceedings of the First Workshop on Information Extraction from Scientific Publications.* 121–130. https://aclanthology.org/2022.wiesp-1.14/.
[4] Aman Ahuja, Kevin Dinh, Brian Dinh, William A. Ingram, and Edward Alan Fox. 2023. A New Annotation Method and Dataset for Layout Analysis of Long Documents. In *Companion Proceedings of the ACM Web Conference 2023*

(WWW '23 Companion), April 30-May 4, 2023, Austin, TX, USA. ACM. https://doi.org/10.1145/3543873.3587609

[5] AllenAI. 2022. Semantic Scholar. https://www.semanticscholar.org/

[6] Abhishek Andhavarapu. 2017. *Learning Elasticsearch* (first ed.). Packt Publishing.

[7] Bipasha Banerjee, William A. Ingram, Jian Wu, and Edward A. Fox. 2022. Applications of data analysis on scholarly long documents. In *2022 IEEE International Conference on Big Data (Big Data)*. 2473–2481. https://doi.org/10.1109/BigData55660.2022.10020935

[8] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proc. EMNLP-IJCNLP*. ACL, Hong Kong, 3615–3620. https://doi.org/10.18653/v1/D19-1371

[9] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The Long-Document Transformer. *arXiv:2004.05150 [cs]* (Dec. 2020).

[10] Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Kilian Thiel, and Bernd Wiswedel. 2009. KNIME - the Konstanz Information Miner: Version 2.0 and Beyond. *SIGKDD Explor. Newsl.* 11, 1 (Nov. 2009), 26–31. https://doi.org/10.1145/1656274.1656280

[11] Federico Bianchi, Silvia Terragni, and Dirk Hovy. 2020. Pre-training is a hot topic: Contextualized document embeddings improve topic coherence. *arXiv preprint arXiv:2004.03974* (2020).

[12] Daniel Blankenberg, Gregory Von Kuster, Nathaniel Coraor, Guruprasad Ananda, Ross Lazarus, Mary Mangan, Anton Nekrutenko, and James Taylor. 2010. Galaxy: A Web-Based Genome Analysis Tool for Experimentalists. *Current Protocols in Molecular Biology* 89, 1 (2010), 19.10.1–19.10.21. https://doi.org/10.1002/0471142727.mb1910s89

[13] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[14] Elasticsearch B.V. 2019. Elasticsearch Scoring. https://www.compose.com/articles/how-scoring-works-in-elasticsearch/ accessed on September 15, 2022.

[15] Elasticsearch B.V. 2019. Open Source Search: The Creators of Elasticsearch, ELK Stack & Kibana | Elastic. https://www.elastic.co/ accessed on August 31, 2022.

[16] Flask B.V. 2020. Flask Website. https://flask.palletsprojects.com/en/2.2.x/ accessed on August 31, 2022.

[17] C.C. Chen, K.H. Yang, H.Y. Kao, and J.M. Ho. 2008. BibPro: A Citation parser based on sequence alignment techniques. In *22nd International Conference on Advanced Information Networking and Applications-Workshops*. IEEE, 1175–1180.

[18] Ewa Deelman, Karan Vahi, Mats Rynge, Gideon Juve, Rajiv Mayani, and Rafael Ferreira da Silva. 2016. Pegasus in the Cloud: Science Automation through Workflow Technologies. *IEEE Internet Computing* 20, 1 (Jan. 2016), 70–76. https://doi.org/10.1109/MIC.2016.15

[19] Alan Devera, Raj Sahu, Nila Masrourisaadat, Nirmal Amirthalingam, and Chenyu Mao. 2022. *CS 5604 Fall 2022 Team 3: Object Detection and Topic Modeling.* CS5604 team term project. Virginia Tech. http://hdl.handle.net/10919/114081 accessed on April 10, 2023.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[21] Hugging Face. 2022. Sentence Transformers. (2022). https://huggingface.co/sentence-transformers accessed on November 19, 2022.

[22] Edward A. Fox. 2023. CS5604 System Reproducibility. https://fox.cs.vt.edu/CS5604SystemReproducibility.pdf accessed on April 16, 2023.

[23] Edward A. Fox and Prashant Chandrasekar. 2021. How Should One Explore the Digital Library of the Future? *Data and Information Management* 5, 4 (2021), 349–362. https://doi.org/doi:10.2478/dim-2021-0003

[24] Edward A. Fox, Marcos André Gonçalves, and Rao Shen. 2012. *Theoretical Foundations for Digital Libraries: The 5S (Societies, Scenarios, Spaces, Structures, Streams) Approach.* Morgan & Claypool Pub. DOI 10.2200/S00434ED1V01Y201207ICR022.

[25] Norbert Fuhr and Kai Grobjohann. 2004. XIRQL - An XML Query Language based on Information Retrieval Concepts. *ACM Transactions on Information Systems* 22, 2 (April 2004), 313 – 356. http://doi.acm.org/10.1145/984321.984326

[26] Kaushik Ganesan, Deepak Nanjundan, Deval Srivastava, Abhilash Neog, Dharneeshkar Jayaprakash, and Aditya Shah. 2022. *CS 5604 Fall 2022 Team 4: Segmentation, Summarization, and Classification.* CS5604 team term project. Virginia Tech. http://hdl.handle.net/10919/114077 accessed on April 10, 2023.

[27] Y. Gil, V. Ratnakar, J. Kim, P. Gonzalez-Calero, P. Groth, J. Moody, and E. Deelman. 2011. Wings: Intelligent Workflow-Based Design of Computational Experiments. *IEEE Intelligent Systems* 26, 1 (Jan 2011), 62–72. https://doi.org/10.1109/MIS.2010.9

[28] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. CiteSeer: An Automatic Citation Indexing System. In *Proc. 3rd ACM Int'l Conf. Digital Libraries, June 23-26, 1998, Pittsburgh*. 89–98. https://doi.org/10.1145/276675.276685

[29] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. IEEE conf. on computer vision and pattern recognition*. 580–587.

[30] Carole Anne Goble and David Charles De Roure. 2007. MyExperiment: Social Networking for Workflow-Using e-Scientists. In *Proceedings of the 2nd Workshop on Workflows in Support of Large-Scale Science* (Monterey, California, USA) *(WORKS '07)*. Association for Computing Machinery, New York, NY, USA, 1–2. https://doi.org/10.1145/1273360.1273361

[31] Google. 2022. Google Scholar. https://scholar.google.com/

[32] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. 2011. *Conventional Workflow Technology for Scientific Simulation.* Springer London, 323–352. https://doi.org/10.1007/978-0-85729-439-5_12

[33] G. Gou and R. Chirkova. 2007. Efficiently Querying Large XML Data Repositories: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 10 (2007), 1381–1403. https://doi.org/10.1109/TKDE.2007.1060

[34] Tony Hey, Stewart Tansley, and Kristin Tolle. 2009. *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research. https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/

[35] Luyang Huang, Shuyang Cao, Nikolaus Nova Parulian, Heng Ji, and Lu Wang. 2021. Efficient Attentions for Long Document Summarization. *CoRR* abs/2104.02112 (2021). arXiv:2104.02112 https://arxiv.org/abs/2104.02112

[36] Olexandr Isayev. 2019. Text mining facilitates materials discovery. *Nature* 571, 7763 (July 2019), 42–43. https://doi.org/10.1038/d41586-019-01978-x

[37] Tanya Jain, Hirva Bhagat, Wen-Yu Lee, Ashrith Reddy Thukkaraju, and Raghav Sethi. 2022. *CS 5604 Fall 2022 Team 1: ETD Collection Management.* CS5604 team term project. Virginia Tech. http://hdl.handle.net/10919/114079 accessed on April 10, 2023.

[38] Palakh Mignonne Jude. 2020. *Increasing Accessibility of Electronic Theses and Dissertations (ETDs) Through Chapter-level Classification.* MS Thesis. Virginia Tech. http://hdl.handle.net/10919/99294

[39] Sampanna Yashwant Kahu, William A Ingram, Edward A Fox, and Jian Wu. 2021. ScanBank: A Benchmark Dataset for Figure Extraction from Scanned Electronic Theses and Dissertations. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. 180–191. https://doi.org/10.1109/JCDL52503.2021.00030

[40] Nikita Kathare, O. Vinati Reddy, and Vishalakshi Prabhu. 2020. A Comprehensive Study of Elasticsearch. *International Journal of Science and Research (IJSR)* (2020).

[41] Jinhyuk Lee, Alexander Wettig, and Danqi Chen. 2021. Phrase Retrieval Learns Passage Retrieval, Too. *arXiv preprint arXiv:2109.08133* (2021).

[42] Kyong H. Lee, Yoon C. Choy, and Sung B. Cho. 2003. Logical Structure Analysis and Generation for Structured Documents: A Syntactic Approach. *IEEE Trans. on Knowl. and Data Eng.* 15, 5 (2003), 1277–1294. https://doi.org/10.1109/TKDE.2003.1232278

[43] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.* Technical Report. http://arxiv.org/abs/1910.13461 arXiv:1910.13461 [cs, stat].

[44] Minghan Li, Diana Nicoleta Popa, Johan Chagnon, Yagmur Gizem Cinar, and Éric Gaussier. 2021. The Power of Selecting Key Blocks with Local Pre-ranking for Long Document Information Retrieval. *CoRR* abs/2111.09852 (2021). arXiv:2111.09852 https://arxiv.org/abs/2111.09852

[45] Yuan Li, Satvik Chekuri, Tianrui Hu, Soumya Arvind Kumar, and Nicholas Gill. 2019. *CS 5604 Fall 2019 ELS.* CS5604 team term project. Virginia Tech. http://hdl.handle.net/10919/96310 accessed on September 15, 2022.

[46] Chee Sun Liew, Malcolm P. Atkinson, Michelle Galea, Tan Fong Ang, Paul Martin, and Jano I. Van Hemert. 2016. Scientific Workflows: Moving Across Paradigms. *ACM Comput. Surv.* 49, 4, Article 66 (2016). https://doi.org/10.1145/3012429

[47] Xiaofan Lin and Yan Xiong. 2006. Detection and analysis of table of contents based on content association. *Int'l J. Document Analysis and Recognition* 8, 2 (1 June 2006), 132–143. https://doi.org/10.1007/s10032-005-0149-4

[48] G. Linden, B. Smith, and J. York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80. https://doi.org/10.1109/MIC.2003.1167344

[49] Ruikai Liu and Jorj McKie. 2020. PyMuPDF: Python bindings for the PDF rendering library MuPDF. https://github.com/pymupdf/PyMuPDF,accessed12/15/22

[50] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. https://link.springer.com/chapter/10.1007/978-3-642-04346-8_62. In *Research and Advanced Technology for Digital Libraries*, Maristella Agosti, José Borbinha, Sarantos Kapidakis, Christos Papatheodorou, and Giannis Tsakonas (Eds.). Springer, 473–474.

[51] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. 2006. Scientific Workflow Management and the Kepler System: Research Articles. *Concurr. Comput.: Pract. Exper.* 18, 10 (Aug. 2006), 1039–1065.

[52] Javaid Akbar Manzoor. 2022. *Segmenting Electronic Theses and Dissertations By Chapters.* MS Thesis. Virginia Tech CS, http://hdl.handle.net/10919/113246.

[53] Gail McMillan and Len Peters. 1999. ETDs: Practical, Operational, and Technical Issues for Universities Implementing Electronic Theses and Dissertations. In *Conference on Preservation and Access for Electronic College and University Records.*

Arizona State University, Mesa, Arizona.

[54] Kevin D. Munroe and Yannis Papakonstantinou. 2000. BBQ: A Visual Interface for Integrated Browsing and Querying of XML. In *Proceedings of VDB*. 277–296. http://db.ucsd.edu/publications/bbq.ps [last visited July 4, 2012].

[55] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *31st AAAI Conf.* 3075–3081. arXiv:1611.04230

[56] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *CoRR* abs/1906.00091 (2019).

[57] Manoj Prabhakar Paidiparthy, Ramaraja Ramanujan, Akshita Teegalapally, Madhuvanti Muralikrishnan, Romil Khimraj Balar, Shaunak Juvekar, and Vivek Murali. 2022. *CS 5604 Fall 2022 Team 2: End Users*. CS5604 team term project. Virginia Tech. http://hdl.handle.net/10919/114080 accessed on April 10, 2023.

[58] David Pinto, Andrew Mccallum, Xing Wei, and Bruce W. Croft. 2003. Table extraction using conditional random fields. In *SIGIR '03*. ACM Press, New York, NY, USA, 235–242. https://doi.org/10.1145/860435.860479

[59] Animesh Prasad, Manpreet Kaur, and Min-Yen Kan. 2018. Neural ParsCit: a deep learning-based reference string parser. *Int. J. Digit. Libr.* 19, 4 (2018), 323–337. https://doi.org/10.1007/s00799-018-0242-1

[60] PyPI. 2022. Python Requests. (2022). https://pypi.org/project/requests/ accessed on September 19, 2022.

[61] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[62] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

[63] Subhro Roy, Shyam Upadhyay, and Dan Roth. 2016. Equation Parsing : Mapping Sentences to Grounded Equations. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1088–1097. https://doi.org/10.18653/v1/d16-1117

[64] Gerard Salton, J. Allan, and Chris Buckley. 1993. Approaches to passage retrieval in full text information systems. In *Proc. 16th Annual International ACM SIGIR Conf. on R&D in Information Retrieval*. Pittsburgh, 49–58.

[65] Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proc. ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, Regina Barzilay and Min-Yen Kan (Eds.). 1073–1083. https://doi.org/10.18653/v1/P17-1099

[66] Anmol Shukla, Aaron Travasso, Harish Babu Manogaran, Pallavi Kishor Sisodia, and Yuze Li. 2022. *CS 5604 Fall 2022 Team 5: Integration*. CS5604 team term project. Virginia Tech. http://hdl.handle.net/10919/114078 accessed on April 10, 2023.

[67] Jeremy Singer-Vine. 2022. pdfplumber. https://github.com/jsvine/pdfplumber original-date: 2015-08-24T03:14:48Z.

[68] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing* 21, 3 (2017), 12–18. https://doi.org/10.1109/MIC.2017.72

[69] Akash Srivastava and Charles Sutton. 2017. Autoencoding variational inference for topic models. *arXiv preprint arXiv:1703.01488* (2017).

[70] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine* 42, 3 (2021), 7–18. https://doi.org/10.1609/aimag.v42i3.18140

[71] Hussein Suleman and Edward A. Fox. 2002. Towards Universal Accessibility of ETDs: Building the NDLTD Union Archive. In *ETD'2002*. Provo, Utah.

[72] Ian Taylor, Matthew Shields, Ian Wang, and Andrew Harrison. 2007. *The Triana Workflow Environment: Architecture and Applications*. Springer London, London, 320–339. https://doi.org/10.1007/978-1-84628-757-2_20

[73] Silvia Terragni, Elisabetta Fersini, Bruno Giovanni Galuzzi, Pietro Tropeano, and Antonio Candelieri. 2021. OCTIS: Comparing and Optimizing Topic models is Simple!. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. 263–270.

[74] Sentence Transformers. 2022. distilroberta. (2022). https://huggingface.co/distilroberta-base accessed on November 2, 2022.

[75] Sentence Transformers. 2022. LaBSE. (2022). https://huggingface.co/sentence-transformers/LaBSE accessed on November 2, 2022.

[76] Sentence Transformers. 2022. MiniLM L12 V2. (2022). https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2 accessed on November 2, 2022.

[77] Sentence Transformers. 2022. MPNET Base V2. (2022). https://huggingface.co/sentence-transformers/all-mpnet-base-v2 accessed on November 2, 2022.

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[79] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696* (2022).

[80] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696* (2022).

[81] William A. Ingram, Edward A. Fox, and Jian Wu. 2022. IMLS Grant LG-37-19-0078-19: Opening Books and the National Corpus of Graduate Research. https://opening-etds.github.io/. Accessed: 3-July-2022.

[82] Katherine Wolstencroft, Robert Haines, Donal Fellows, Alan Williams, David Withers, Stuart Owen, Stian Soiland-Reyes, Ian Dunlop, Aleksandra Nenadic, Paul Fisher, Jiten Bhagat, Khalid Belhajjame, Finn Bacall, Alex Hardisty, Abraham Nieva de la Hidalga, Maria P. Balcazar Vargas, Shoaib Sufi, and Carole Goble. 2013. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Research* 41, W1 (05 2013), W557–W561. https://doi.org/10.1093/nar/gkt328

[83] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. (2019).

[84] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html

[85] Chengxiang Zhai and Sean Massung. 2016. *Text data management and analysis: a practical introduction to information retrieval and text mining* (first ed.). Number 12 in ACM Books. ACM: Association for Computing Machinery, New York, NY.