GovScape: A Public Multimodal Search System for 70 Million Pages of Government PDFs

Kyle Deeds
Department of Computer Science
Boston University

Ying-Hsiang Huang
Information School
University of Washington

Claire Gong
Paul G. Allen School of Computer Science & Engineering
University of Washington

Shreya Shaji
Paul G. Allen School of Computer Science & Engineering
University of Washington

Alison Yan
Paul G. Allen School of Computer Science & Engineering
University of Washington

Leslie Harka
Information School
University of Washington

Samuel J Klein
Berkman Klein Center
Harvard University

Shannon Zejiang Shen
CSAIL
MIT

Mark Phillips
University Libraries
University of North Texas

Trevor Owens
American Institute of Physics

Benjamin Charles Germain Lee
Information School
University of Washington

***Abstract*—Efforts over the past three decades have produced web archives containing billions of webpage snapshots and petabytes of data. The End of Term Web Archive alone contains, among other file types, millions of PDFs produced by the federal government. While preservation with web archives has been successful, significant challenges for access and discoverability remain. For example, current affordances for browsing the End of Term PDFs are limited to downloading and browsing individual PDFs, as well as performing basic keyword search across them. In this paper, we introduce GovScape, a public search system that supports multimodal searches across 10,015,993 federal government PDFs from the 2020 End of Term crawl (70,958,487 total PDF pages) – to our knowledge, all renderable PDFs in the 2020 crawl that are 50 pages or under. GovScape supports four primary forms of search over these 10 million PDFs: in addition to providing (1) filter conditions over metadata facets including domain and crawl date and (2) exact text search against the PDF text, we provide (3) semantic text search and (4) visual search against the PDFs across individual pages, enabling users to structure queries such as "redacted documents" or "pie charts." We detail the constituent components of GovScape, including the search affordances, embedding pipeline, system architecture, and open source codebase. Significantly, the total estimated compute cost for GovScape's pre-processing pipeline for 10 million PDFs was approximately $1,500, equivalent to 47,000 PDF pages per dollar spent on compute, demonstrating the potential for immediate scalability. Accordingly, we outline steps that we have already begun pursuing toward multimodal search at the 100+ million PDF scale. GovScape can be found at https://www.govscape.net.**

***Index Terms*—multimodal search, exploratory search, web archives, End of Term Web Archive, collections as data**

I. Introduction

Longstanding efforts by cultural heritage institutions such as the Internet Archive and the Library of Congress have yielded enormously rich web archives over the past two decades [1]. These web archives represent an unparalleled opportunity to study the history of the 21st century. The End of Term Web Archive is unique among them due to its size as a public domain web archive and its historical significance, consisting of expansive crawls of federal .gov domain websites at the end of every presidential administration since 2004 [2]. It is a rich source for journalists, academic researchers, and the American public alike.

Despite the availability of bulk data, the End of Term Web Archive – like all web archives of its size – remains difficult to search [3]. Currently, in order to search the PDFs, users must navigate the basic metadata facets (i.e., domain and crawl date) by downloading and querying massive CDX files and then downloading individual PDFs, or performing basic keyword search through the Internet Archive's search functionality [4]. End-users face this challenge with web archives writ large. Given the centrality of web archives to understanding the federal government in the 21st century, the importance of addressing this challenge of searchability at scale is redoubled.

Recent research within library and information science as well as the digital humanities has demonstrated the value of multimodal models such as CLIP [5] for searching digital cultural heritage collections [6]–[10]. Previous work has also documented that born-digital government PDFs are not just textual but visual documents [11]. For example, these PDFs contain redactions, figures, presentation slides, form structures, and – in the case of digitized documents – artifacts of their materiality, all of which are visual in nature. Moreover, allowing users to formulate semantic natural language queries beyond standard keyword search (in which documents containing the exact search string or similar ones are returned) enables wider searches with higher precision. We believe that end-users should be able to formulate expressive queries across the PDFs in the End of Term Web Archive in particular – and files

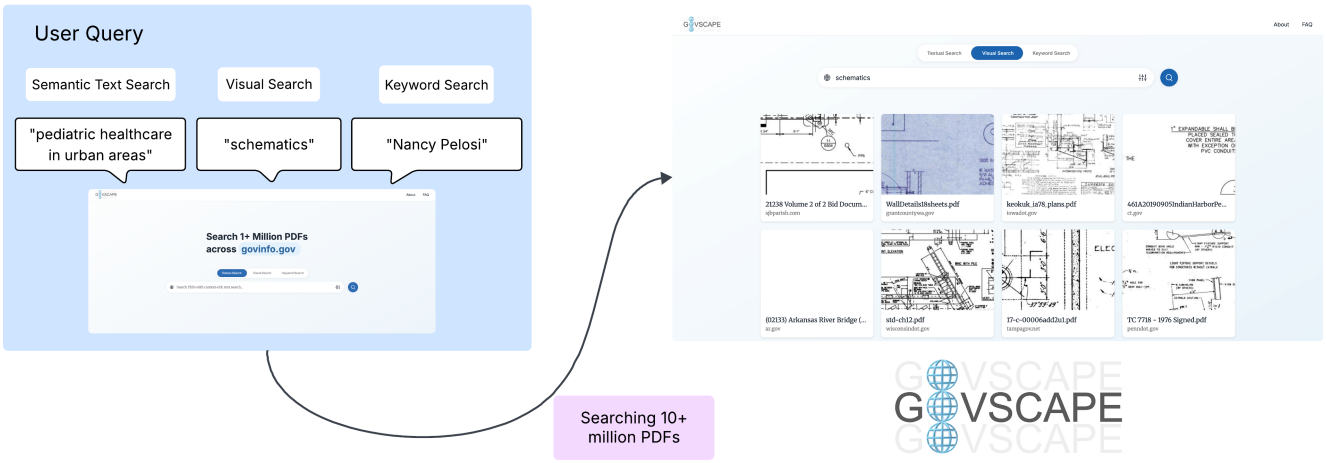


Fig. 1: An overview of GovScape. Our public search system supports three types of search over 10,015,993 million government PDFs (70,958,487 PDF pages): 1) semantic text search over PDF text, 2) visual search over individual PDF pages (treated as images), and 3) keyword search over PDF text, all of which can be applied in conjunction with filter conditions against metadata, including domain and crawl date.

in web archives more generally – according to their inherent multimodal characteristics.

In this paper, we present GovScape, a publicly-deployed search system containing 10 million PDFs from the 2020 End of Term crawl, covering the online presence of the federal government under the first Trump administration. In Figure 1, we show an overview of GovScape, including example queries that can be performed. Built with a scalable back-end consisting of Faiss [12], our implementation of GovScape supports multiple types of searches over the textual and visual features of these PDFs:

- Semantic text search
- Visual search over individual PDF pages (treated as images)
- Keyword search (i.e., exact text search)

All of these search methods can be combined with metadata filtering over facets extracted from the CDX files.

The contributions of our paper are as follows:

1) We introduce GovScape, a publicly-deployed, multimodal search system for 10 million government PDFs (70 million pages) from the 2020 crawl in the End of Term Web Archive. GovScape is available at: https://www.govscape.net.
2) We detail our multimodal search affordances and their implementations, including visual search and semantic text search.
3) We describe our development and public deployment of GovScape, including our scalable embedding pipeline and system architecture. Significantly, our estimated compute cost for processing all 10 million PDFs is approximately $1,500 (amounting to approximately 47,000 pages per dollar spent on compute).
4) We release all of our open-source code at: https://github.com/bcglee/govscape/. This code is extensible to other large-scale document collections.
5) We detail ongoing work to expand GovScape to the 100+ million PDF scale.

II. Related Work

A. Searching Web Archives

Monumental efforts to archive and preserve petabytes of web data – of interest to journalists, academics, and members of the public – have been widely successful [1], [13]. However, questions of access remain. The primary mode of access remains single-URL lookup [14], as popularized by the Internet Archive's Wayback Machine. While this mode of access is useful if an end-user knows the specific URL(s) of archived data they are interested in studying, it is also fundamentally restrictive: for example, it is not easy for an end-user to search *across* webpages.

While keyword search over web archives is extremely valuable and increasingly supported – as evidenced by the Internet Archive's text search functionality across 64 million .gov PDFs [4] – this method has its own intrinsic limitations. For example, keyword search requires that a string match be present within PDF text to be returned. Moreover, queries are restricted to the text that is present. Visual content, for example, can be entirely lost, requiring a relevant caption or alt-text to register during a keyword search.

Recent efforts have provided promising new modes of access to web archives. For example, the Harvard Library Innovation Lab's open-source WARC-GPT project enables retrieval-augmented generation (RAG)-style search over web archive files, "allow[ing] for creating custom chatbots that use a set of web archive files as their knowledge base, letting users

explore collections through conversation" [15]. The Archives Unleashed toolkit is an open-source codebase that enables large-scale data extraction and analysis of web archives, similarly focusing on textual and metadata-based visualization and analysis [16]–[18]. The SafeDocs File Observatory application enables searching over digital documents according to low-level metadata features [19]. Produced by the National Library of Hungary, SolrWayBack supports reverse image search in addition to free text search across file types, site domain visualization, and image geo search for up to 20 terabytes of WARC data [20]. With GovScape, we narrow our focus to PDFs but widen the scope and expressivity of potential searches through different forms of multimodal queries.

B. Government Documents & PDFs

Government documents are essential not only to journalists but also to researchers in fields ranging from media studies to history, economics, public policy, and law [21], [22]. Projects such as the FOIArchive [23] and the Data Liberation Project [24] demonstrate the research possibilities enabled by large-scale access to government documents [25], [26]. Federal .gov websites, and the documents contained within them, offer unique opportunities to study a vast range of questions of importance to the humanities, social sciences, and journalism. Questions range from performing large-scale analysis of specific federal agencies, to studying the aesthetic choices embedded within PowerPoint design [27], to excavating the histories of specific born-digital files [28], [29], to understanding the role of the federal government's web presence in disseminating information, to accessing information that has otherwise been modified or outright deleted. Significantly, the PDF is central to these inquiries. In *Paper Knowledge: Toward a Media History of Documents*, scholar Lisa Gitelman devotes an entire chapter to the PDF, arguing for the importance of the file type to the history of documents [21].

C. End of Term Web Archive

The End of Term Web Archive is an ongoing collaboration between institutions including the Internet Archive, the Library of Congress, and the University of North Texas [2]. Produced by crawling .gov and .mil domain sites at the end of each presidential term, it is a unique archive of the federal government's web presence in the digital age. As detailed by Phillips & Alam [30], the full End of Term Web Archive crawls for 2008, 2012, 2016, and 2020 are available through the Amazon AWS Open Data Sponsorship Program.

Significantly, PDFs comprise a substantive fraction of the End of Term Web Archive. In the 2008 End of Term crawl, for example, `application/pdf` is the fourth-most common MIME type, behind only `text/html`, `image/jpeg`, and `image/gif` (for further corpus-level analysis of these PDFs, we refer the reader to [31]). The PDFs within the End of Term Web Archive are publicly available and are also searchable within the Internet Archive's search functionality via basic text search [4]. GovScape builds on this work to provide multimodal search functionality across 10 million of these PDFs from the 2020 End of Term crawl – to our knowledge, all renderable PDFs 50 pages or under – enabling more expressive, comprehensive, and flexible searches within a dedicated search system and interface.

D. Multimodal Search in Cultural Heritage

As described by Smits & Wevers [8], the development of models such as CLIP has led to a recent "multimodal turn" in the digital humanities [5]. Research surrounding the application of multimodal models to digital collections has demonstrated new possibilities for discoverability, enabling expressive searches across visual collections [6], [7], [9], [10]. GovScape builds on this body of work to provide multimodal search capabilities for government documents, including natural language queries over visual features.

E. Enabling Semantic Search with Vector Indices

Embedding models such as CLIP transform complex objects (e.g., text and images) into vectors of numbers [5]. Specifically, they do so with the promise that similar objects will be mapped to similar vectors. With this promise, a user's query like "water planning" will be embedded in a similar vector to text documents outlining state water management plans. Conceptually, search is then a two-step process:

1) embedding the user's query into a vector
2) identifying the documents in the dataset that have the most similar vectors

The second step, referred to as *nearest neighbor search*, becomes challenging when there are millions or billions of documents in the dataset. At this scale, brute force comparison of the query vector with all document vectors takes far too long to support interactive search.

To solve this problem, there is a large literature of work on *vector indices*, which build an index structure over the document vectors in order to support fast nearest neighbor search [32]–[34]. These indices typically sacrifice perfect accuracy in order to provide millisecond latency. GovScape uses the open-source Faiss indexing library to handle its nearest neighbor search [32].

III. PDF Selection & Pre-Processing Pipeline

A. PDFs Included in GovScape

We isolated PDFs within the End of Term Web Archive's 2020 crawl by identifying files listed within the CDX indices that have a file extension of .pdf or a MIME type of `application/pdf`. We deduplicated PDFs by hash, resulting in 10,532,521 total renderable PDFs. We then filtered out 516,528 PDFs with a page count above 50 pages to improve efficiency in our pipeline. The resulting 10,015,993 PDFs from 24,877 different domains were sent to our pipeline and included in GovScape. In total, this amounts to 70,958,487 pages of government PDFs. In Table I on the next page, we break down the PDFs included within GovScape by domain.

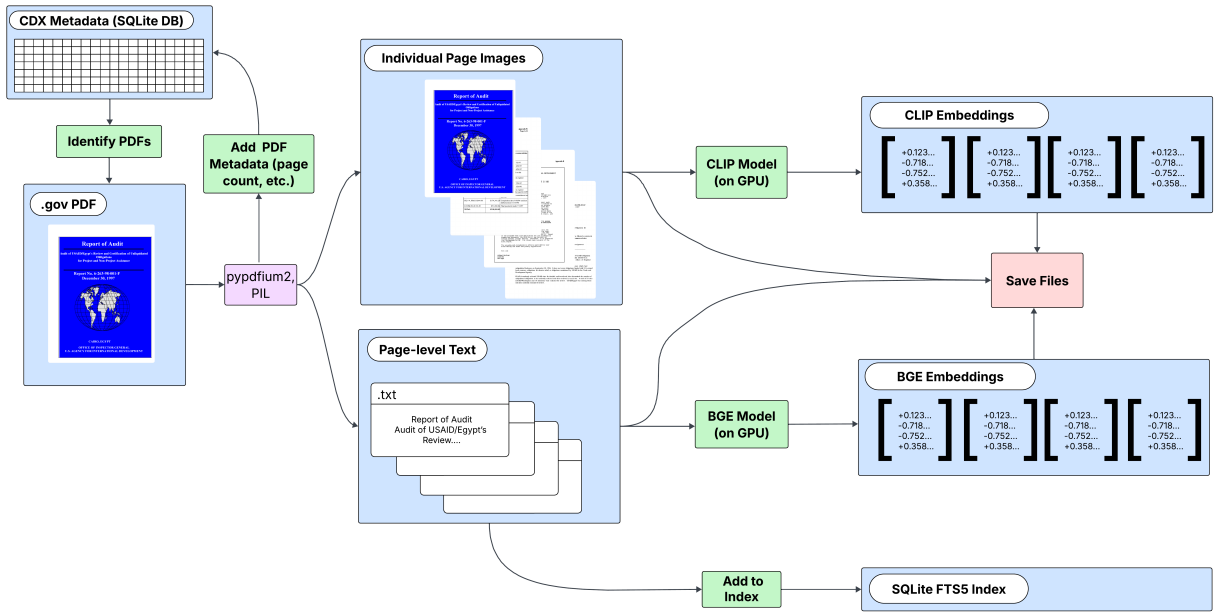


Fig. 2: An overview of the GovScape pre-processing pipeline, showing how a single PDF in GovScape is parsed and semantified.

Domain	# of PDF pages	# of PDFs
sec.gov	7,324,033	791,197
govinfo.gov	3,051,220	1,078,761
ca.gov	2,860,263	375,405
bls.gov	2,261,400	234,667
wa.gov	2,210,997	248,777
usda.gov	2,149,605	311,660
epa.gov	1,990,902	186,806
ed.gov	1,978,199	157,778
nasa.gov	1,954,285	192,714
noaa.gov	1,770,684	195,122
cdc.gov	1,493,247	157,830
illinois.gov	1,212,092	121,004
uscourts.gov	1,178,300	141,622
federalregister.gov	1,128,733	162,980
fcc.gov	1,113,856	183,620
uspto.gov	1,026,527	104,719
utah.gov	962,937	200,700
nih.gov	959,782	97,499
census.gov	914,647	223,834
hawaii.gov	894,776	157,264

TABLE I: A breakdown of the 20 most prevalent domains in GovScape, along with the associated number of PDFs included. We note that counts in this table (and this table alone) reflect PDFs before de-duplication.

B. Pre-processing Pipeline

Our pre-processing pipeline for GovScape is shown in Figure 2. Our pipeline consists of the following components:

1) **PDF Identification.** We identify PDFs using the CDX indices for the 2020 End of Term crawl, which we migrated into a SQLite database. We then extract the PDFs from the corresponding WARC files.
2) **PDF Parsing.** We split each PDF into individual pages, generating page-level images and extracting the page-level text embedded within the PDF file using `pypdfium2` and `PIL`.[1]
3) **Text Embedding.** We embed the text for each page using the `BAAI/bge-base-en-v1.5` model [35] – chosen for its balance of cost and performance as demonstrated in the MTEB leaderboard [36], [37]. Because this model has a context window of 512 tokens, we pass the first 512 tokens from the PDF page for embedding.
4) **CLIP Embedding.** We embed each page-level image using the `openai/clip-vit-base-patch32` embedding model [5] for visual recognition.
5) **Metadata Generation.** We extract relevant metadata for the PDF from the End of Term CDX files. This includes metadata fields such as URL and crawl date. We further enrich this with the number of pages in the PDF after parsing and rendering.
6) **Keyword Indexing.** We add the page-level full text to our SQLite FTS5 index, which supports traditional keyword searches over text.

[1]For some pages, no corresponding text representation is found embedded within the PDF (for example, a PDF might only have images, or it might be a digitized document without OCR). These pages are currently excluded.

Task Type	EC2 Instance Type	Time	Cost
Pipeline Minus Indexing (prior run with 4,736,080 PDFs)	`g4dn.4xlarge`	515.67 hours	$620.87
1. List		8.62 hours	
2. Download		19.31 hours	
3. PDF to Text and Image		89.16 hours	
4. Text Embedding		274.49 hours	
5. Image Embedding		100.00 hours	
6. Metadata		1.95 hours	
7. Upload		22.14 hours	
Pipeline Minus Indexing (Extrapolated to 10,015,993 PDFs)	`g4dn.4xlarge`	1,090.55 hours	**$1,313.02**
FAISS Index Building for 10,015,993 PDFs	`g4dn.4xlarge`	54.64 hours	**$65.79**
Keyword Index Building for 10,015,993 PDFs	`m5.4xlarge`	38.79 hours	**$29.79**
Total pre-processing (Extrapolated)			**$1,408.60**

TABLE II: A breakdown of compute for the GovScape pre-processing pipeline, including task type, compute specifications, time, and cost. All compute is reported using AWS `g4dn.4xlarge` instances (each with 16 Intel Cascade Lake vCPUs & 1 Nvidia T4 GPU) and m5.4xlarge instances (each with 16 Intel Xeon Platinum 8175 vCPUs) in the `us-east-2` region. We have separated out index building in this table because each index can only be built by a single server at a time. For the other steps, we extrapolate out our compute cost using a previous run of 4,736,080 PDFs, which we had configured for benchmarking.

C. PDF Pre-processing: Compute & Timing

In Table II, we break down the computing specifications, runtime, and cost for our pre-processing pipeline. Using AWS `g4dn.4xlarge` instances (each with 16 Intel Cascade Lake vCPUs and 1 Nvidia T4 GPU), we parallelized our pre-processing pipeline. We performed keyword text index building on `m5.4xlarge` instances (each with 16 Intel Xeon Platinum 8175 vCPUs) The estimated total cost of the compute required for our pre-processing pipeline was $1,408.60 (all costs are computed using the $1.204/hour on-demand cost for a `g4dn.4xlarge` instance and $0.768/hour on-demand cost for a `m5.4xlarge` instance). We report $1,500 throughout to account for our extrapolation. We note that this cost did not factor in associated AWS S3 fees, which are specific to our cloud computing setup. We incurred of order $\$1,000$ in associated fees writing to our S3 bucket. The network transfer fees between S3 and the EC2 instances were $0 because our compute and storage were co-located, and AWS does not charge for data transfer within a region.

IV. Search Affordances

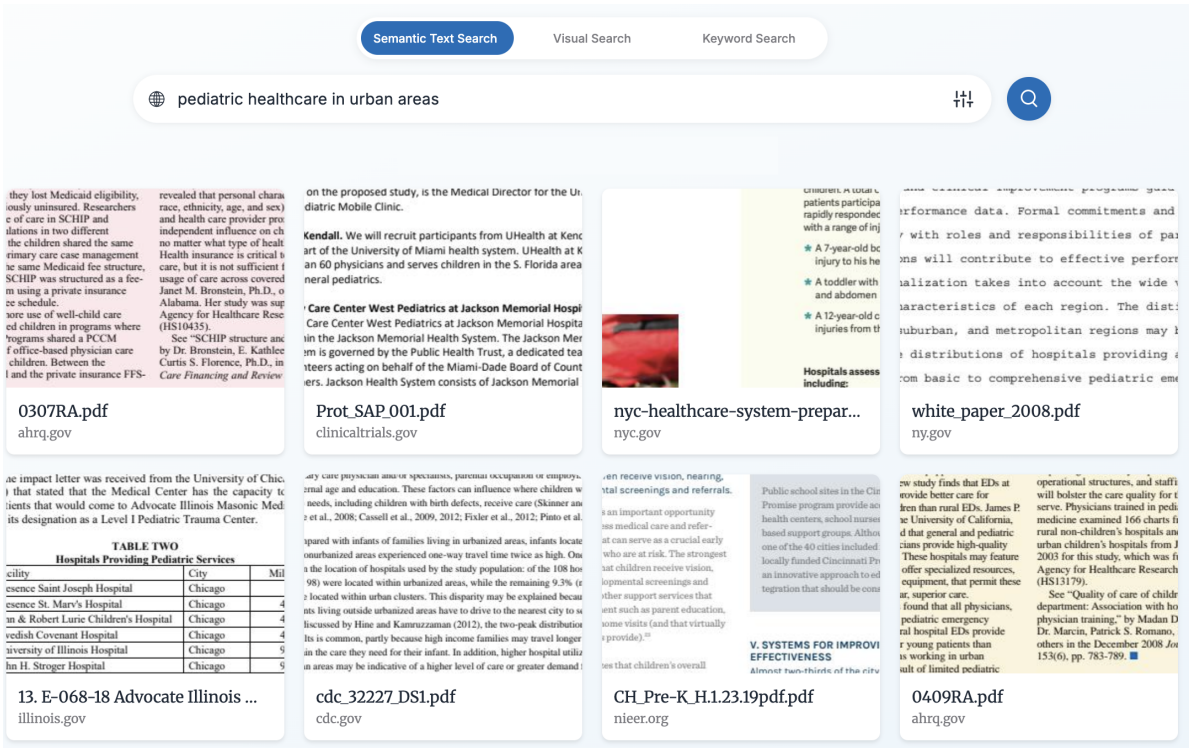
In this section, we detail all three search types supported, along with example searches and returned results. All three search types described in this section can be combined with faceted search approaches (i.e., filter conditions) against metadata including PDF domain and crawl date.

A. Semantic Text Search

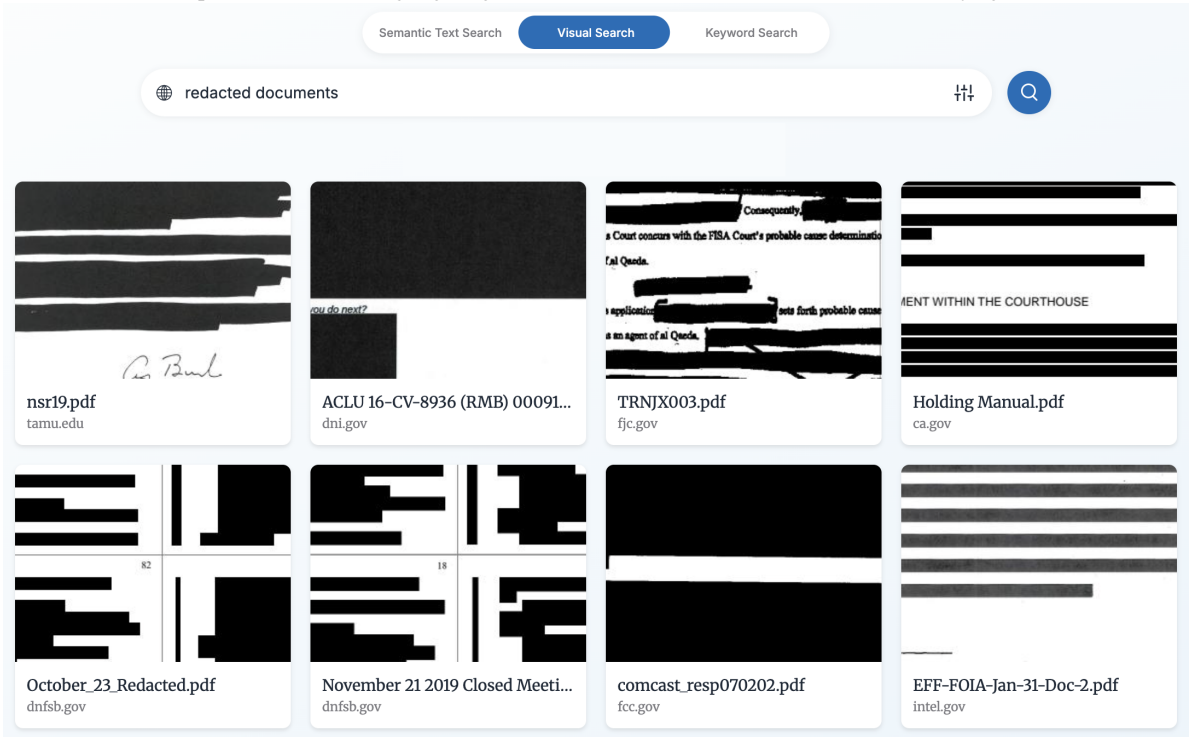
For our semantic search functionality, we leverage the `BAAI/bge-base-en-v1.5` embedding model from our pre-processing pipeline. We vectorize an end-user's natural language query such as: "economic data pertaining to geologic surveys" by embedding the query on the back-end. We then compare this embedding to the pre-computed text embeddings for all other PDF pages and retrieve the nearest neighbors using Faiss. Here, all PDF pages are ranked according to similarity (unlike exact text search, which filters results). Again, we stress that this semantic search functionality is distinct from exact text search and enables end-users to define more expressive and flexible queries. An example of semantic text search can be found in Figure 3a.

B. Visual Search

For our visual search functionality, we leverage the `openai/clip-vit-base-patch32` embedding model from our pre-processing pipeline [5]. Trained using contrastive language-image pre-training (CLIP), this CLIP model jointly embeds image and text data. Consequently, we can embed any natural language query over the visual features of the PDFs and perform a nearest-neighbors search in order to retrieve relevant results. For example, an end-user can formulate queries such as: "redacted documents," "aerial photographs," "Gantt charts," or "weather patterns." Here, we perform the analogous procedure as for semantic text search: on the back-end, each query is embedded using this CLIP model; we then compare this embedding to the pre-computed CLIP embeddings for all other PDF page images and retrieve the nearest neighbors using Faiss. As with semantic text search, all PDF pages meeting filter conditions are ranked according to similarity. An example of visual search can be found in Figure 3b.



(a) An example semantic text search in GovScape. Here, we show an example search of "pediatric healthcare in urban areas." The top three returned results include an article entitled, "Urban influence codes reveal more about children's patterns of health care use and coverage" from the HHS periodical *Research Activities* (No. 319, March 2007) from ahrq.gov; descriptions of pediatric care at the University of Miami and Jackson Memorial Hospital from a protocol entitled, "Evaluating the Effectiveness of an eHealth EBI for Latino Youth in Primary Care" from clinicaltrials.gov; and the July, 2016, to June, 2017, New York City Heath Care System Preparedness Annual Report (Section 4: "Highlighting the NYC Pediatric Disaster Coalition") from nyc.gov.



(b) An example visual search in GovScape. Here, we show an example search of "redacted documents."

Fig. 3: Examples of semantic text search (Figure 3a) and visual search (Figure 3b) in GovScape.

C. Exact Text Search

We use the FTS5 extension of SQLite to perform exact text search over the full dataset. This extension creates an on-disk inverted document index and supports both phrase and keyword search. Due to the on-disk nature of this index, this search is slower than the in-memory FAISS indices that support semantic search.

V. System Architecture

The full architecture of our GovScape implementation can be found in Figure 4. The client (top-left) sends a query to the server (middle). If the query is a semantic text search or visual search, the query is then embedded server-side and sent to Faiss (bottom-left), which returns the top k nearest-neighbor results using the logic described in Section IV-A or IV-B, respectively. For exact text search, the requests are handled by our SQLite FTS5 keyword index (bottom), as described in Section IV-C. These results are then sent to the SQLite Metadata DB bucket (bottom-right) to filter PDFs based on filter conditions (if not enough PDFs are returned, we loop back to grab more). The IDs of these filtered PDFs are sent to the client. PDF thumbnails and higher-resolution page images are retrieved from the AWS S3 bucket, and the search results are rendered for the end-user.

In this section, we further detail these constituent components. Our open-source code for GovScape is available at: https://github.com/bcglee/govscape/.

A. Back-end

The back-end of GovScape uses a Flask API Server architecture, with serving managed by Gunicorn and HTTP routing managed by nginx. As queries come in, they are handled with the following process:

1) **HTTP Handling:** The nginx server receives the query from the front-end and routes it to Gunicorn. While doing this, it handles basic functionality like encryption and rate-limiting.
2) **Load Balancing:** The Gunicorn server sends the request to one of its many threads that are currently running the application code, handling load balancing across CPU cores.
3) **Embedding:** If the query type is semantic text or visual, the handler thread embeds the query using the corresponding embedding model.
4) **Index Lookup:** Using the (potentially embedded) query, the index corresponding to the search type is queried to retrieve the top-k most relevant results.
5) **Filtering:** These results are then filtered based on the metadata stored in a SQLite database. If there are too few filtered results, k is doubled and we return to the previous step.
6) **Packaging:** The results are packaged into an HTTP response and transmitted back to the front-end.

During this process, the back-end server only requires access to the indices constructed over the data, allowing it to avoid accessing the archived data in the S3 bucket. Instead, it constructs S3 addresses for the images and full PDFs, enabling the client to perform the download directly from S3. This maximizes the throughput of the back-end server by distributing the network transfer across all client devices.

B. Front-end

The front-end architecture of GovScape is conceptually informed by the modular design patterns in the Digital Collections Explorer [38], especially in its component-based structure and the abstraction of its API communication layer. However, due to the specific needs of GovScape as a large-scale, public-facing service, we built the interface using Svelte, an open-source front-end framework. Unlike frameworks that rely on a runtime Virtual DOM, Svelte compiles components at build time into optimized vanilla JavaScript, eliminating the need for runtime diffing and reconciliation. This leads to much smaller bundle sizes and improved runtime performance.

The application is composed of modular, single-responsibility components. The main application view, `+page.svelte`, serves as the primary container, orchestrating the interactions between child components. This component-based structure enables several key features for the end-user:

- **Multi-modal search interaction**: the search interface provides three distinct search modalities: `Textual` (semantic text), `Visual` (CLIP-based), and `Keyword` search.
- **Advanced filtering**: filter conditions are managed by the `AdvancedSearch.svelte` component. This allows users to filter search results based on specific metadata or criteria.
- **An interactive results gallery** is rendered by the `ResultsGrid.svelte` component. This component displays a grid of document thumbnails based on the search results.
- **Detailed document inspection** is provided by the `PDFPreview.svelte` component.

All communication with the back-end is managed by a service layer, abstracted in `src/lib/utils/fetch.js`. When a user initiates a query, the corresponding UI component calls the `apiFetch` function, which dispatches the API request to the appropriate back-end endpoint based on the selected search mode. Upon receiving the response, the store's state – located in `src/lib/stores` – is updated, triggering a reactive re-render of the interface to display the results.

This clear separation of concerns between UI components, state management, and API services ensures the system is both scalable and maintainable.

In Figure 3, we show paginated views of ranked search results for semantic text and visual searches, respectively. In Figure 5, we show the detailed view of a PDF once a PDF page has been selected from the search results. In this view, we show each individual PDF page, along with the PDF's metadata. We also provide a button that, when clicked, downloads the PDF.

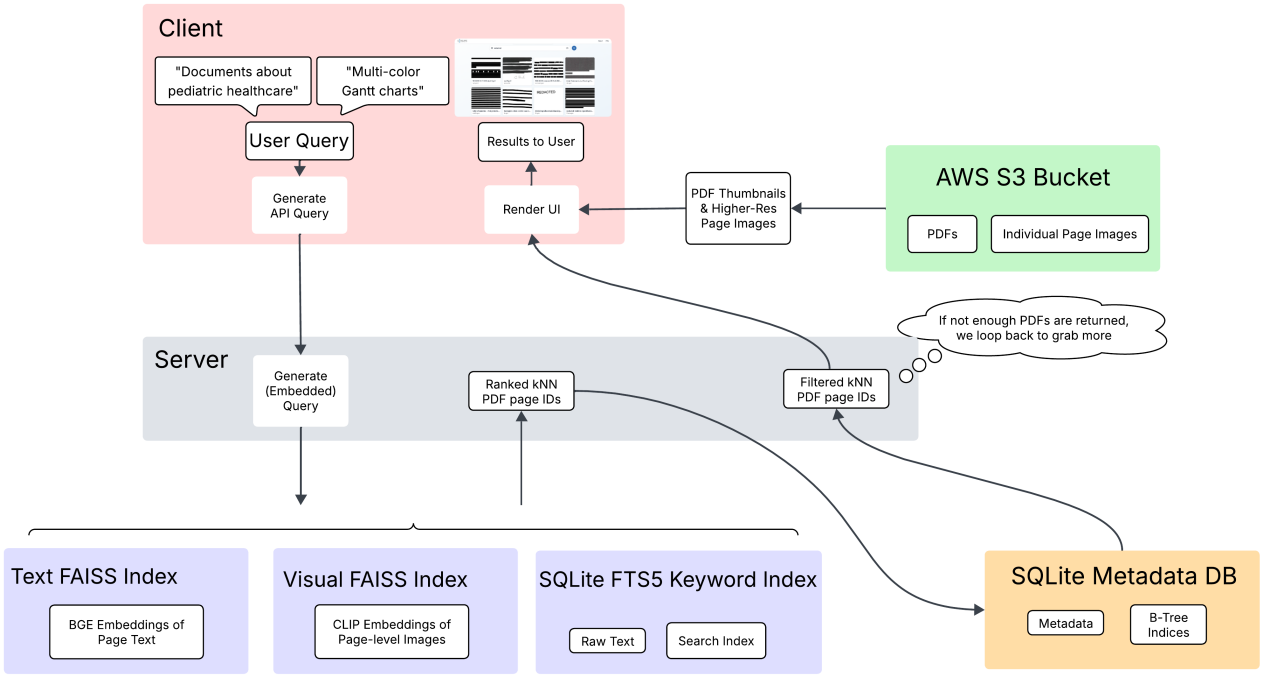


Fig. 4: An overview of the GovScape architecture, showing how the constituent parts of the system interact with one another.

VI. Conclusion & Future Work

In this paper, we have introduced GovScape, a public search system for 10,015,993 PDFs (70,958,487 PDF pages) from the 2020 crawl in the End of Term Web Archive. We have detailed the pre-processing pipeline and system architecture for GovScape, as well as highlighted its multimodal search affordances, including (1) semantic text search, enabling queries such as "pediatric healthcare in rural areas," (2) visual search over individual pages, enabling queries such as "redacted documents," (3) exact keyword search over PDF text, and (4) filter conditions over metadata. Our total estimated compute cost for processing and multimodally embedding these 70 million PDF pages was approximately $1,500. Below, we detail future work, including steps toward scaling GovScape to 100+ million government PDFs.

A. Toward 100+ Million PDFs

We are already working to scale up GovScape to comprehensively support PDF search across the entire End of Term Web Archive for all crawls – including the 2024 crawl when it is made publicly available – and beyond. The majority of compute required to accomplish this is in the pre-processing phase; based on our results in Section III-C and Table II, this goal is achievable on a relatively modest compute budget. To scale our back-end serving architecture to billions of PDF pages, we will need to transition away from the in-memory Faiss index for vector search and instead use an on-disk index like Disk-ANN, which we have already begun to prototype [39].

B. Log Analysis

We also plan to conduct anonymized log analysis of searches performed with our publicly-deployed version of GovScape (we have already received IRB exemption for this work). With this analysis, we will better understand what kinds of searches end-users are most interested in performing. Accordingly, we will refine our search affordances and interface for our system. To accomplish this, we also hope to perform further user studies beyond log analysis. Lastly, we plan to incorporate general feedback from the researchers and practitioners who use our system.

C. Optical Character Recognition & Text Search

Regarding underlying text representations, many PDFs do not contain text data that can be extracted using pypdfium2 (for example, scanned documents might not contain an underlying text representation). In the future, we plan to incorporate optical character recognition (OCR) functionality using olmOCR [40], [41] to support ingesting the text of these PDFs. We plan to route PDFs with poor-quality text extracted by pypdfium2 to olmOCR as well. Lastly, we currently embed PDF text using an English-only model; we plan to support multilingual PDFs in future iterations.

D. Model Evaluation & Finetuning

The embedding models that we have utilized are off-the-shelf models. While the performance of these models has been strong based on our qualitative observation, we plan to quantitatively assess model performance. In order to improve the quality of our embeddings for government documents in

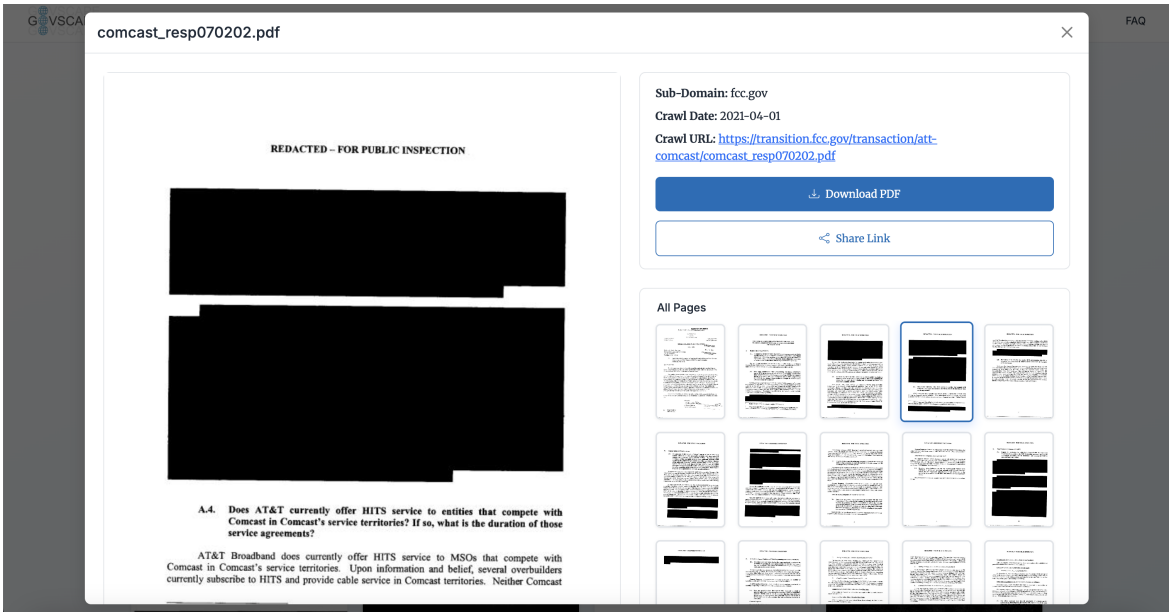


Fig. 5: A screenshot showing the selected PDF view for detailed document inspection (in this case, the fourth page of a redacted FCC document). Clicking on the first search result brings up this view, which shows the selected PDF page in the modal, its associated metadata (domain, crawl date, and crawl URL), a button to download the PDF, a button to share a link to the PDF, and thumbnail views of the other PDF pages.

particular, we also plan to finetune our embedding models on PDFs from the End of Term Web Archive and benchmark performance.

E. Mixed File Types

In the long term, we plan to expand GovScape to cover mixed file types beyond PDFs, enabling GovScape to provide search functionality across an even wider subset of the End of Term Web Archive. Future work includes developing efficient methodologies for embedding webpages, as well as document types beyond the PDF.

VII. Reproducibility Statement

We have released all our code for GovScape in an open-source codebase, available at: https://github.com/bcglee/govscape/. This codebase includes our pre-processing pipeline as well as the full GovScape search application. The End of Term Web Archive, including all data, can be found at: https://eotarchive.org/data/.

VIII. Acknowledgments

We would like to thank the University of Washington's Information School and Paul G. Allen School for Computer Science & Engineering for facilitating this collaboration and providing compute. We thank Magda Bałazińska and Moe Kayali at the University of Washington and Jake Poznanski and Kyle Lo at the Allen Institute for Artificial Intelligence for their generative feedback throughout this project. In accordance with the ACM AI Policy, we would like to note that we used Copilot and Cursor while developing the GovScape software.

References

[1] I. Milligan, "History in the age of abundance? : how the web is transforming historical research," Montreal, 2019.

[2] M. E. Phillips, K. K. Phillips, and S. Alam, "End of term web archive dataset: Longitudinal web archive of .gov and .mil domains," in *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, 2023, pp. 98–101.

[3] J. Ogden and E. Maemura, "'go fish': Conceptualising the challenges of engaging national web archives for digital research," *International journal of digital humanities*, vol. 2, no. 1-3, pp. 43–63, 2021.

[4] I. Archive, "Collection search." [Online]. Available: https://web.archive.org/collection-search

[5] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 8748–8763.

[6] J. Mahowald and B. C. G. Lee, "Integrating visual and textual inputs for searching large-scale map collections with clip," 2024. [Online]. Available: https://arxiv.org/abs/2410.01190

[7] T. Smits, B. Warner, P. Fyfe, and B. C. G. Lee, "A fully-searchable multimodal dataset of the illustrated london news, 1842–1890," *Journal of Open Humanities Data*, 2025. [Online]. Available: https://doi.org/10.5334/johd.284

[8] T. Smits and M. Wevers, "A multimodal turn in digital humanities. using contrastive machine learning models to explore, enrich, and analyze digital visual historical collections," *Digital Scholarship in the Humanities*, vol. 38, no. 3, pp. 1267–1280, 03 2023. [Online]. Available: https://doi.org/10.1093/llc/fqad008

[9] *Towards multimodal computational humanities : using CLIP to analyze late-nineteenth century magic lantern slides*, ser. CEUR-WS ; 2989. CEUR-WS.org, 2021. [Online]. Available: https://hdl.handle.net/10067/1833300151162165141

[10] A. Barancová, M. Wevers, and N. van Noord, "Blind dates: Examining the expression of temporality in historical photographs," 2023.

[11] B. C. G. Lee and T. Owens, "Grappling with the scale of born-digital government publications: Toward pipelines for processing

and searching millions of pdfs," *International Journal of Digital Humanities*, vol. 3, pp. 91 – 114, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:257159777

[12] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P.-E. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," 2025. [Online]. Available: https://arxiv.org/abs/2401.08281

[13] I. Milligan, "Averting the digital dark age : How archivists, librarians, and technologists built the web a memory," Baltimore, 2024.

[14] A. Ben-David and H. Huurdeman, "Web archive search as research: Methodological and theoretical implications," *Alexandria*, vol. 25, no. 1-2, pp. 93–111, 2014. [Online]. Available: https://doi.org/10.7227/ALX.0022

[15] M. Cargnelutti, K. Mukk, and C. Stanton, February 2014. [Online]. Available: https://lil.law.harvard.edu/blog/2024/02/12/warc-gpt-an-open-source-tool-for-exploring-web-archives-with-ai/

[16] N. Ruest, J. Lin, I. Milligan, and S. Fritz, "The archives unleashed project: Technology, process, and community to improve scholarly access to web archives," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, ser. JCDL '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 157–166. [Online]. Available: https://doi.org/10.1145/3383583.3398513

[17] R. Deschamps, N. Ruest, J. Lin, S. Fritz, and I. Milligan, "The archives unleashed notebook: madlibs for jumpstarting scholarly exploration of web archives," in *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. Piscataway, NJ, USA: IEEE Press, 2019, pp. 337–338.

[18] S. Fritz, I. Milligan, N. Ruest, and J. Lin, "Fostering community engagement through datathon events: The archives unleashed experience," *Digital humanities quarterly*, vol. 15, no. 1, 2021.

[19] R. Stonebraker, M. Milano, and A. Mensikova, "jpl-safedocs/file-observatory: V1.6.1," Jul. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8132495

[20] SolrWayback, "Solrwayback." [Online]. Available: https://github.com/netarchivesuite/solrwayback

[21] L. Gitelman, *Paper Knowledge: Toward a Media History of Documents*, ser. Sign, storage, transmission. Duke University Press, 2014.

[22] M. Connelly, *The Declassification Engine: What History Reveals About America's Top Secrets*. Pantheon, 2023.

[23] M. J. Connelly, R. Hicks, R. Jervis, A. Spirling, and C. H. Suong, "Diplomatic documents data for international relations: the freedom of information archive database," *Conflict Management and Peace Science*, vol. 38, no. 6, pp. 762–781, 2021. [Online]. Available: https://doi.org/10.1177/0738894220930326

[24] D. L. Project, "The data liberation project." [Online]. Available: https://www.data-liberation-project.org/

[25] R. R. Souza, F. C. Coelho, R. Shah, and M. Connelly, "Using artificial intelligence to identify state secrets," 2016. [Online]. Available: https://arxiv.org/abs/1611.00356

[26] M. Connelly, R. Hicks, R. Jervis, and A. Spirling, "New evidence and new methods for analyzing the iranian revolution as an intelligence failure," *Intelligence and National Security*, vol. 36, no. 6, pp. 781–806, 2021. [Online]. Available: https://doi.org/10.1080/02684527.2021.1946959

[27] P. Ford, "Amazing military infographics," May 2014. [Online]. Available: https://medium.com/message/amazing-military-infographics-1ba60bdc32e7

[28] T. Owens, B. C. G. Lee, and J. Estess, "Powell.pps: Close & distant reading of primary sources in web archives," 2024.

[29] T. Owens and J. Estess, "Slide decks as government publications: exploring two decades of powerpoint files archived from us government websites." *Archival Science*, vol. 23, pp. 223–246, 2023.

[30] M. Phillips and S. Alam, "Moving the end of term web archive to the cloud to encourage research use and reuse," *2022 Web Archiving and Digital Libraries Virtual Workshop*, 2022. [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc1998717/

[31] M. Phillips and K. Murray, "Improving access to web archives through innovative analysis of pdf content," *Archiving (IS & T's Archiving Conference)*, vol. 10, no. 1, pp. 186–192, 2013. [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc155622/

[32] M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P. Mazaré, M. Lomeli, L. Hosseini, and H. Jégou, "The faiss library," *CoRR*, vol. abs/2401.08281, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.08281

[33] S. Gollapudi, N. Karia, V. Sivashankar, R. Krishnaswamy, N. Begwani, S. Raz, Y. Lin, Y. Zhang, N. Mahapatro, P. Srinivasan, A. Singh, and H. V. Simhadri, "Filtered-diskann: Graph algorithms for approximate nearest neighbor search with filters," in *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Y. Ding, J. Tang, J. F. Sequeda, L. Aroyo, C. Castillo, and G. Houben, Eds. ACM, 2023, pp. 3406–3416. [Online]. Available: https://doi.org/10.1145/3543507.3583552

[34] R. Krishnaswamy, M. D. Manohar, and H. V. Simhadri, "The diskann library: Graph-based indices for fast, fresh and filtered vector search," *IEEE Data Eng. Bull.*, vol. 48, no. 3, pp. 20–42, 2024. [Online]. Available: http://sites.computer.org/debull/A24sept/p20.pdf

[35] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-pack: Packaged resources to advance general chinese embedding," 2023.

[36] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," 2023. [Online]. Available: https://arxiv.org/abs/2210.07316

[37] K. Enevoldsen, I. Chung, I. Kerboua, M. Kardos, A. Mathur, D. Stap, J. Gala, W. Siblini, D. Krzemiński, G. I. Winata, S. Sturua, S. Utpala, M. Ciancone, M. Schaeffer, G. Sequeira, D. Misra, S. Dhakal, J. Rystrøm, R. Solomatin, Ömer Çağatan, A. Kundu, M. Bernstorff, S. Xiao, A. Sukhlecha, B. Pahwa, R. Poświata, K. K. GV, S. Ashraf, D. Auras, B. Plüster, J. P. Harries, L. Magne, I. Mohr, M. Hendriksen, D. Zhu, H. Gisserot-Boukhlef, T. Aarsen, J. Kostkan, K. Wojtasik, T. Lee, M. Šuppa, C. Zhang, R. Rocca, M. Hamdy, A. Michail, J. Yang, M. Faysse, A. Vatolin, N. Thakur, M. Dey, D. Vasani, P. Chitale, S. Tedeschi, N. Tai, A. Snegirev, M. Günther, M. Xia, W. Shi, X. H. Lù, J. Clive, G. Krishnakumar, A. Maksimova, S. Wehrli, M. Tikhonova, H. Panchal, A. Abramov, M. Ostendorff, Z. Liu, S. Clematide, L. J. Miranda, A. Fenogenova, G. Song, R. B. Safi, W.-D. Li, A. Borghini, F. Cassano, H. Su, J. Lin, H. Yen, L. Hansen, S. Hooker, C. Xiao, V. Adlakha, O. Weller, S. Reddy, and N. Muennighoff, "Mmteb: Massive multilingual text embedding benchmark," 2025. [Online]. Available: https://arxiv.org/abs/2502.13595

[38] Y.-H. Huang and B. C. G. Lee, "Digital collections explorer: An open-source, multimodal viewer for searching digital collections," 2025. [Online]. Available: https://arxiv.org/abs/2507.00961

[39] S. J. Subramanya, Devvrit, R. Kadekodi, R. Krishaswamy, and H. V. Simhadri, *DiskANN: fast accurate billion-point nearest neighbor search on a single node*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[40] J. Poznanski, A. Rangapur, J. Borchardt, J. Dunkelberger, R. Huff, D. Lin, A. Rangapur, C. Wilhelm, K. Lo, and L. Soldaini, "olmocr: Unlocking trillions of tokens in pdfs with vision language models," 2025. [Online]. Available: https://arxiv.org/abs/2502.18443

[41] J. Poznanski, L. Soldaini, and K. Lo, "olmocr 2: Unit test rewards for document ocr," 2025. [Online]. Available: https://arxiv.org/abs/2510.19817