Computational Humanities Research

www.cambridge.org/chr

Digital collections explorer: An open-source, multimodal viewer for searching digital collections

Ying-Hsiang Huang[1] and Benjamin Charles Germain Lee[2]

[1]Information School, University of Washington, USA and [2]Information School, University of Washington, USA

Software Paper

Cite this article: Huang Ying-Hsiang and Benjamin Charles Germain Lee. 2025. "Digital collections explorer: An open-source, multimodal viewer for searching digital collections" *Computational Humanities Research*, **1**:e14, https://doi.org/10.1017/chr.2025.10017

Received: 22 July 2025
Revised: 10 October 2025
Accepted: 23 October 2025

Keywords:
computing cultural heritage; exploratory search; information retrieval; photograph viewer; multimodal machine learning; open source

Corresponding author:
Benjamin Charles Germain Lee;
E-mail: bcgl@uw.edu

Abstract

We present Digital Collections Explorer, a web-based, open-source exploratory search platform that leverages Contrastive Language-Image Pre-training for enhanced visual discovery of digital collections. Our Digital Collections Explorer can be installed locally and configured to run on a visual collection of interest on disk in just a few steps. Building upon recent advances in multimodal search techniques, our interface enables natural language queries and reverse image searches over digital collections with visual features. This article describes the system's architecture, implementation and application to various cultural heritage collections, demonstrating its potential for democratizing access to digital archives, especially those with impoverished metadata. We present case studies with maps, photographs and PDFs extracted from web archives in order to demonstrate the flexibility of the Digital Collections Explorer, as well as its ease of use. We demonstrate that the Digital Collections Explorer scales to hundreds of thousands of images on a MacBook Pro with an M4 chip. Lastly, we host a public demo of Digital Collections Explorer.

Plain language summary

In the computational humanities, researchers are experimenting with the application of multimodal models to digital cultural heritage collections in order to improve discoverability and semantic analysis. However, it is difficult for end-users to make use of these multimodal advances, as they are in need of open-source packages and interfaces for producing embeddings and interacting with these collections, respectively. In this article, we introduce our Digital Collections Explorer, an easy-to-install exploratory search platform that can be run locally to 1) produce multimodal embeddings for a digital collection and 2) spin up a local exploratory interface for searching the digital collection in a multimodal fashion. To demonstrate the extensibility of the Digital Collections Explorer, we show case studies across photojournalism collections, digitized maps and PDFs extracted from web archives and demonstrate that the Digital Collections Explorer can scale to hundreds of thousands of images. Lastly, we include a tutorial enumerating how to build an exploratory, multimodal search interface for a digital collection. A demo of an example collection made searchable with Digital Collections Explorer can be found at: https://www.digital-collections-explorer.com. Our code can be found at: https://doi.org/10.5281/zenodo.15744570.

Introduction

Despite the significant advances in providing access to both digitized and born-digital collections over the past three decades, digital collections – particularly those with visual features – face significant challenges surrounding discoverability. While manually-curated metadata for photographs, maps and other visual culture are incredibly valuable when searching a collection, this approach simply does not scale to millions of items. The digitized *Chronicling America* newspaper collection now has over 20 million individual pages digitized, and born-digital collections are even larger, with petabytes of data comprising billions of items. As a result, collections often lack basic descriptive metadata – and without basic metadata facets, it is fundamentally difficult to search collections.

Researchers in the computational humanities and cultural heritage have long been interested in automated approaches to metadata augmentation, as evidenced by the long history of optical character recognition (OCR) for the text transcription of digitized documents (Cordell 2020). The advent of multimodal models such as Contrastive Language-Image Pre-training (CLIP) (Radford et al. 2021) that capture visual and textual information jointly have shown great promise for addressing this challenge for collections ranging from maps (Mahowald and Lee 2024) to newspapers (Smits et al. 2025). While this research has demonstrated the ability to search over collections with little to no associated metadata, this research must still be translated

This article was awarded Open Materials badge for transparent practices. See the Data availability statement for details.

© The Author(s), 2025. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives licence (https://creativecommons.org/licenses/by-nc-nd/4.0), which permits non-commercial re-use, distribution, and reproduction in any medium, provided that no alterations are made and the original article is properly cited. The written permission of Cambridge University Press must be obtained prior to any commercial use and/or adaptation of the article.

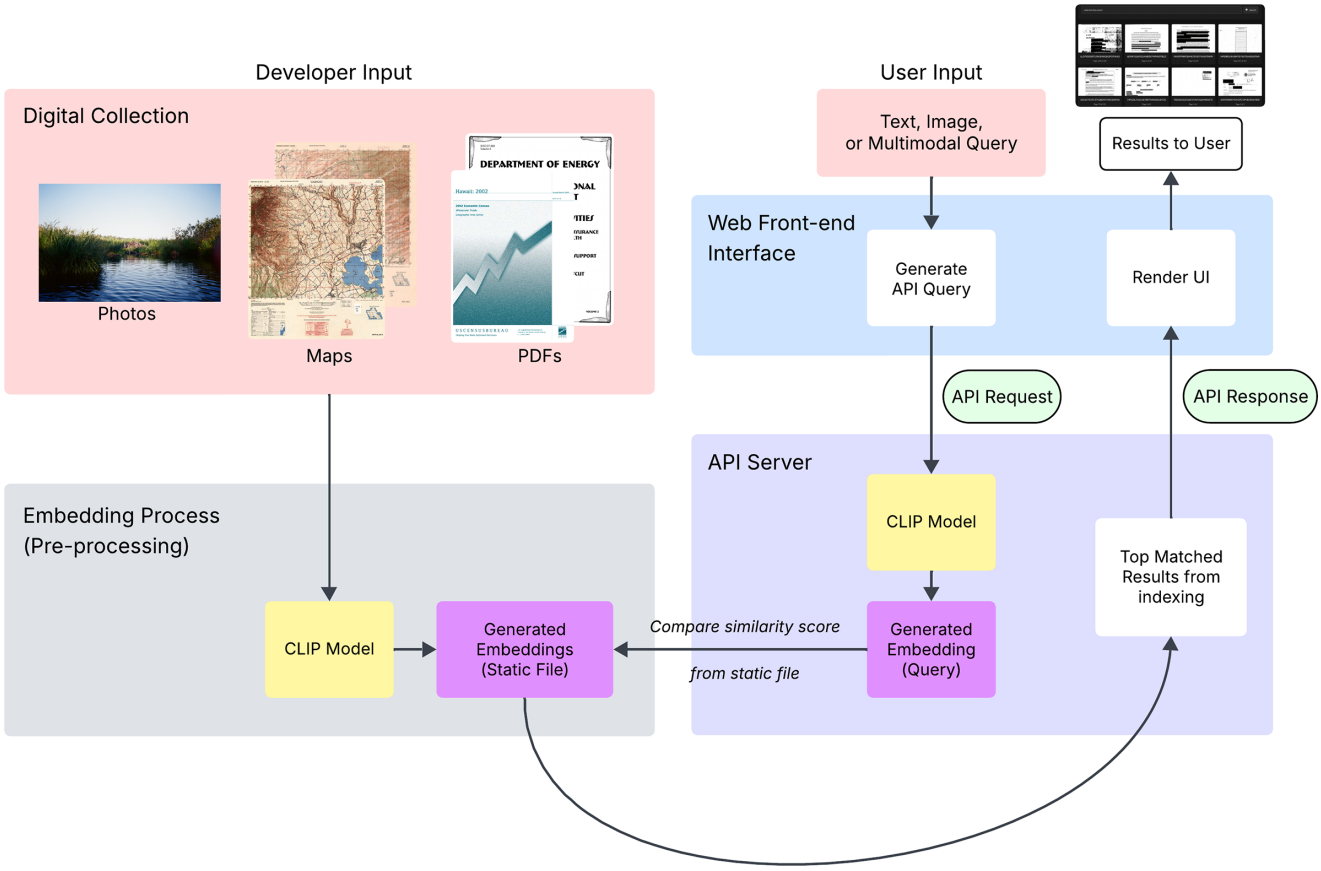


Figure 1. An overview of the Digital Collections Explorer, showing the central components: 1) the developer input, 2) the embedding process, 3) user input, 4) the web front-end interface, and 5) the API server.

into practice. Stewards of these collections are in need of democratized solutions for making their collections discoverable using these approaches – in particular, ones that are accessible to non-experts with access to only standard, staff-issued hardware (e.g., a MacBook).

In this article, we introduce our Digital Collections Explorer, which can be run locally on a laptop with only a few steps in order to spin up a multimodal search interface for a digital collection with hundreds of thousands of items. With the Digital Collections Explorer, end-users can interactively search large-scale collections using multiple input modalities, including both natural language inputs (e.g., "redacted documents") and visual inputs (i.e., reverse image search[1] or image-to-image search). Our inspiration for and implementation of the Digital Collections Explorer is based on extended collaborations with stewards of collections who have articulated precisely these needs.

The Digital Collections Explorer is designed to be easy to use for non-experts and extensible to a wide range of collections with visual features, from visual culture to documents with visual layouts and other semantic features encoded visually. In Figure 1, we show an overview of how the Digital Collections Explorer works. To spin up the Digital Collections Explorer, the developer inputs a digital collection (red, top-left). This initiates the embedding process (gray, bottom-left), which generates CLIP embeddings for all of the items in the collection.[2] Based on our case studies presented in this article, this step can scale to hundreds of thousands of items on a personal laptop. Once the embedding pre-processing is finished, the developer can spin up the viewing interface (blue, top-right), which:

1. takes a user's searches as input (red, top-right);
2. communicates with the API server (violet, bottom-right) to embed the search query and identify the top results using the CLIP embeddings;
3. renders the front-end interface and displays the results to the user (blue, top-right).

Our Digital Collections Explorer is designed to be run end-to-end locally, meaning that the embedding pipeline utilizes a locally-installed model (without transferring a digital collection to any external API), and the viewer can be spun up on a local machine as well, without being made publicly visible. In this regard, the Digital Collections Explorer can be applied even to digital collections with sensitivities surrounding privacy and access. Users interact with the system through a React-based front-end, which supports

[1] Here, we adopt the canonical definition of "reverse image search": using an example image to search for similar images, without any keywords provided.

[2] Embeddings are low-dimensional vectors (in this case, 512-dimensional vectors) that capture meaningful semantic associations. In the case of CLIP, both text and images can be embedded by the model into the same embedding space, and embeddings with similar cosine distance are more likely to share semantic similarities. For example, images of stop signs and the phrase "stop sign" will all have CLIP embeddings that are close to one another in the CLIP embedding space.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

natural language queries, reverse image search and multimodal inputs. The back-end, implemented in FastAPI, handles search requests by comparing query embeddings with precomputed image embeddings.

To aid those interested in using our software, we provide a tutorial for running the Digital Collections Explorer with the goal of facilitating use by researchers and practitioners in the computational humanities, as well as in galleries, libraries, archives and museums (GLAMs). Our intent is for the Digital Collections Explorer to be of use to a range of audiences, including individual researchers, curators, photo editors and even artists, such as photographers – all of whom share the challenge of searching visual collections that may not have much metadata. We demonstrate the extensibility of our Digital Collections Explorer with four different collections: two photojournalism collections provided to us by collaborators due to the collections' lack of descriptive metadata (and thus persistent difficulties searching them); a collection of 562,842 images of maps held by the Library of Congress; and a collection of a thousand born-digital PDFs produced by the federal government. In doing so, we demonstrate how the Digital Collections Explorer can facilitate searching even in the limit of no metadata. Lastly, to demonstrate the functionality of the Digital Collections Explorer, we host a public demo at https://www.digital-collections-explorer.com for searching these 500,000+ images of maps from the Library of Congress.

Contributions

This article presents several contributions:

1. We introduce our Digital Collections Explorer, a cultural heritage viewer for visual culture exploration. The system provides institutions with a robust foundation for digital collection management and discovery, while addressing key challenges in user interaction. Significantly, our Digital Collections Explorer can be spun up locally, meaning that both the machine learning embedding pipeline and the viewer can be spun up without making any data visible to the public or to any machine learning APIs.
2. Our Digital Collections Explorer implements a metadata-agnostic approach, enabling semantic search and exploration capabilities even for collections lacking traditional metadata structures. By leveraging CLIP embeddings, this approach significantly expands the accessibility of previously hard-to-search archival materials.
3. Our research contributes to the open-source community through a comprehensive implementation, available via a public repository, as well as our tutorial for applying our Digital Collections Explorer to other collections of interest. The codebase is publicly available at https://doi.org/10.5281/zenodo.15744570 and is available with a CC-BY-4.0 license.
4. We demonstrate the Digital Collections Explorer's adaptability across diverse collection types, including photographs, maps and born-digital documents.
5. We host a public demo of Digital Collections Explorer on an example collection of 562,842 digitized map images from the Library of Congress at: https://www.digital-collections-explorer.com.

Related work

In this section, we contextualize our work in relation to existing projects and literature surrounding the collections as data, multimodal cultural heritage and open-source viewers for digital cultural heritage.

Collections as data and responsible AI

We build on extensive work over the past decade to develop "Collections as Data" approaches (Padilla 2018; Padilla et al. 2019). "Collections as Data" principles emphasize "computational use of digitized and born digital collections," "lower[ing] barriers to use," "shared documentation help[ing] others find a path to doing the work," and "valu[ing] interoperability" (Padilla et al. 2019), all of which are principles that we bring with our Digital Collections Explorer.

We also draw from the related area of work surrounding responsible uses of AI for GLAMs (Lee 2023; Padilla 2020; Potter 2023). In particular, we have drawn from this literature during our development process and have chosen to emphasize the development of tooling that uses AI in order to improve access and democratize its application, while also ensuring that privacy and stewardship are emphasized through the adoption of open models and local interfaces.

Multimodal AI models

Our Digital Collections Explorer builds upon a rich body of research in multimodal search and digital cultural heritage. Recent advancements in multimodal machine learning have yielded the development of open models, such as CLIP (Radford et al. 2021) and more recently, LlaVa (Liu et al. 2024) and Molmo (Deitke et al. 2025). As one example, the open-source model Molmo was pre-trained on a dataset of 712,000 images with 1.3 million captions (Deitke et al. 2025). Similarly, CLIP (the model we adopt in this article) has been pre-trained on publicly-available image-caption pairs derived from webpages and machine learning datasets, such as the YFCC100M dataset (Radford et al. 2021; Thomee et al. 2016). Created by OpenAI, CLIP was first released in 2021 and is publicly available via GitHub and HuggingFace (in this article, we have utilized HuggingFace for access).

Multimodal cultural heritage

CLIP, along with other multimodal models, has enabled semantic alignment between text and image embeddings, facilitating a wealth of searches across language and vision. Prior work has explored the application of these models to cultural heritage collections (Barancová, Wevers, and van Noord 2023; Mahowald and Lee 2024; Smits and Kestemont 2021; Smits and Wevers 2023; Smits et al. 2025) and has demonstrated promising possibilities for improving the discoverability of large-scale visual collections, especially those with little descriptive metadata. However, challenges remain in democratizing these approaches and integrating them into user-friendly systems for viewing. Our Digital Collections Explorer addresses this challenge by prioritizing extensibility for non-experts. In the tradition of open machine learning models that can be run locally – without sharing information with proprietary AI companies – our Digital Collections Explorer is designed to use open multimodal models, ensuring that digital collections can be stewarded properly.

Open-source viewers for digital collections

Researchers and practitioners in digital cultural heritage have long contributed to the creation of open-source image viewing

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

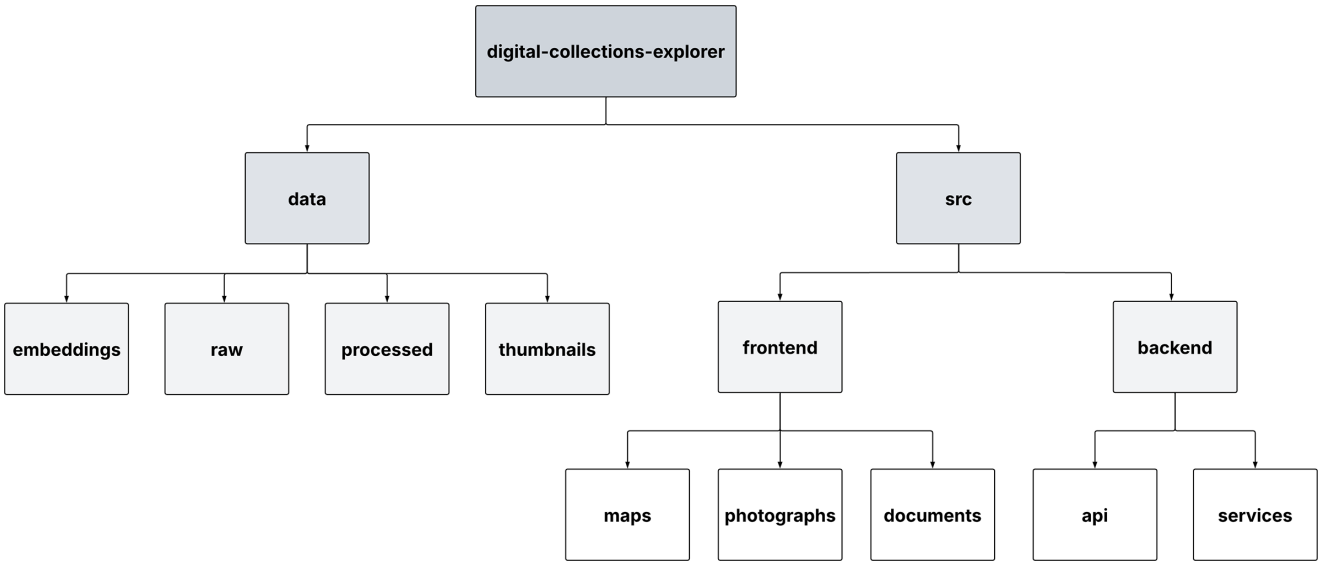


Figure 2. The directory layout of the Digital Collections Explorer codebase, showing the high-level organization into `data`, `src/frontend` and `src/backend`, which correspond to distinct functional layers of the system.

software for digital collections, enabling non-experts to spin up interfaces for viewing. Viewers, such as CollectionBuilder (Becker, Williamson, and Wikle 2020) and Omeka (Cohen 2008), provide faceted viewing options and have been heavily utilized within the digital humanities, computational humanities and library communities. Viewers, such as PixPlot (Duhaime 2020), CollectionScope (Natural History Science Visualization Group 2021) and artexplorer.ai (van der Weide and Lockhorst 2024), support visual and multimodal semantic search in a more exploratory fashion via cluster-based search. Other innovative viewers include the Vikus Viewer (Glinka, Pietsch, and Dörk 2018). Our Digital Collections Explorer builds on this movement in order to provide new modes of open-source viewing. Our solution is designed with both extensibility and scale in mind, providing semantic viewing capabilities over hundreds of thousands of items seamlessly.

Digital collections explorer: An overview

In this section, we include an overview of our system architecture, including the embedding generation pipeline, front-end and back-end components, as well as a tutorial describing how to spin up a local instance of the Digital Collections Explorer, along with a public demo.

System architecture

The Digital Collections Explorer is designed as a modular system, ensuring maintainability, scalability and ease of reuse. The overall structure of the codebase is illustrated in Figure 2, consisting of three central branches: `data`, `src/frontend` and `src/backend`.

Embedding generation pipeline

The system uses a local implementation of CLIP to generate embeddings of visual collections. This ensures privacy because no data is sent to external servers, making the system suitable for sensitive collections. Likewise, it ensures efficiency, as embeddings are generated locally, reducing dependency on external APIs and ensuring consistent performance. By default, the system loads the publicly available pre-trained model:[3]

`openai/clip-vit-base-patch32`

The generation pipeline is managed through a set of clearly defined directories within the `data` folder, as illustrated in Figure 2, which is systematically organized as follows:

- **`raw`**: This directory serves as the initial input location for the user's original collection files in their native format (e.g., JPGs, PNGs, TIFFs or PDFs).
- **`processed`**: Before embedding, certain files require pre-processing. For instance, PDFs are converted into a series of images during pre-processing, and these intermediate files are stored here.
- **`thumbnails`**: To ensure a smooth user experience in the gallery view, the system automatically generates low-resolution thumbnails for each item, which are stored in this directory for rapid loading.
- **`embeddings`**: The final output of the pipeline, the computed tensor embeddings, is saved as `.pt` files in this directory. Instead of generating embeddings with Digital Collections Explorer, users may skip the pipeline and place their pre-computed embeddings here for the system to use directly.[4]

[3] The model card for `clip-vit-base-patch32` can be found at: https://huggingface.co/openai/clip-vit-base-patch32 (in this context, a model card is a form of documentation for AI models popularized by Mitchell et al. (2019), utilized to document various dimensions of the model's training, uses, evaluation and beyond).

[4] To use pre-computed embeddings, three conditions must be met:

1) An `embeddings.pt` file containing a single PyTorch tensor of shape `[N, D]`, where `N` is the total number of items and `D` is the embedding dimension.
2) An `item_ids.pt` file containing a Python list of `N` unique string identifiers.
3) The dimensionality of the custom embeddings must precisely match the output dimension of the model specified in the `config.json`.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

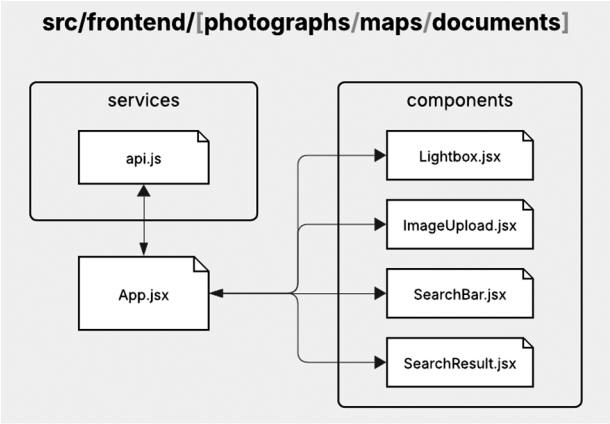


Figure 3. The component-based front-end architecture of the Digital Collections Explorer. The parent (`App.jsx`) component acts as a stateful controller, mediating between the reusable UI and an abstracted layer (`api.js`).

The model choice is fully configurable: users can swap in any Hugging Face transformers-compatible model by editing a single line in the project's `config.json`. As described in the tutorial later in this section, once a digital collection is placed in the `raw` directory, the embedding pipeline can be run with a single command.

Front-end

The user-facing interface is built using React, providing an intuitive and responsive experience. As shown in Figure 3, the architecture is centered around the `App.jsx` component, which serves as the primary container and state manager. This architecture enables several key features for the end-user, including:

- *Search interaction*, supported by the `SearchBar.jsx` and `ImageUpload.jsx` components. These components provide interfaces for natural language queries and reverse image search, respectively. An example of the landing page, as shown in Figure 4, demonstrates both text and image search functionalities.
- *A gallery view for browsing collections*, which is rendered by the `SearchResult.jsx` component. This component renders a grid of thumbnails based on the search results, as shown in Figure 5 for the query "arctic ocean."
- *Detailed image inspection*, provided by the `Lightbox.jsx` component. As demonstrated in Figure 6, this feature presents a high-resolution version of a selected item in a modal overlay.

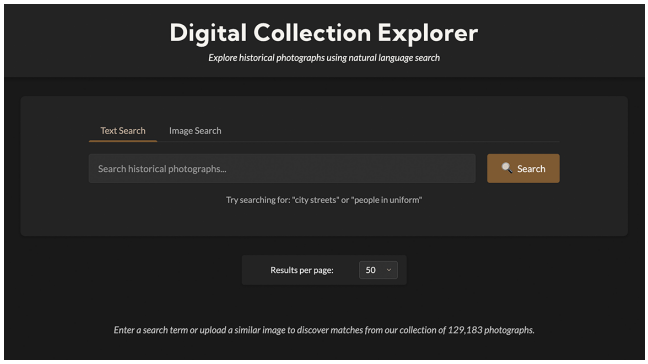
For greater modularity and ease of reuse, the front-end is structured as independent React applications for each collection type (photographs, maps and documents). Each application is self-contained, allowing developers to isolate and utilize a single front-end implementation for their specific needs. Additionally, this approach allows for collection-specific customization within a consistent structure; for example, while all collection types share common API call logic, the `frontend/documents/src/components` folder consists of a `PDFViewer.jsx` component tailored to its specific PDF content.

All communication with the back-end is handled by a dedicated service layer, `api.js`. When a user initiates a search query, the corresponding UI component notifies `App.jsx`, which then invokes the necessary function from the `api.js` service. This service manages the asynchronous API request and returns the data to `App.jsx`, which updates its state, triggering a re-render of the interface to display the results.

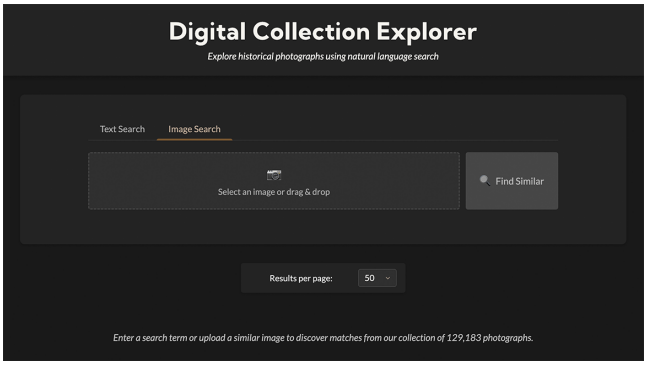
Back-end

The API server is implemented using FastAPI, a high-performance Python web framework. As depicted in Figure 2, the back-end logic within `src/backend` is divided into two core sub-directories: `api` and `services`. The `api` directory defines the public-facing endpoints that the front-end communicates with, while the `services` directory contains the core logic, such as the CLIP model inference and embedding management. This separation of concerns ensures maintainability. The front-end interacts primarily with two main endpoints: `/api/search/text` for natural language queries and `/api/search/image` for reverse image search. Both endpoints accept parameters for pagination, such as `limit` and `offset`, allowing for efficient loading of large result sets. Upon receiving a request, the back-end processes the query and returns a ranked list of relevant items. Each item in the response payload includes a unique identifier, its similarity score, and any associated metadata, providing the front-end with all necessary information for rendering.

To implement the embedding functionality, our system leverages the *Transformers* library by Wolf et al. (2019). This library provides a robust and efficient implementation of the CLIP model. By building upon this widely adopted open-source tool, we ensure



(**a**) Text search.



(**b**) Image search.

Figure 4. Examples of the landing page for the photographs collection interface, which presents an end-user with two options for searching: text search via natural language (Figure 4a) and image search (Figure 4b).

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

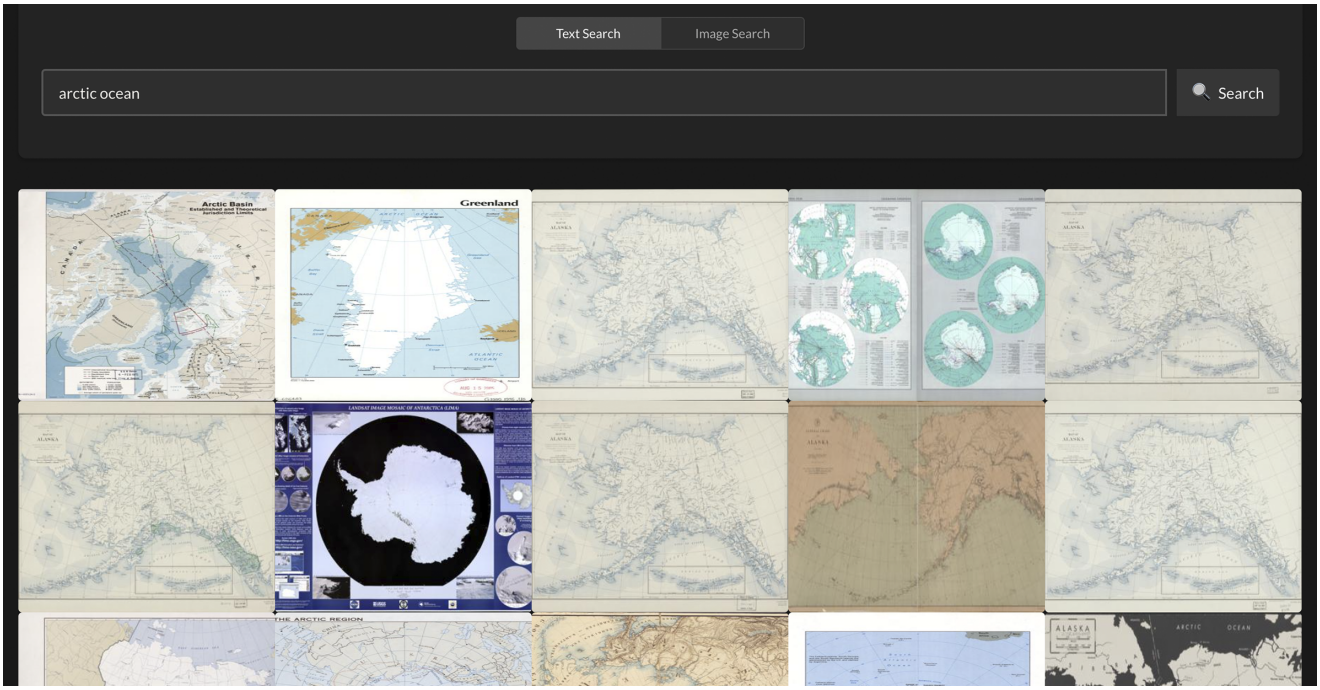


Figure 5. An example of the historical maps gallery view rendered by the `SearchResult.jsx` component in response to a user query "arctic ocean." The component's responsibility is to render a grid with thumbnails of the maps.

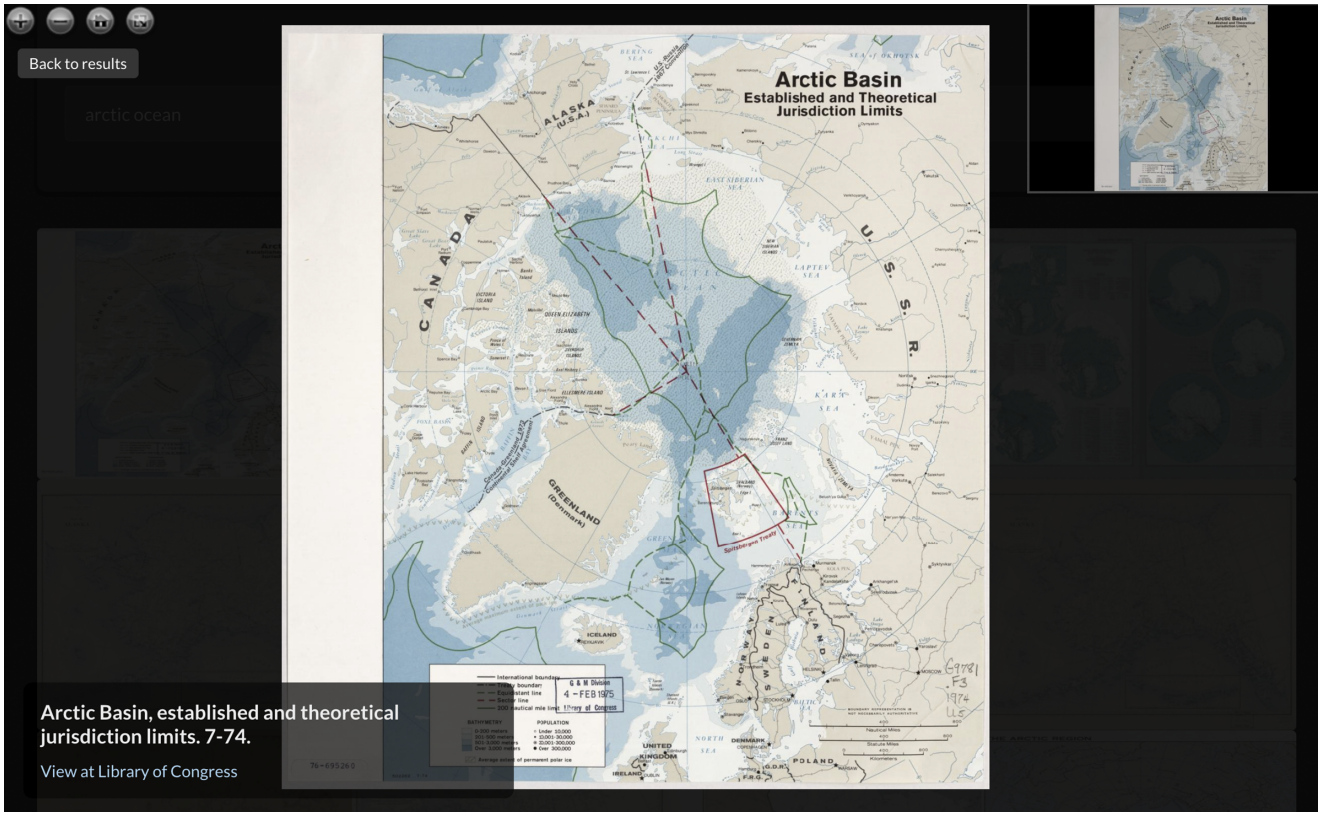


Figure 6. The lightbox view, built into the maps collection interface, enables detailed inspection of a historical map. This modal interface provides tools for zooming and panning, allowing for a detailed examination of a map's features.

that our system is not only reliable but also easily extensible, allowing for future integration of other pre-trained models from the Hugging Face ecosystem.

The core of our back-end is the retrieval engine, which implements the procedure detailed by Mahowald and Lee (2024). For any given query (either text or image), the system computes a CLIP embedding and retrieves the nearest neighbors from the pre-computed embeddings of the collection, ranked by cosine distance. Our principal contribution resides in the implementation of this methodology, transitioning it from its initial Jupyter

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

Table 1. Embedding generation times with a 2024 MacBook Pro M4 Chip with 10-Core CPU, 10-Core GPU and 16-GB Unified Memory (∗=reported by Mahowald and Lee (2024) using similar hardware).

Collection	Items	Embedding time
Library of Congress Maps	562,842 map images	Under 24 hours∗
San Francisco Chronicle Photo Collection	129,386 photographs	1 hour 27 minutes 32 seconds
Library of Congress .gov PDF dataset	1,000 PDFs (12,287 pages)	8 minutes 57 seconds
Chris Morris Photo Collection	1,025 photographs	4 minutes 5 seconds

Note: We report failures on 75 images from the San Francisco Chronicle Photo Collection (0.058%) and 17 Library of Congress PDFs (1.7%).

Notebook prototype in Mahowald and Lee (2024) to a production-ready system. This was accomplished through the design of a FastAPI application, wherein the retrieval process is delivered via a high-performance, non-blocking API endpoint. The FastAPI's asynchronous capabilities were crucial in this engineering effort, providing the necessary throughput to support scalable queries across hundreds of thousands of items with minimal latency.

Software engineering practices

The implementation of the Digital Collections Explorer adheres to established software engineering practices that prioritize maintainability and extensibility. The back-end is organized according to a layered architecture that separates HTTP routing, service logic and data modeling according to the principle of separation of concerns. Each FastAPI route is intentionally minimal. It handles only request validation and response formatting, while the core logic is implemented in separate service modules, such as `clip_service.py` and `embedding_service.py`. This design follows the service layer pattern and single responsibility principle, allowing each component to serve a distinct purpose.

Working with the digital collections explorer: A tutorial

Whether one is utilizing historical photographs, historic maps or born-digital documents, the Digital Collections Explorer offers a streamlined setup process and scalability for customization with a wide range of digital collections. This section demonstrates how researchers can easily and efficiently set up the system to meet their specific collection requirements. Though the Digital Collections Explorer requires knowledge of the command line, we have made every effort to minimize the number of commands required to utilize our codebase.

System setup

The system initialization process is designed to be straightforward by assigning the specific collection type as an argument. The following examples demonstrate the setup process for different collection types:

Photographs
```
npm run setup -- --type=
    photographs
```
Configures a gallery interface with grid and masonry layouts,[5] optimized for large-scale image browsing.

Maps
```
npm run setup -- --type=maps
```
Implements an OpenSeadragon viewer for high-resolution zoomable maps with smooth pan and zoom capabilities.

Documents
```
npm run setup -- --type=
    documents
```
Provides a temporal navigation interface with a document viewer optimized for PDFs extracted from web archives.

Each setup command automatically configures the appropriate front-end components and back-end services optimized for the specific collection type.

We note that the Digital Collections Explorer can also be utilized to explore collections of mixed types. In this case, we recommend that the most appropriate setting be chosen (for example, a collection of mostly photographs with some maps is compatible with the "photographs" setting). Sometimes, a different category might be the best choice: for a collection of very high-resolution photographs, the "maps" setting might be most appropriate in order to take advantage of the OpenSeadragon viewer.

Data preparation and embedding generation

For a given digital collection, we begin by placing the collection in the `data/raw/` directory; however, this location can be configured in the `config.json`. The system recursively retrieves images from subdirectories, so any existing directory structure is acceptable, so long as all of the images exist nested within `data/raw/`. The system supports common image formats, including JPG, PNG, TIFF or PDFs (PDFs are split at the page level and converted to images as part of running this pre-processing pipeline). Embeddings are generated by running:

```
python -m src.models.clip.generate_
    embeddings
```

This command processes all images in the `data/raw` directory and creates embeddings in the `data/embeddings` directory. We report embedding generation times for multiple collection examples in the next section of the article on case studies. For instance, as shown in Table 1, the San Francisco Chronicle Photo Collection, comprising 129,386 photographs, was processed in 1 hour and 28 minutes on a single MacBook Pro.

Starting the server

After embedding generation, the back-end server is launched to provide API endpoints for search and exploration by running the following command:

```
python -m src.backend.main
```

The API server will then start at http://localhost:8000.

[5] A masonry layout is one in which each image is proportionally adjusted to the same width, and images are sorted in columns of even width.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

Front-end customization and build process

To enable front-end customization, we have active development with hot reloading. Once the back-end server is up, starting the front-end development server can be accomplished with:

```
cd src/frontend/[photographs|maps|
documents]npm run dev
```

These commands start a temporary development server at http://localhost:5173 with hot-reloading enabled. The port number is configurable and used only for local development. In production, the front-end assets are served by the back-end, allowing multiple collections to be deployed under the same instance. For production deployment, front-end assets must be built using the following command:

```
npm run frontend-build
```

Then restart the back-end server to serve the updated front-end assets. The build process is only required when deploying to production environments (such as cloud servers) or when generating optimized JavaScript bundles for enhanced performance. For local development and testing purposes, running the front-end development server is sufficient.

Publicly hosting a digital collections explorer

It is straightforward to host a web application of Digital Collections Explorer for public access on cloud services such as Amazon Web Services (AWS). Although manual setup can be done by following the tutorial above, we strongly recommend this containerized approach due to its significant advantages in ensuring environmental consistency, simplifying dependency management and enhancing security. As a result, we provide a `Dockerfile` that serves as the cornerstone for both deployment and scientific reproducibility. This Dockerfile encapsulates the application stack – the Python back-end, the compiled JavaScript front-end and all package dependencies. By doing so, it creates a portable image of the entire system. The general deployment process involves:

1. *Build the Docker Image.* The image should be tailored to a specific collection type by passing the `–build-arg` flag during the build process:

   ```
   docker build --build-arg COLLECTION\_TYPE
      =<type> -t <image-tag> .
   ```

2. *(Optional) Push to a Container Registry.* For distribution, the newly created image can be pushed to a container registry, such as Docker Hub or AWS ECR:

   ```
   docker push <image-tag>
   ```

   This step is not required if the image is built directly on the target machine.
3. *Run the Container.* Finally, the application is launched by running the container from the Docker image.

   ```
   docker run -p 8000:8000 -v ./data:/app/
      data <image-tag>
   ```

Public demo

For those who would like to experiment with an instantiation of Digital Collections Explorer, we host a public demo at: https://www.digital-collections-explorer.com. This demo supports searching over 562,842 map images from the Library of Congress – one of our case study collections described in detail in the next section.

To create this live, interactive demonstration of this system, we deployed Digital Collections Explorer on an AWS EC2 instance using the following process. First, we built a Docker image on a MacBook Pro M4 and pushed it to our public Docker Hub repository: https://hub.docker.com/repository/docker/hinxcode/digital-collections-explorer. Following this, we provisioned an AWS EC2 c6gd.large instance ($0.08/hour), pulled the image, and launched the application by running "`docker run`."

For this specific deployment, we diverged from the standard setup in two key ways. First, instead of using the embedding generation pipeline, we directly used the pre-computed embeddings provided by Mahowald and Lee (2024). Second, to augment the pre-computed embeddings with essential metadata, we developed a Library of Congress data preprocessing script. Our Python script, `create_loc_assets.py`,[6] processes the original image identifiers and performs a record lookup against `merged_files.csv` to generate two key assets: 1) a new index file, `item_ids.pt`, which replaces the original identifiers with stable keys while preserving their sequence and 2) a `metadata.json` file that maps each key to its corresponding metadata, including a direct link to the item's entry in the Library of Congress. These output files are then placed within the `/data/embeddings` directory, adhering to the file structure outlined previously. We note that our approach is not meant to replace traditional forms of metadata, given their immense value in exploratory search. Rather, the Digital Collections Explorer is an additional layer of discoverability that can be utilized to draw new insights across collections by using CLIP as a way of identifying similarity. In cases of photographs or other visual materials that do not have much associated metadata, or any at all – such as photo morgues or unprocessed collections – the Digital Collections Explorer can provide a base level of discoverability. In the case of our demo, the metadata is not embedded with CLIP. Rather, we retain the connection to each map's Library of Congress metadata for each search result so end-users can learn more from an authoritative source.

Discussion: Case studies

As detailed in Figure 1, the Digital Collections Explorer employs a three-stage pipeline for collection exploration: 1) Data Preparation, 2) Embedding Generation, and 3) Search and Exploration. In this section, we describe our case studies with photographs, maps and born-digital documents using the Digital Collections Explorer.

Data preparation

Collections are ingested into the system by placing images in a designated directory. For this study, we used four datasets to demonstrate the system's capabilities:

1. A collection of 1,025 photographs of Russia provided by the photojournalist Christopher Morris. Morris shared this subset of photographs with us directly via a hard drive for the purposes of experimenting with new ways of searching his photographs.
2. A large-scale collection of 129,386 photographs from the San Francisco Chronicle.

[6]The script for Library of Congress maps preprocessing is included in the project repository at: `scripts/create_loc_assets.py`.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

Table 2. Total processing times (including parsing, thumbnails generation and embeddings generation) with a 2024 MacBook Pro M4 Chip with 10-Core CPU, 10-Core GPU and 16-GB Unified Memory.

Collection	Items	Total processing time
San Francisco Chronicle Photo Collection	129,386 photographs	3 hours 20 minutes 19 seconds
Library of Congress .gov PDF dataset	1,000 PDFs (12,287 pages)	28 minutes 24 seconds
Chris Morris Photo Collection	1,025 photographs	10 minutes 41 seconds

Note: Here, we omit the Library of Congress maps because we used the embeddings from Mahowald and Lee (2024).

3. The Library of Congress dataset of 1,000 random .gov PDFs extracted from the Library of Congress web archives, amounting to 12,287 pages of PDFs in total (the value of searching these PDFs visually has been described by Lee and Owens (2021)).
4. 562,842 images of maps held by the Library of Congress, retrieved using the Library of Congress API by Mahowald and Lee (2024).

Embedding generation

In Table 1, we report the times to generate embeddings for the collections with a 2024 MacBook Pro M4 Chip with 10-Core CPU, 10-Core GPU and 16 GB; in Table 2, we report the total processing times (including parsing, thumbnails generation and embeddings generation) with the same machine. As reported, the Digital Collections Explorer can scale to hundreds of thousands of images in a tractable fashion. We note that these times do not scale precisely linearly for multiple reasons, including file size (and thus re-sizing during embedding generation) and different required pre-processing steps (such as PDF parsing).

Search and exploration

Here, we present example search results using two of our case studies: the Library of Congress maps and .gov PDFs, both of which are public domain collections (we have withheld screenshots of our other case studies due to copyright considerations).

In Figure 7, we present two natural language searches against the 1,000 .gov PDF dataset from the Library of Congress. Here, searches for "redacted document" and "multicolor graphs" result in ranking the PDF pages according to relevance to the search performed. As evidenced by these examples, we are able to query the visual features of the documents, rather than just their textual content.

In Figure 8, we present searches against the 562,842 map images from the Library of Congress API. Figure 8a shows a natural language search of "tattered and worn map"; we note that these results match the results from Figure 5a in Mahowald and Lee (2024), thereby confirming our ranking logic. This time, however, the searches can be performed in a production-ready user interface, rather than in Jupyter notebooks. Figure 8b shows a reverse image search returning relevant results. Any user can reproduce these searches in Figure 8 and try others using our demo at: https://www.digital-collections-explorer.com.

We note that some searches work better than others. For example, the Digital Collections Explorer supports searching over visual content in the .gov PDFs – figures, images, etc. – but does not support semantically searching the text. For a more thorough investigation of the strengths and weaknesses in the search methods we employ, we refer the reader to Mahowald and Lee (2024). In particular, this article includes a detailed analysis of specific search examples for the Library of Congress maps, including searches with higher and lower accuracy. We note that searches for abstract visual features tend to perform better than searches for specific locations or landmarks.

Maintenance and development

We recognize that maintaining digital infrastructure is just as important as initial development. One central component of maintenance is computing costs. Our current demo of the Digital Collections Explorer for over half a million map images is currently hosted on an Amazon EC2 instance for only $0.08 an hour, or $1.84 daily. Given this low cost, we believe that other users of the Digital Collections Explorer should be able to spin up versions on their own collections without having computing costs as the primary barrier to sustainability (a challenge that the authors understand first-hand from having worked on projects with significant monthly computing bills).

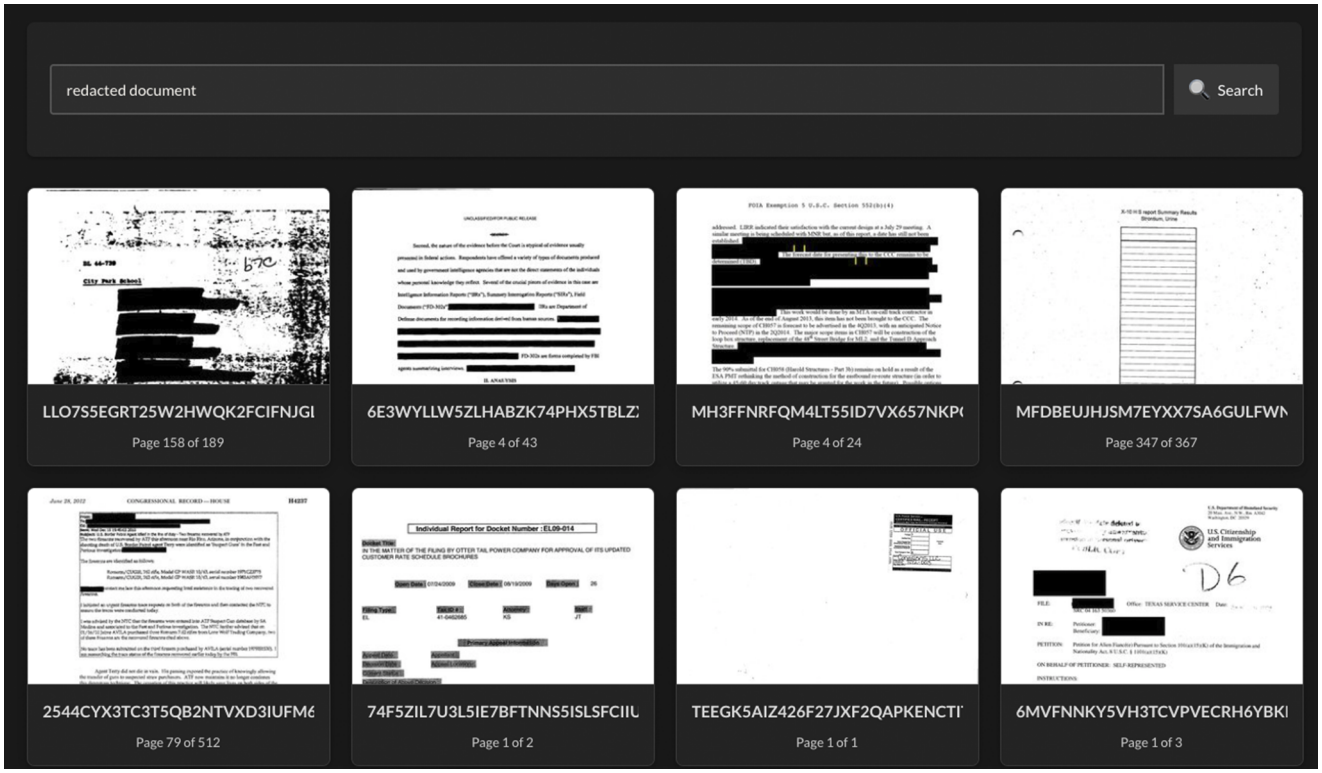
Regarding active maintenance of our codebase, we plan to make updates according to the directions we articulate in the next section. Moreover, we plan to perform routine testing of new multimodal models and hope to incorporate them over time so that the Digital Collections Explorer remains maximally useful. We recognize that the landscape of AI models is changing extremely rapidly, and our goal is to prioritize keeping the Digital Collections Explorer modular enough as to enable us to periodically swap in new model options and allow end-users to test how different models impact search results over a given collection.

Conclusion and future work

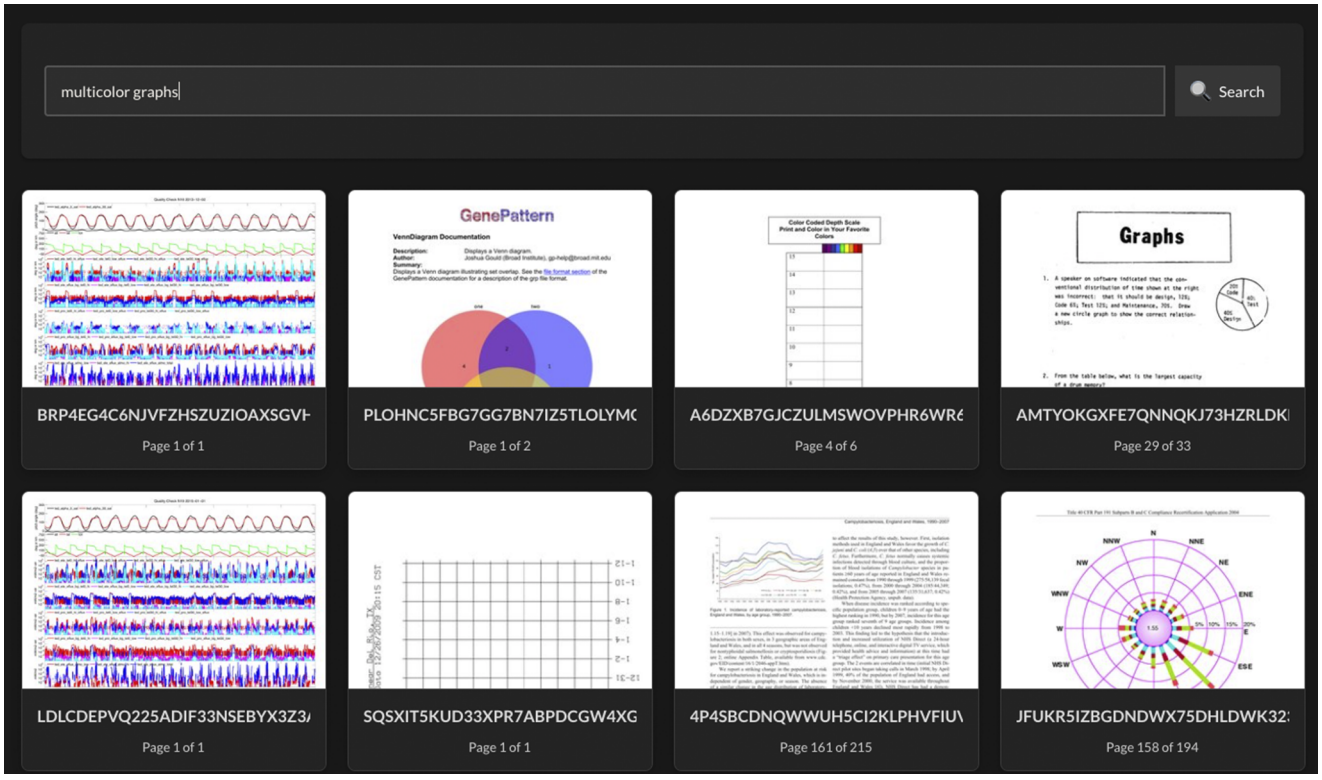
In this article, we have introduced our Digital Collections Explorer. With this open-source platform, researchers and practitioners can spin up a search interface on top of a digital collection of interest for enhanced visual discovery using both textual and visual inputs. Our work builds on the emerging body of research demonstrating the value of multimodal search and analysis for digital collections held by libraries, archives and museums. Our Digital Collections Explorer extends this work by providing tooling to non-experts, enabling them to explore a digital collection in a multimodal fashion in just a few steps on a staff-issued laptop, such as a current-generation MacBook Pro. Our platform is designed to scale to hundreds of thousands of images in this context. We have released the Digital Collections Explorer as open-source software under a CC-BY-4.0 license.

In order to preserve the privacy of digital collections, all steps of the Digital Collections Explorer can be run locally, from pre-processing to viewing, meaning that no data is transferred via proprietary APIs or publicly-visible endpoints. The Digital Collections Explorer is intended to be particularly useful as a method for exploring collections with little-to-no descriptive metadata.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press



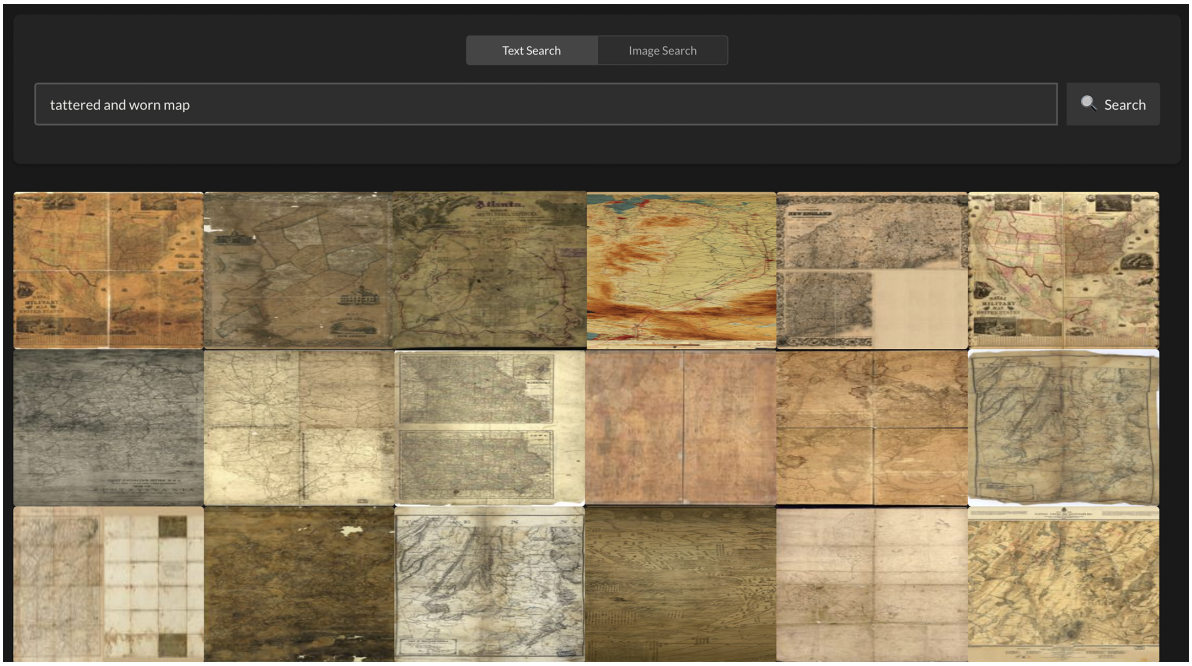
(**a**) A search of "redacted document"



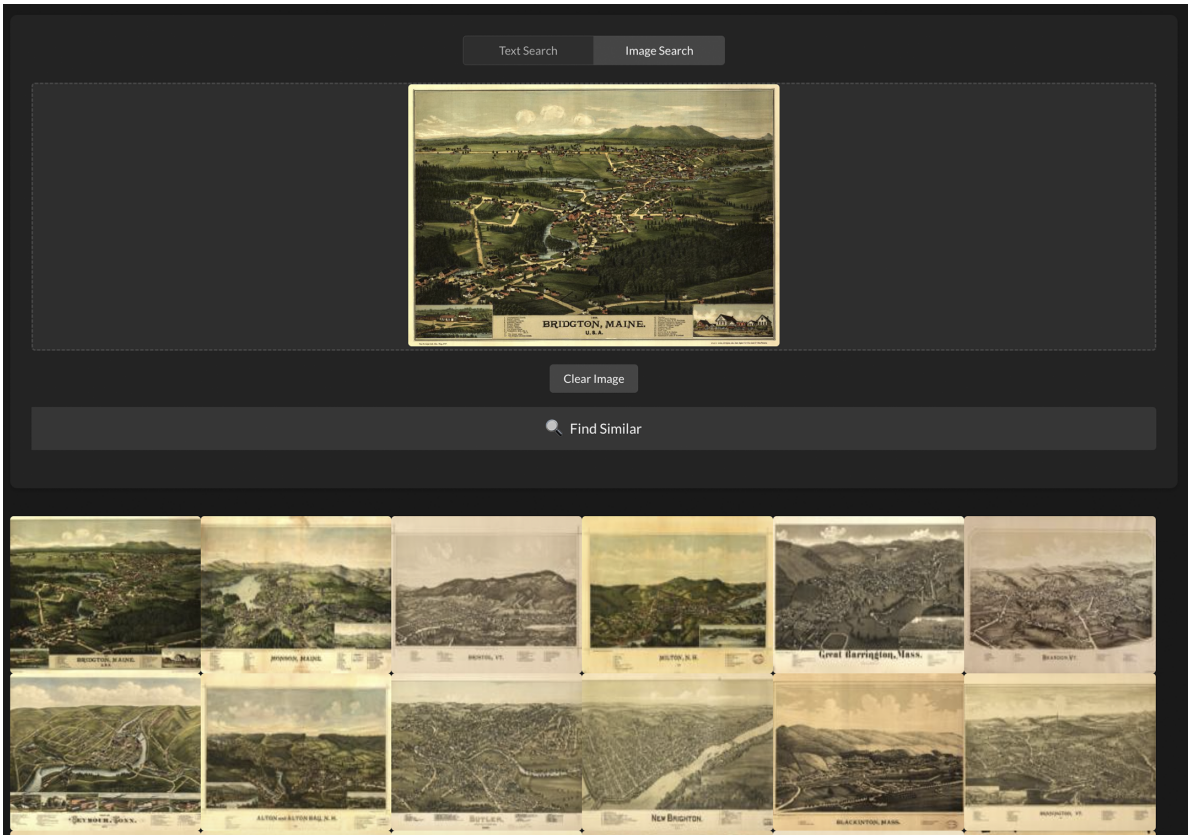
(**b**) A search of "multicolor graphs"

Figure 7. Search results for two different natural language queries across the 1,000 Library of Congress .gov PDFs demonstrating the effectiveness of semantic retrieval: (a) "redacted documents" and (b) "multicolor graphs." The filenames shown refer to the PDF filenames (given by the hash in the Library of Congress web archives).

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press



(a) A search of "tattered and worn map."



(b) A reverse image search, with the input image shown at the top.

Figure 8. Searches against the 562,842 map images from the Library of Congress API. (a) shows a natural language search of "tattered and worn map" and (b) shows a reverse image search with a panoramic map of 1888 Bridgerton, Maine, from the Library of Congress's collections (http://hdl.loc.gov/loc.gmd/g3734b.pm002434). These results can be reproduced in our demo: https://www.digital-collections-explorer.com.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press

Throughout this article, we have introduced the system architecture, walked through a tutorial of how to use the Digital Collections Explorer, and presented case studies across maps, photojournalism collections and born-digital PDFs extracted from web archives. All of our code is available at: https://doi.org/10.5281/zenodo.15744570, and our publicly-available demo is available at: https://www.digital-collections-explorer.com.

Our current version of the Digital Collections Explorer is informed by our collaborations with stakeholders, including photo editors, photographers and curators whose collections comprise our four case studies. The Digital Collections Explorer was inspired by shared challenges articulated by all stakeholders about the current limitations of searching their visual collections, which lack metadata. Our current version has served as an exploratory mechanism for re-interpreting their collections. Their feedback has inspired a number of different directions of future work.

First, we plan to provide support for additional input modalities, such as audio or video. Second, we plan to incorporate different multimodal models into our system beyond the one default CLIP model – including models finetuned for cultural heritage collections. Third, we plan to experiment with models that are better-suited for searching text representations. Fourth, we hope to experiment with new modes of presenting metadata, as well as integrating external metadata sources and knowledge bases to enhance search capabilities. Fifth, on the basis of our continued collaboration with stakeholders, we will continue to work to scale up the Digital Collections Explorer to support viewing millions of images – a challenge that will require articulating hardware requirements and pre-processing runtime expectations. Along these lines, we will also plan to incorporate more options for GPU utilization in the embedding pipeline. Sixth, we hope to include ways of using the Digital Collections Explorer that do not require knowledge of the command line.

Lastly, we will collect input and feedback from researchers and practitioners, which will inform future updates to the Digital Collections Explorer. We believe that user studies with curators and end-users would be extremely valuable in clarifying many of the future directions of work articulated above. In doing so, we also hope to clarify the ways in which the Digital Collections Explorer might be utilized in forms of humanistic analysis that go beyond search and discovery – for example, drawing quantitative insights into frequencies of visual features, or systematically identifying patterns across a collection. By conducting studies with humanists who work with digital collections of visual culture, we will further elucidate this direction. We welcome contributions from the computational humanities and digital cultural heritage communities via submitting pull requests to our GitHub repository.

Acknowledgements. We are grateful to the University of Washington Information School for supporting this work. In particular, we thank the Center for the Advancement of Libraries, Museums, and Archives (CALMA), for providing a research grant for the photojournalism portion of this work. We thank Chris Morris for his eagerness to share his photographs of Russia with us as an initial case study. We thank Kira Pollack for facilitating this collaboration, working with us and providing invaluable feedback during this process. We also thank Nicole Fruge for providing us photographs from the San Francisco Chronicle as another case study for our system and for providing us invaluable feedback as well.

Data availability statement. All code for the Digital Collections Explorer is available at: https://doi.org/10.5281/zenodo.15744570. In this GitHub repository, readers can find detailed instructions on how to install the Digital Collections Explorer and use it to view their own digital collections. The publicly-available datasets we have utilized in this article are available as follows:

1. The 1,000 .gov PDF dataset by the Library of Congress is available at: https://lccn.loc.gov/2020445568.
2. The digitized maps are available through the Library of Congress's API.

Author contributions. Our author contributions following the Contributor Role Taxonomy[7] are as follows: Conceptualization: Y.-H.H., B.L.; Data curation: Y.-H.H., B.L.; Formal analysis: Y.-H.H.; Funding acquisition: B.L.; Investigation: Y.-H.H., B.L.; Methodology: Y.-H.H., B.L.; Project administration: B.L.; Resources: B.L.; Software: Y.-H.H.; Supervision: B.L.; Writing - original draft: Y.-H.H., B.L.

Funding statement. This research was supported by the University of Washington Information School, including a grant from the Center for Advances in Libraries, Museums, and Archives.

Competing interests. The authors declare none.

Ethical standards. The research meets all ethical guidelines, including adherence to the legal requirements of the study country. In addition, we have followed best practices from responsible AI in creating the Digital Collections Explorer.

References

Barancová, Alexandra, Melvin Wevers, and Nanne van Noord. 2023. "Blind Dates: Examining the Expression of Temporality in Historical Photographs." *Computational Humanities Research Conference 2023 Proceedings*, pp. 490–499. https://ceur-ws.org/Vol-3558/paper5790.pdf.

Becker, Devin, Evan Williamson, and Olivia M. Wikle. 2020. "Collectionbuilder-CONTENTdm: Developing a Static Web 'Skin' for CONTENTdm-Based Digital Collections." *Code4Lib Journal* 49, https://journal.code4lib.org/articles/15326.

Cohen, Dan. 2008. "Introducing Omeka." https://doi.org/10.13021/MARS/2890.

Cordell, Ryan Charles. 2020. "Machine Learning + Libraries: A Report on the State of the Field." [in English (US)]. LC Labs. Library of Congress.

Deitke, Matt, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi 2025. "Molmo and PixMo: Open Weights and Open Data for State-of-the-Art Vision-Language Models." In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, 91–104. Nashville: TN, USA. https://doi.org/10.1109/CVPR52734.2025.00018.

Duhaime, Douglas. 2020. "Pixplot." https://github.com/YaleDHLab/pix-plot.

Glinka, Katrin, Christopher Pietsch, and Marian Dörk. 2018. "Vikus Viewer." https://github.com/cpietsch/vikus-viewer.

Lee, Benjamin Charles Germain. 2023. "The "Collections as ML Data" Checklist for Machine Learning and Cultural Heritage." *Journal of the Association for Information Science and Technology* 76, no. 2: 1–12. https://doi.org/10.1002/asi.24765.

Lee, Benjamin Charles Germain, 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. https://api.semanticscholar.org/CorpusID:257159777.

Liu, Haotian, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024. "Improved Baselines with Visual Instruction Tuning." In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 26286–96. Seattle: WA, USA. https://doi.org/10.1109/CVPR52733.2024.02484.

[7] https://credit.niso.org/

Mahowald, Jamie, and Benjamin Charles Germain Lee. 2024. "Integrating Visual and Textual Inputs for Searching Large-Scale Map Collections with Clip." *Computational Humanities Research Conference 2024 Proceedings*, pp. 528–547. https://ceur-ws.org/Vol-3834/paper17.pdf.

Mitchell, Margaret, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. "Model Cards for Model Reporting." In *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT*'19*, 220–9. Atlanta, GA: Association for Computing Machinery. https://doi.org/10.1145/3287560.3287596.

Natural History Science Visualization Group. 2021. "American Museum of Collectionscope." https://github.com/amnh-sciviz/collectionscope#read0me.

Padilla, Thomas. 2018. "Collections as Data: Implications for Enclosure." *College & Research Libraries News* 79, no. 6: 296. https://doi.org/10.5860/crln.79.6.296.

Padilla, Thomas. 2020. *Responsible Operations: Data Science, Machine Learning, and AI in Libraries*. OCLC. Last Modified: 2020-5-12. Dublin, Ohio. https://doi.org/10.25333/xk7z-9g97.

Padilla, Thomas, Laurie Allen, Hannah Frost, Sarah Potvin, Elizabeth Russey Roke, and Stewart Varner. 2019. "Final Report—Always Already Computational: Collections as Data. Version 1." https://doi.org/10.5281/zenodo.3152935.

Potter, Abigail. 2023. "Introducing the LC Labs Artificial Intelligence Planning Framework." https://blogs.loc.gov/thesignal/2023/11/introducing-the-lc-labs-artificial-intelligence-planning-framework/.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever, 2021. "Learning Transferable Visual Models from Natural Language Supervision." In *International Conference on Machine Learning*. virtual. https://api.semanticscholar.org/CorpusID:231591445.

Smits, Thomas, and Mike Kestemont. 2021. "Towards Multimodal Computational Humanities: Using Clip to Analyze Late-Nineteenth Century Magic Lantern Slides." *Computational Humanities Research 2021 Proceedings*, pp. 149–158. https://ceur-ws.org/Vol-2989/short_paper23.pdf.

Smits, Thomas, 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* 149–158. https://doi.org/10.5334/johd.284.

Smits, Thomas, 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, no. 3: 1267–80. https://doi.org/10.1093/llc/fqad008.

Thomee, Bart, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. "Yfcc100m: The New Data in Multimedia Research." *Communications of the ACM*, 59, no. 2: 64–73. https://doi.org/10.1145/2812802.

Weide, Stefan van der, and Sjors Lockhorst. 2024. "A Semantic Search for Artworks." https://blog.lockhorst.dev/projects/art-search.

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. "Huggingface's Transformers: State-of-the-Art Natural Language Processing." Preprint, arXiv:1910.03771. https://arxiv.org/abs/1910.03771.

https://doi.org/10.1017/chr.2025.10017 Published online by Cambridge University Press