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

Ying-Hsiang Huang[1], Claire Gong[1], Shreya Shaji[1], Alison Yan[1], Leslie Harka[1], Albert Du[1], Anjali Gopal[1], Samuel J. Klein[2], Shannon Zejiang Shen[3], Mark Phillips[4], Trevor Owens[5], Kyle Deeds[6], Benjamin Charles Germain Lee[1]

[1]University of Washington, [2]Harvard University, [3]Massachusetts Institute of Technology, [4]University of North Texas, [5]American Institute of Physics, [6]Boston University

Correspondence: bcgl@uw.edu

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 millions of PDFs produced by the federal government. While preservation with web archives has been successful, significant challenges for access and discoverability remain. 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: 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 GovScape's 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. We evaluate GovScape by (1) analyzing 1,679 search queries and (2) benchmarking vector and keyword index efficiency using these queries. GovScape can be found at **https://www.govscape.net**.

1 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 (Milligan, 2019). 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 government websites at the end of every presidential administration since 2004 (Phillips et al., 2023). It is a rich source for journalists, academic researchers, and the public.

Despite the availability of bulk data, the End of Term Web Archive – like all web archives of its size – remains difficult to search (Ogden and Maemura, 2021). Currently, 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 (Archive). End-users face this challenge with web archives writ large. Given the centrality of these archives to understanding the federal government in the 21st century, the importance of addressing this challenge 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 (Radford et al., 2021) for searching digital cultural heritage collections (Mahowald and Lee, 2024; Smits et al., 2025; Smits and Wevers, 2023; Smits and Kestemont, 2021; Barancová et al., 2023) (for a more detailed list of related work, we refer the reader to Appendix A). Previous work has also documented that born-digital government PDFs are not just textual but visual documents (Lee and Owens, 2021). For example, these PDFs contain redactions, figures, presentation slides, and form structures, all of which are visual in nature. Moreover, allowing users to formulate semantic natural language queries beyond keyword search (in which documents containing the exact search string or similar ones are returned) enables wider searches with higher precision. End-users should be able to formulate expressive multimodal queries across the

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 231–241

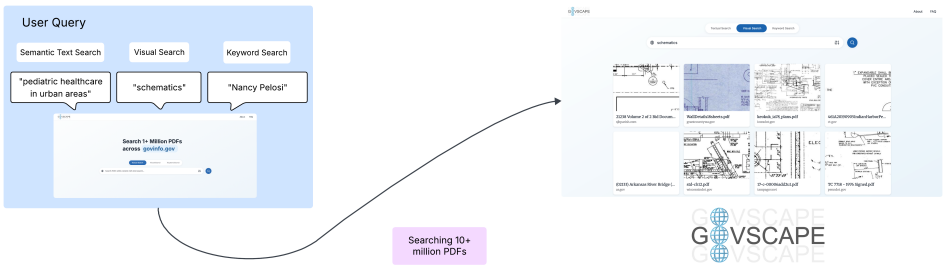

Figure 1: Our public search system supports three types of search over 10,015,993 million government PDFs (70,958,487 PDF pages): 1) semantic text search over PDF text, 2) visual search over individual PDF pages (treated as images), and 3) keyword search over PDF text, all of which can be applied with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

PDFs in the End of Term Web Archive in particular and files in web archives more generally.

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 (Douze et al., 2025), our implementation of GovScape supports multiple types of searches over the textual and visual features of these PDFs: semantic text search, visual search over 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 from the CDX files. A demo video can be found here: https://youtu.be/VpyiYW0nWp4.

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 affordances, 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. Our estimated compute cost for processing all 10 million PDFs is approximately $1,500.
4. We release all of our open-source code at: https://github.com/govscape/govscape/. This code is extensible to other information retrieval tasks and datasets.
5. We perform two forms of evaluation: log analysis of 1,679 end-user search queries, and vector & keyword index benchmarking using these queries.

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

Table 1: A breakdown of the 10 most prevalent domains in GovScape. Counts in this table (and this table alone) reflect PDFs before de-duplication.

2 Pre-Processing Pipeline

2.1 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 1, we break down the PDFs included within GovScape by domain.

2.2 Pre-processing Pipeline

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

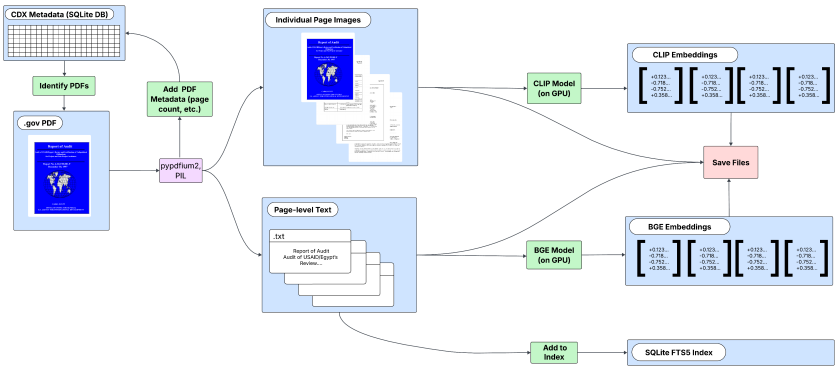

Figure 2: An overview of the GovScape pre-processing pipeline, showing how a single PDF in GovScape is parsed and semantified.

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`. We exclude pages with no text embedded within the PDF (it might only have images or be a digitized document without OCR, etc.).
3. **Text Embedding.** We embed the text for each page using the `BAAI/bge-base-en-v1.5` model (Xiao et al., 2023) chosen for its balance of cost and performance as demonstrated in the MTEB leaderboard (Muennighoff et al., 2023; Enevoldsen et al., 2025). This model has a context window of 512 tokens, and we pass the first 512 tokens from the PDF page.
4. **CLIP Embedding.** We embed each page-level image using the embedding model `openai/clip-vit-base-patch32` (Radford et al., 2021) 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.
6. **Keyword Indexing.** We add the page-level full text to our Lucene index, which supports traditional keyword searches over text.

2.3 PDF Pre-processing: Compute & Timing

In Table 2, we break down the computing specifications, runtime, and cost for our pre-processing

Task Type	Cost
Pipeline Minus Indexing (prior run w/ 4,736,080 PDFs)	$620.87
1. List	
2. Download	
3. PDF to Text & Image	
4. Text Embedding	
5. Image Embedding	
6. Metadata	
7. Upload	
Pipeline Minus Indexing (Extrapolated)	**$1,313.02**
FAISS Index Building (all PDFs)	**$65.79**
Keyword Index Building (all PDFs)	**$29.79**
Total Pre-processing (Extrapolated)	**$1,408.60**

Table 2: A breakdown of compute for the GovScape pre-processing pipeline. All compute is reported using AWS `g4dn.4xlarge` instances (each with 16 Intel Cascade Lake vCPUs & 1 Nvidia T4 GPU) and, with keyword index building, m5.4xlarge instances (each with 16 Intel Xeon Platinum 8175 vCPUs) in the `us-east-2` region. We list index building separately because each index can only be built by one server at a time. For other steps, we extrapolate our cost using a previous run of 4,736,080 PDFs, which we had configured for benchmarking. We note that this keyword index building cost reflects an earlier build using SQLite FTS5; given our results in Table 3, building our current Lucene keyword index is cheaper.

(a) A semantic text search: "pediatric healthcare in urban areas."

(b) A visual search: "redacted documents."

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

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. The estimated total cost of the compute required for our pre-processing pipeline was $1,408.60.[1] We report $1,500 to account for our extrapolation.[2]

3 Search Affordances

All three search affordances detailed below can be combined with filter conditions against metadata.

3.1 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.

3.2 Visual Search

For our visual search functionality, we leverage the `openai/clip-vit-base-patch32` embedding model from our pre-processing pipeline (Radford et al., 2021). Trained using contrastive language-image pre-training (CLIP), this model jointly embeds image and text data. 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" or "aerial photographs." 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. Figure 3b shows an example of visual search.

3.3 Exact Text Search

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

4 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

[1] All costs are computed using the $1.204/hr on-demand cost for a `g4dn.4xlarge` instance and $0.768/hr on-demand cost for a `m5.4xlarge` instance.

[2] 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 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.

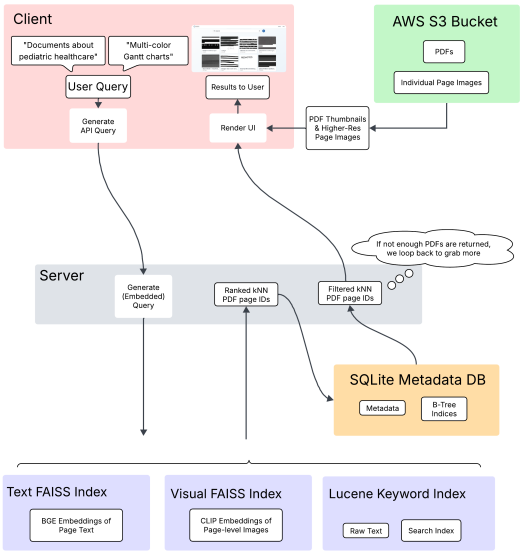

Figure 4: An overview of the GovScape architecture, showing how the constituent parts of the system interact.

nearest-neighbor results using the logic described in Section 3.1 or 3.2, respectively. For exact text search, the requests are handled by our Lucene keyword index (bottom), as described in Section 3.3. These results are then sent to the SQLite Metadata DB bucket (bottom-right) to filter PDFs based on filter conditions. 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/govscape/govscape/`.

4.1 Back-end

The back-end of GovScape uses a Flask API Server architecture, with serving managed by Gunicorn and HTTP routing managed by nginx. Queries are handled via a six-step process:

1. **HTTP Handling:** The nginx server receives the query from the front-end and routes it to Gunicorn. It also handles functionality like encryption and rate-limiting.
2. **Load Balancing:** The Gunicorn server sends the request to one of its threads that are 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 filtered against the metadata stored in a SQLite database. If there are too few filtered results, k is doubled and we repeat the previous step.
6. **Packaging:** The results are packaged into an HTTP response and sent 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.

4.2 Front-End

The front-end architecture of GovScape is conceptually informed by the modular design patterns in the Digital Collections Explorer (Huang and Lee, 2025), 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 comprises 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`.
- **Advanced filtering**: filter conditions are managed by the `AdvancedSearch.svelte` component. This allows users to filter search results based on specific criteria.

- **An interactive results gallery** is rendered by the `ResultsGrid.svelte` component. This component displays a grid of document thumbnails.
- **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. 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. Once a PDF page has been selected from the search results, we also surface a detailed view of a PDF. 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.

5 Evaluation

5.1 Log Analysis

In the three months between launching on November 18th, 2025, and February 18th, 2026, GovScape received 1,679 total searches across all three search types.[3] In total, GovScape received 992 semantic text search queries (59%), 271 visual search queries (16%), and 416 keyword search queries (25%).

Example semantic text searches include: "budgetary data for environmental surveys," "mediation prisoner litigation," and "policies regarding rainwater collection." Example visual searches include: "network," "comics," and "monument." 8% of semantic text queries have at least part of the search in quotation marks (some of the queries also contain boolean logic), suggesting that some users have conflated semantic text search and keyword search. Future work includes better communicating these search types to end-users, developing a single, catch-all query type, and conducting more nuanced log analysis as more queries are performed.

[3]All of these numbers reflect full deduplication (i.e., if a user searched for the same term using the same search type on different days, it is counted only once).

Index	Build (s)	QPS	Size (MB)
Lucene	**610**	**418**	**2806**
LanceDB	930	134	28710
SQLite	617	11	14122
Whoosh	T/O	T/O	T/O

Table 3: Keyword Index Benchmark

5.2 Index Benchmarking

To explore the tradeoffs between implementations, we performed a benchmark of various open source keyword and vector indices, as shown in Tables 3 and 4. For keyword indices, we considered Apache Lucene, a keyword search engine that powers major systems like ElasticSearch (Apache); LanceDB, a recent multi-modal search engine that offers keyword search (LanceDB, 2026); SQLite's FTS5 keyword search extension (Hipp and SQLite Development Team, 2026); and Whoosh, an open-source pure-python keyword search engine (Chaput, 2016). To test the keyword search functionality, we downloaded 500,000 pages of text from the govscape PDF corpus, inserted them into each index, then queried each index 10,000 times using real keyword queries from the search logs. Across all metrics, Apache Lucene was the most efficient index with a similar build time to SQLite but almost $40\times$ higher query throughput and $5\times$ smaller disk size. This shows the benefits of Lucene's advanced compression techniques for its inverted index.

For vector indices, we considered four different index types offered by the FAISS library for vector search (Douze et al., 2024b): IVF, a clustering-based index; IVFPQ, a combination of IVF with product quantization to compress the embeddings; HNSW, a widely-used graph-based index; and FLAT, the brute-force algorithm that compares all query and embedding vectors using BLAS routines. Finally, we again include LanceDB using its default vector index implementation and prefetching data into memory to provide an even footing. We tested these methods using $500,000$ embeddings from the Govscape corpus and $10,000$ query embeddings (produced from the search log). For use in Govscape, these indices must also contain a mapping from the embedding ID to the PDF page that generated it, so we also store dictionaries that hold these mappings for the FAISS methods. For LanceDB, we use its internal metadata storage mechanisms.

Across methods, we find that the HNSW index provides the best speed and accuracy tradeoff, but suffers in memory consumption due to its lack of

Index	Build (s)	QPS	Size (MB)	Recall@10
IVF	596	500	2960	.9674
IVFPQ	620	414	1588	.9829
HNSW	101	**750**	3062	.9998
Flat	**1.5**	8.0	2933	**1.000**
LanceDB	49.8	125.9	**1509**	.8689

Table 4: Vector Index Benchmark

embedding compression. Because space is at a premium as the dataset grows and accuracy is critical for correct search results, IVFPQ was selected as Govscape's primary vector index. However, as data scales beyond memory, it will be necessary to utilize a disk-based index like LanceDB.

6 Conclusion & Future Work

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 semantic text search, visual search, exact keyword search over PDF text, and filter conditions. Our estimated compute cost for processing and multimodally embedding these 70 million PDF pages was approximately $1,500. We have also provided two forms of evaluation: search log analysis and vector & keyword index benchmarking. A demo video of GovScape can be found at: https://youtu.be/VpyiYW0nWp4.

We are already working to scale up GovScape to comprehensively support PDF search across all crawls in the End of Term Web Archive (including the 2024 crawl when it is publicly available). The majority of compute required is in pre-processing; our results in Table 2 suggest that this goal is achievable on a relatively modest budget.

Many PDFs do not contain text data that can be usefully extracted using pypdfium2 (e.g., scanned documents might not contain an underlying text representation). We plan to incorporate optical character recognition (OCR) functionality using olmOCR and are already working to benchmark OCR performance against our documents (Poznanski et al., 2025a,b). Moreover, we currently embed PDF text using an English-only model; we plan to support multilingual PDFs.

The embedding models that we have utilized are off-the-shelf models. While their performance has been strong based on our qualitative observation, we plan to quantitatively assess model performance via a range of strategies. To improve the quality of our embeddings for government documents, we plan to finetune our embedding models. We will also begin incorporating vision language models for more sophisticated forms of search & retrieval. In the long term, we plan to expand GovScape to cover mixed file types and provide search across an even wider set of government information.

Lastly, we plan to continue our log analysis of our deployed system in order to establish relevance-based metrics and study user intent. This analysis will inform our refinement of our search affordances and evaluation of embedding methods. In the long term, we hope to conduct in-person studies with end-users to augment our log analysis.

7 Ethics & Broader Impact Statement

We have designed GovScape with broader impact in mind. We created and developed GovScape with the central goal of improving public access to information produced by the United States Federal Government. To ensure reproducibility and public availability, we have released all our code for GovScape in an open-source codebase available at: https://github.com/govscape/govscape/. This includes our pre-processing pipeline as well as the full GovScape search application. The full End of Term Web Archive can be found at: https://eotarchive.org/data/. We note that we used Copilot and Cursor while developing the GovScape codebase. For our log analysis, we received an IRB exemption from the University of Washington Human Subjects Division, ensuring ethical human subjects practices.

8 Acknowledgments

We 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 AI for their feedback.

References

Apache. Welcome to apache lucene.

Internet Archive. Collection search.

Alexandra Barancová, Melvin Wevers, and Nanne van Noord. 2023. Blind dates: Examining the expression

of temporality in historical photographs. *Preprint*, arXiv:2310.06633.

Anat Ben-David and Hugo Huurdeman. 2014. Web archive search as research: Methodological and theoretical implications. *Alexandria*, 25(1-2):93–111.

Matteo Cargnelutti, Kristi Mukk, and Clare Stanton. 2014. Warc-gpt: An open-source tool for exploring web archives using ai.

Matt Chaput. 2016. Whoosh: Fast, pure-python full text indexing, search, and spell checking library. Python Package Index.

Matthew Connelly. 2023. *The Declassification Engine: What History Reveals About America's Top Secrets*. Pantheon.

Matthew Connelly, Raymond Hicks, Robert Jervis, and Arthur Spirling. 2021a. New evidence and new methods for analyzing the iranian revolution as an intelligence failure. *Intelligence and National Security*, 36(6):781–806.

Matthew J Connelly, Raymond Hicks, Robert Jervis, Arthur Spirling, and Clara H Suong. 2021b. Diplomatic documents data for international relations: the freedom of information archive database. *Conflict Management and Peace Science*, 38(6):762–781.

Ryan Deschamps, Nick Ruest, Jimmy Lin, Samantha Fritz, and Ian Milligan. 2019. The archives unleashed notebook: madlibs for jumpstarting scholarly exploration of web archives. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 337–338, Piscataway, NJ, USA. IEEE Press.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024a. The faiss library. *CoRR*, abs/2401.08281.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024b. The faiss library.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2025. The faiss library. *Preprint*, arXiv:2401.08281.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, and 67 others. 2025. Mmteb: Massive multilingual text embedding benchmark. *Preprint*, arXiv:2502.13595.

Paul Ford. 2014. Amazing military infographics.

Samantha Fritz, Ian Milligan, Nick Ruest, and Jimmy Lin. 2021. Fostering community engagement through datathon events: The archives unleashed experience. *Digital humanities quarterly*, 15(1).

Lisa Gitelman. 2014. *Paper Knowledge: Toward a Media History of Documents*. Sign, storage, transmission. Duke University Press.

Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. 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*, pages 3406–3416. ACM.

D. Richard Hipp and SQLite Development Team. 2026. SQLite FTS5 extension. `https://www.sqlite.org/fts5.html`. Module for full-text search, released in version 3.9.0.

Ying-Hsiang Huang and Benjamin Charles Germain Lee. 2025. Digital collections explorer: An open-source, multimodal viewer for searching digital collections. *Preprint*, arXiv:2507.00961.

Ravishankar Krishnaswamy, Magdalen Dobson Manohar, and Harsha Vardhan Simhadri. 2024. The diskann library: Graph-based indices for fast, fresh and filtered vector search. *IEEE Data Eng. Bull.*, 48(3):20–42.

LanceDB. 2026. LanceDB: An open-source, serverless vector database for AI. `https://github.com/lancedb/lancedb`. Accessed: 2026-02-25.

Benjamin Charles Germain Lee and Trevor Owens. 2021. Grappling with the scale of born-digital government publications: Toward pipelines for processing and searching millions of pdfs. *International Journal of Digital Humanities*, 3:91 – 114.

Jamie Mahowald and Benjamin Charles Germain Lee. 2024. Integrating visual and textual inputs for searching large-scale map collections with clip. *Preprint*, arXiv:2410.01190.

Ian Milligan. 2019. History in the age of abundance? : how the web is transforming historical research.

Ian Milligan. 2024. Averting the digital dark age : How archivists, librarians, and technologists built the web a memory.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

Jessica Ogden and Emily Maemura. 2021. 'go fish': Conceptualising the challenges of engaging national web archives for digital research. *International journal of digital humanities*, 2(1-3):43–63.

Trevor Owens and Jonah Estess. 2023. Slide decks as government publications: exploring two decades of powerpoint files archived from us government websites. *Archival Science*, 23:223–246.

Trevor Owens, Benjamin Charles Germain Lee, and Jonah Estess. 2024. Powell.pps: Close & distant reading of primary sources in web archives.

Mark Phillips and Sawood Alam. 2022. Moving the end of term web archive to the cloud to encourage research use and reuse. *2022 Web Archiving and Digital Libraries Virtual Workshop*.

Mark Phillips and Kathleen Murray. 2013. Improving access to web archives through innovative analysis of pdf content. *Archiving (IS & T's Archiving Conference)*, 10(1):186–192.

Mark E. Phillips, Kristy K. Phillips, and Sawood Alam. 2023. End of term web archive dataset: Longitudinal web archive of .gov and .mil domains. In *2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 98–101.

Jake Poznanski, Aman Rangapur, Jon Borchardt, Jason Dunkelberger, Regan Huff, Daniel Lin, Aman Rangapur, Christopher Wilhelm, Kyle Lo, and Luca Soldaini. 2025a. olmocr: Unlocking trillions of tokens in pdfs with vision language models. *Preprint*, arXiv:2502.18443.

Jake Poznanski, Luca Soldaini, and Kyle Lo. 2025b. olmocr 2: Unit test rewards for document ocr. *Preprint*, arXiv:2510.19817.

Data Liberation Project. The data liberation project.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. 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*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Nick Ruest, Jimmy Lin, Ian Milligan, and Samantha Fritz. 2020. 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*, JCDL '20, page 157–166, New York, NY, USA. Association for Computing Machinery.

Thomas Smits and Mike Kestemont. 2021. Towards multimodal computational humanities : using clip to analyze late-nineteenth century magic lantern slides.

Thomas Smits, Bethany Warner, Paul Fyfe, and Benjamin Charles Germain Lee. 2025. A fully-searchable multimodal dataset of the illustrated london news, 1842–1890. *Journal of Open Humanities Data*.

Thomas Smits and Melvin Wevers. 2023. 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*, 38(3):1267–1280.

SolrWayback. Solrwayback.

Renato Rocha Souza, Flavio Codeco Coelho, Rohan Shah, and Matthew Connelly. 2016. Using artificial intelligence to identify state secrets. *Preprint*, arXiv:1611.00356.

Ryan Stonebraker, Michael Milano, and Anastasija Mensikova. 2023. jpl-safedocs/file-observatory: V1.6.1.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

A Appendix 1: Related Work

A.1 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 (Milligan, 2019, 2024). However, questions of access remain. The primary mode of access remains single-URL lookup (Ben-David and Huurdeman, 2014), as popularized by the Internet Archive's Wayback Machine. While this mode is useful if an end-user knows the specific URL(s) of archived data they are interested in studying, it is also 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 (Archive) – 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" (Cargnelutti et al., 2014). 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 (Ruest et al., 2020; Deschamps et al., 2019; Fritz et al., 2021). The SafeDocs File Observatory application enables searching over digital documents according to low-level metadata features (Stonebraker et al., 2023). The National Library of Hungary's 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 (SolrWayback). With GovScape, we narrow our focus to PDFs but widen the scope and expressivity of potential searches through different forms of multimodal queries.

A.2 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 (Gitelman, 2014; Connelly, 2023). Projects such as the FOIArchive (Connelly et al., 2021b) and the Data Liberation Project (Project) demonstrate the research possibilities enabled by large-scale access to government documents (Souza et al., 2016; Connelly et al., 2021a). 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 (Ford, 2014), to excavating the histories of specific born-digital files (Owens et al., 2024; Owens and Estess, 2023), 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*, Lisa Gitelman devotes an entire chapter to the PDF, arguing for the importance of the file type to the history of documents (Gitelman, 2014).

A.3 End of Term Web Archive

Produced by crawling .gov and .mil domain sites at the end of each presidential term, the End of Term Web Archive is a unique archive of the federal government's web presence in the digital age (Phillips et al., 2023). The full End of Term Web Archive crawls for 2008, 2012, 2016, and 2020 are available through the Amazon AWS Open Data Sponsorship Program (Phillips and Alam, 2022). PDFs comprise a substantive fraction of the 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` (Phillips and Murray, 2013). The PDFs within the End of Term Web Archive are publicly available and are also searchable within the Internet Archive via basic text search (Archive). 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.

A.4 Multimodal Search in Cultural Heritage

As described by Smits & Wevers (Smits and Wevers, 2023), the development of models such as CLIP has led to a recent "multimodal turn" in the digital humanities (Radford et al., 2021). Research surrounding the application of multimodal models to digital collections has demonstrated new possibilities for discoverability, enabling expressive searches across visual collections (Mahowald and Lee, 2024; Smits et al., 2025; Smits and Kestemont, 2021; Barancová et al., 2023). GovScape builds on this body of work to provide multimodal search capabilities for government documents, including natural language queries over visual features.

A.5 Enabling Semantic Search with Vector Indices

Embedding models such as CLIP transform complex objects (e.g., text and images) into vectors of numbers (Radford et al., 2021). 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, and (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 takes far too long to support interactive search.

To solve this problem, there is a large body of work on *vector indices*, which build an index structure over the document vectors in order to support fast nearest neighbor search(Douze et al., 2024a; Gollapudi et al., 2023; Krishnaswamy et al., 2024). These indices typically sacrifice perfect accuracy in order to provide millisecond latency. GovScape uses the open-source Faiss library to handle its nearest neighbor search (Douze et al., 2024a).