Open Library of Humanities Journal

Sanderson, R 2024 Implementing Linked Art in a Multi-Modal Database for Cross-Collection Discovery. *Open Library of Humanities*, 10(2): pp. 1–25. DOI: https://doi.org/10.16995/olh.15407

Open Library of Humanities

Implementing Linked Art in a Multi-Modal Database for Cross-Collection Discovery

Robert Sanderson, Yale University, New Haven, CT 06520, USA, robert.sanderson@yale.edu

Yale University has implemented a knowledge graph-based discovery system, that brings together the various art, natural history, archival, conservation and bibliographic collections using Linked Open Usable Data standards such as Linked Art and IIIF. This system comprises more than 41 million records, which would expand to more than 2 billion RDF triples. This paper presents the lessons learned from the five-year effort to establish usability of linked data structures across the organization, and the technologies needed to make use of the knowledge in a performant way. It also considers the appropriate design paradigms for front end applications which make the graph easily and intuitively accessible to researchers and the public, including the necessity of consistency in data modeling, that records are an essential concept worth maintaining in multi-modal systems, and the use of hypertext and web caches to maintain the separation between systems.

Introduction

Between 2018 and 2023, Yale University designed, implemented, and published a novel cross-collection and cross-domain discovery application to enable unified digital access to the cultural heritage items in its museums, libraries, archives, and special collections. This application, called LUX,[1] was built using Linked Open Usable Data (LOUD) (Raemy, 2024; Sanderson, 2018). The fundamental paradigm was to ensure that the collections were presented in a connected and coherent fashion, rather than merely aggregating the records. Prototypes were built and evaluations performed throughout, to find the most appropriate solutions around data, technology and the interactions between systems and people—be they end users, data and software engineers, or content providers. The resulting application is in active use by faculty, staff, students, and the general public, who use it to discover and engage with the collections held by the university. The code is also available via the GitHub platform.[2]

This article reports on the implementation and analysis performed to evaluate the overall utility and usability of Linked Data for cultural heritage discovery systems. We show that, by following design principles that emphasize usability over semantic completeness, implementations can be richer and provide generous interfaces to the benefit of users. It is divided into sections based on the three core areas of work: the data, the technologies used, and the design of the front-end application.

Linked Art Data

The first aspect to consider for a cross-domain discovery system that will provide access to multiple collections, is the availability, quality, and feasibility of modeling all existing data from the systems into a coherent and connected whole. The resulting aggregation needs to be understandable to users interacting with it as a single platform, rather than a Frankensteinian horror with disjointed parts roughly sewn together. This section discusses the history, requirements and resulting selection and customization of the Linked Art data model for use as the backbone of the LUX application (Sanderson, et al. 2024).

Background

Starting in 2009, Yale embarked upon a cross-collection discovery platform (Bellinger, 2011) for its museum collections using technologies available at the time like Solr (Apache, 2024), XML (W3C, 2008) and OAI-PMH (OAI, 2015). For various reasons, this project was not maintained but it did serve as a starting point and an experiment that

[1] Available at https://lux.collections.yale.edu/.

[2] Available at https://github.com/project-lux/.

the LUX development project took into account. The initial prototype for LUX used these same technologies, but with the intent of extending the data to include libraries and archives, and to reconcile references to people, organizations, places, and concepts across the collections through the use of both student effort and automation. This prototype demonstrated the limitations of the technology stack and data structures, quickly becoming overwhelmed by the use of identifiers for every entity referenced in the records. Moreover, it was unable to provide the desired interactions either for cross-record searching, or the presentation of the reconciled contextual entities as individual pages. This led to the establishment of a set of foundational requirements with which to assess data structures and the technologies that would support them.

Requirements

The initial requirements for the data were generated to evaluate the feasibility of the desired interactions, given the existing data and the estimated ability over the course of the project to reconcile references across collections and datasets. This practical, results-driven target was extremely valuable, as it avoided many cross-domain discussions about metadata standards: instead of comparing standard A with standard B, both were independently and relatively objectively compared to the same set of requirements.

A summary of the core initial requirements and the intent of their inclusion are as follows:

1. The data must describe all collection items, as well as the people and organizations, places, and subjects that they reference, to provide context for the items.
2. There must be a single, normalized record for each item and referenced entity, to bring information from across collections together and provide a "hub" page in the user interface, through which the user can discover related items.
3. The user must be able to navigate between item pages and hub pages seamlessly, understanding the relationship between the item and the hub page entity, but each must be independently understandable.
4. All significant fields of the underlying source records must be represented in a clear fashion in the shared data structure, respecting the uniqueness and value of the different domains while enabling the presentation of a single, coherent interface, to ensure that all collections are well represented.
5. Users must be able to search for collection items using the information about both them and their related entities, to allow precise targeting of queries and to expand upon functionality already available within the individual collections' discovery systems.

Although not an absolute requirement, it was readily and enthusiastically acknowledged that if the data structure was easy to understand and use by software engineers, then more time would be available to work on LUX itself because there would be less investment needed to understand an arcane or inconsistent specification.

These requirements, plus a prototype built using the Yale Center for British Art dataset (YCBA, 2024) and the open-source Arches platform (Farallon, 2024), demonstrated the importance of the knowledge graph paradigm and facilitated the transition from a custom XML schema to the Linked Art metadata application profile.

Linked Art Data Model

The Linked Art data model was initially developed for the Mellon-funded American Art Collaborative (AAC, 2017) project, and subsequently adopted by the J. Paul Getty Trust as a basis for their linked data efforts. It was then brought back to the community for discussion and engagement, with support from projects funded by the Kress Foundation and the Arts and Humanities Research Council in the UK.

It is built as a metadata application profile of the CIDOC Conceptual Reference Model (CIDOC-CRM) (CIDOC, 2024) plus minimal extensions, with the intent to reduce the number of choices that developers need to make, for increased ease of use, understanding and interoperability. Through use of CIDOC-CRM, it inherits a well-discussed and standardized conceptual model and ontology of classes and relationships, and thus the development of Linked Art focused on aligning that model with real-world practices, data and use cases. It is necessary to briefly outline the features of the model before turning to its appropriateness for the product and how well it meets the requirements set out.

Classes

Linked Art uses a small number of classes from CIDOC-CRM and provides more memorable names for them for developers to use in data—rather than requiring the memorization of arbitrary numbers. If more specific information is available, for example to distinguish between a painting and a sculpture, then this information is conveyed as a classification at the vocabulary level, rather than as a class at the ontology level. The classes align well with the requirements for LUX, and maintain the separation desired for both "item" and "hub" entities.

Collection items are either physical objects using the class HumanMadeObject, or digital files which are expressed with the class DigitalObject. People and organizations use the classes Person and Group respectively, and geographic places are well

accommodated by the Place class. Concepts or subjects have a wider range of classes, based on the nature of the conceptual entity, and rely on the specific classes of Language, Material, and MeasurementUnit, with the broader class Type used for general subjects. This fulfilled the baseline requirements; however, in the assessment, the bibliographic notion of object (or "holding" in library parlance) being separate from the work became important, as well as events that took place using the collection items, such as exhibitions.

In order to distinguish between multiple physical or digital copies of a particular text without repeating the information in every record, and similarly between multiple copies of a particular piece of visual content, Linked Art has the notion of intellectual works in the same way as existing bibliographic ontologies and models such as the Library of Congress's BibFrame (LOC, 2024), or IFLA's LRM or FRBR (IFLA, 2018). Unlike those more complex systems, Linked Art uses only two classes: LinguisticObject for works that are primarily based on language and intended to be read, and VisualItem for works that are primarily based on image and intended to be seen. A single instance of a LinguisticObject, such as the text of a book on philosophy, can be carried by multiple physical and/or digital objects, allowing the holding and the bibliographic information to be connected. The same applies with visual content, such as multiple casts of a sculpture or prints of a photograph.

In order to deal with exhibitions and other activities such as provenance, Linked Art has a broad class of Activity. This can then have classifications that are more specific to the sort of activity described, and connects the items used, the people and organizations that participated in the activity, the place where it was carried out, and the time or dates when it occurred.

The last class needed is for dealing with archival collections, departmental or institutional collections, and the sets of objects used in an exhibition. The underlying CIDOC-CRM ontology does not have the capability of including both physical and digital items, let alone conceptual works or other classes, and thus the Linked Art metadata application profile has a small extension that defines a class called Set. This class represents a mathematical set that can have members of different types.

For example, the painting 'The Artist's Garden in Giverny' (LUX, 2024) is a HumanMadeObject, produced by the Person: Claude Monet. It is classified as being a particular Type: a painting, and is made of the Material: oil paint. It shows a VisualItem, which in turn depicts the Type: flowers and the Place: Giverny. It has a home page, represented as a DigitalObject at its home museum of the Yale University Art Gallery. It is a member of the Set of objects which represents the European Art Collection at

the Gallery, which is in turn curated by an Activity carried out by the Group which represents the European Art department.

These classes more than fulfilled the baseline requirements and helped to clarify understanding across domains, by expressing data in a foundational and conceptual model; rather than trying to cross-walk between domain-specific models, which has been the downfall of many projects throughout information science history.

Relationships

The relationships between the core classes are more extensive and express much of the knowledge in the graph but are still few enough to be easily understood and used.

Instances of any class can be member_of a Set, and can be classified_as a Type, including instances of Set and Type respectively. Place instances can be part_of a larger Place, allowing the description of a spatial hierarchy, but otherwise do not have relationships out to other classes. Similarly, Type instances and other concepts can have a broader Type, and are conceived into existence through an activity, which can be carried_out_by a Person or Group, took_place_at a Place, and can be influenced_by entities of any class, but otherwise do not have other relationships. Instances of Person and Group are born or formed_by an event embedded within their record, which in turn took_place_at a Place. They can also have a residence at a Place, and can be member_of a Group. This leaves the core relationships to be carried by the main entities of interest: the collection items and the intellectual works.

HumanMadeObject instances can carry a LinguisticObject or show a VisualItem, and in parallel a DigitalObject can digitally_carry a LinguisticObject or digitally_show a VisualItem. Digital collection items are created_by an embedded activity, and physical items use the produced_by relationship. While there is a relationship for the destruction of an item, it would no longer be in the collection, and hence we do not need this in the Yale use case. Objects that enter the cultural record by being found rather than created use the encountered_by relationship. Physical objects can be made_of a Material, and also have a current_owner of a Person or Group and a current_location of a Place.

Instances of LinguisticObject can also come into existence through the use of a created_by relationship to an activity, and are published by being used_for a publication activity. Both can be about an instance of any other class to describe their subjects using the about relationship, and VisualItem can have a represents property referring to any other class that the image directly depicts. LinguisticObject instances refer to the Language they are written in through the language property. These classes and relationships are summarized in **Figure 1**.

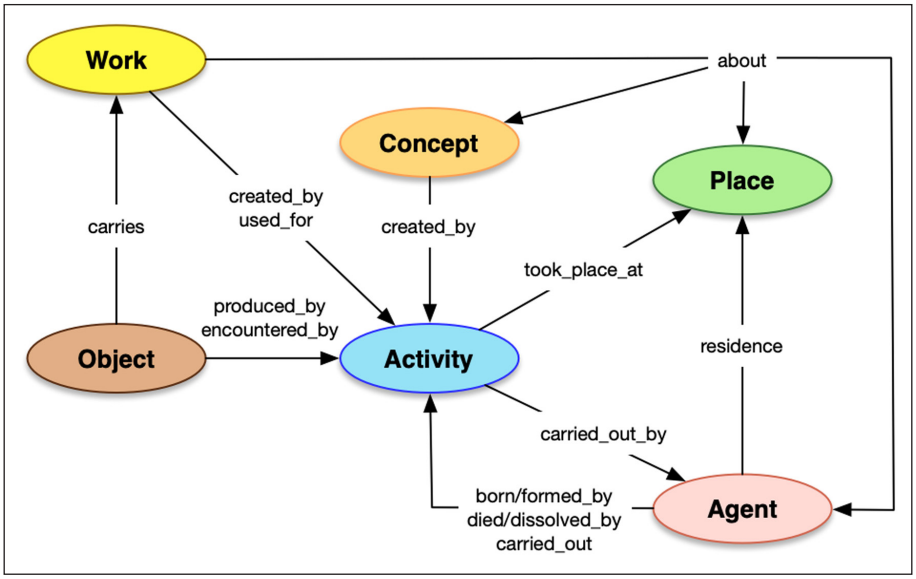

Figure 1: Summary of Linked Art Classes used in LUX.

Data patterns

The relationships described above are the pathways between the main classes; however, there are also several patterns, which are used on all of the classes within the records in order to express information directly about them (properties), rather than relationships between the records. These patterns include important information such as names, identifiers, descriptions, and external links outside of the data. Rather than go into detail about their internal structure, it suffices to enumerate them; to demonstrate that all of the core data modeling requirements can be met.

The patterns in use are:

1. Name or Title, through the identified_by property, which can have content, language and classifications using classified_as.
2. Identifiers, also through the identified_by property, which have content and classified_as, but not language. Identifiers are also used to describe the address or contact_point of an entity.
3. Descriptions or statements of any sort, through the referred_to_by property, which mirror the Name structure, but have longer textual content.
4. Dimensions, through the dimension property, with a numerical value, a MeasurementUnit instance, and a classification such as height or width.
5. References to external websites, images, APIs or other digital resources through the subject_of property, including a name, classification and the access_point URI to retrieve them.

6. External resources which also describe the same entity are linked through the equivalent property.
7. Finally, the date and time are embedded within the activities (either within records or separately) using a TimeSpan construction that records the earliest possible beginning, latest possible end, and a human readable form of the time.

We can now assess the model against the requirements, data, usability by software engineers, and the core use cases.

Evaluation

It is clear from the above description that the majority of the requirements are met by the Linked Art model. The aspects that bear further discussion are: how well it can deal with domains outside of its target of art museum collections, the real-world data that is managed in the different collection management systems, and its usability by software engineers without a background in art history or library science.

Cross domain modeling and data

Linked Art was easily able to capture the vast majority of the information managed by the two art museums' collection management systems. A small number of scenarios were not able to be captured as structured data, as they had been determined by the Linked Art community to be too complex compared to the value, such as knowing explicitly that the artist was one person or the other but not both and other detailed descriptions of uncertainty of attribution. Other edge cases included when the museums wanted a specific label or presentation in the application but that information was not carried by the metadata and could not be generalized across all of the datasets. Some of these were later addressed with additional statements associated with parts of the records, for example the role of a particular maker within the production of an object can be described in more detail in a statement to render, whereas the reference to the person and the type of activity can be expressed as structured data along with that descriptive content.

The information about the natural history collections of the Yale Peabody Museum (YPM, 2024) was perhaps the furthest from the domain of art and cultural heritage, being concerned with animal, plant and mineral specimens collected from around the globe. The Peabody is also the steward of other objects including Babylonian tablets and a collection on the history of science, which fall more cleanly into the art museum case. The primary difference for the specimens is that they do not carry an intellectual work, as they were not produced by a human with creative intent. Cleanly separating the physicality of the object from the intellectual work is a strength of the data model, allowing an accurate description of museum objects which do not carry designations

as creative works. The compromise, made for consistency, usability and standards-conformance of the data, was that even though the specimens are not made by humans, the system would still use HumanMadeObject for the class. Part of the rationale was that some specimens really are HumanMadeObjects, as they were actively repaired, reconstructed, or fabricated by conservators over time. The alternative would have been to have a separate PhysicalObject class, used only for specimens; the gain would have been purely semantic rather than practical.

The Type class was easily able to deal with taxonomic hierarchies, and the Group and Activity distinction meant that expeditions could be separated as to who went (the Group) from where and when they went (the Activity). The area which was the most difficult to model was stratigraphy, which simultaneously describes both geologic time and spatial location through the analysis of the layers of rock (or strata) and their positions relative to each other. This detail is the subject of ongoing work, with an initial model that introduces a new class to handle caves, arches, strata and other inseparable features of the natural world.

The modeling of archives using Linked Art was relatively easy after an initial decision was made as to the nature of an "Archive". Archival tradition conflates the description of the intellectual arrangement of the accessioned objects and their physical location into a single hierarchy for convenience of non-graph based descriptive formats, such as XML. It was necessary, therefore, to decide if the archive was a physical and/or spatial hierarchy of things at locations, or the product of human creativity in arranging the objects in that particular way. By separating the location from the arrangement, we could then determine that an archival collection was a Set (as a conceptual entity) of either other Sets (further divisions within the arrangement) or instances of HumanMadeObject or DigitalObject (the items themselves). Some of those items have known locations in space (Places) or are contained within other HumanMadeObjects, such as a letter being within a folder. This analysis was then discussed by the Linked Art community, as many art museums have related archives.

The vast majority of the records came from the Yale University Library collections (YUL, 2024), totaling some 13 million works, and comprising more than 30 million individual Linked Art records. The split between objects and works was important for this discussion as well, as some objects carry multiple works (such as a manuscript that contains several distinct texts by different authors), and many works are carried by multiple objects (such as multiple copies of the same textbook). Again, the vast majority of the information managed in MARC was easy to map into the Linked Art model. No serious attempt was made to reconcile works, as there is insufficient information in existing databases to do it in a way that would introduce more value than errors; however, if that were possible, the data model would support it.

The challenge regarding library data was not modeling but rather extracting the knowledge from the MARC format in a way that had never been done before, thereby exposing significant issues with the interpretation of the record as knowledge, rather than its rendering as HTML. An amusing error was when the barcode of an object was mistyped into a subfield of a person, resulting in it appearing to be the person's death date of some trillions of years in the future. While that was a straightforward fix, an ongoing issue is the use of incorrect subject heading subfields for places, resulting in states being part of cities, and countries on opposite sides of the world being embedded.

Overall, while there were some areas that required minimal additions and a small number of situations that were not feasible to model. However, Linked Art easily met all of the requirements to describe these four different domains with a coherent, connected and relatively complete data model.

Usability

The second test of the model was the extent to which it was easy for data and software engineers to produce and consume records that conform to the model. In LUX there are three distinct areas that require engineering effort: the creation of the linked art records from the collection records as handled by software engineers within the collecting units, in conjunction with subject matter experts such as curators or collection managers; the integration, reconciliation and enrichment of those records into a single coherent dataset by data engineers centrally; and, the use of the resulting data to produce an intuitive and highly functional discovery system for end users, by software engineers and user experience designers.

The first project participants to directly interact with the data model were the collection system managers and software engineers within the individual collecting units. This required, for each unit, around six to ten hours of discussion to generate a mapping from the system of record into the data model, followed by iterative work to implement that mapping in code. This included the four distinct domains of art, natural history, archival collections and bibliographic records, across five collection management systems. Each collecting unit designed and implemented different processes for the transformation based on the nature of their data and the skills of the available software engineers. These transformations were accomplished quickly, with the exception of the library data, due to the scale of that collection—some 13 million items. For the mapping and transformation from existing systems of record, this demonstrated that a cross-domain, graph data model was not an obstacle. This has subsequently led to other collections being mapped and transformed, including the School of Music's collection of musical instruments, the Campus Art Collection owned by the University directly, conservation related collections in the Institute for

the Preservation of Cultural Heritage, and the collections of the Paul Mellon Center in the UK, such that they can be added to LUX.

The records are then harvested, as discussed in the next section, and further manipulated to generate a single, coherent dataset. The assessment of usability for this purpose is heavily biased, as the central data engineering was performed by the author and a data engineer with several years of experience with the model. With that in mind, the features of the model lent themselves adroitly to the tasks to be performed, especially for the alignment of the internal records with external datasets that describe the same entities and the subsequent merging of up to more than a dozen records into a single, enriched description. With a strong conceptual model, and consistent data structures, comparison of records from very different domains and systems was significantly easier than dealing with the information in the native formats.

Thirdly, once there was a dataset built, software and UX engineers engaged with the data model to design and implement an understandable and useful discovery environment. This required understanding both the abstract model, in order to design the interactions between records for searching and linking, as well as the details of the JSON-LD (W3C, 2020) format, in order to extract, index, and render the different fields appropriately. The software engineers had had only limited exposure to the cultural heritage domain, and the main front-end engineer had a mere two years of post-degree experience when she started, yet they were able to quickly understand and productively use the data. This end-to-end workflow, encompassing a variety of experience, tools and platforms, has demonstrated that Linked Art is easy to implement, using a variety of environments, by relatively junior technical staff members.

Reconciliation and Enrichment

Beyond the internal sources of data, at the time of writing more than 20 external data sources were also used to both reconcile and enrich the knowledge about the person, organization, place and subject records. These sources were mapped and transformed by the data engineering team as part of the transformation pipeline, including cultural heritage authority data such as the Getty, Library of Congress, and OCLC vocabularies, large scale international datasets such as the National Libraries of France, Spain, Germany, Japan and Sweden to provide additional international perspectives, and more general datasets such as Wikidata. This further tested the data model and provided significant knowledge to ensure that the entities had sufficient description and relationships to play an active contribution to the overall knowledge graph.

A core benefit of having a target *lingua franca* for the knowledge mapping was that processing and merging the records, regardless of where they came from, was relatively

easy in comparison to having to deal with all of the systems' native representations. The availability of equivalent URIs for the same entity from both internal and external sources meant that these could be quickly matched up to create a cloud of information about the entity, with a high degree of confidence that the records really did describe the same thing. This was more successful for people, organizations and places than the subjects and other more theoretical concepts; however, the value added through the process greatly outweighed the errors that it also introduced. While it is out of scope to discuss the exact algorithms and challenges of the reconciliation and enrichment pipeline in this article, the project would not have been possible without this functionality built into the architecture, and the data model directly facilitated the processing.

Linked Data Technology

Data sitting on a computer has no value, without software to manage and make it available to the target audience; the author's long experience of engaging with linked data tools is that they are either unusably slow, unintuitive, incredibly complex, very expensive, and frequently all of the above. For Yale to implement Linked Art, a solution was needed that would make the knowledge easily accessible to all, without breaking either the bank or the minds of the engineering team.

While there is a wide variety of experience and systems in use across the collections at Yale, linked open data was merely an interesting concept before the project to build LUX. Efficient and usable technology was needed that did not replace the systems of record but enhanced access to them. Research as to which technology platforms would meet the needs of the project was required, and is valuable to consider in the broader context, as it represents a difficult decision that institutions must face in their own implementation paths.

Requirements

To ensure a fair comparison of systems and architectures, 30 requirements were established before research began, under the headings: data, search and query capabilities, performance and developer happiness. Over a period of four months, seven different systems were assessed, before selecting two, highly capable contenders.

The initial selection of candidate technologies was done based on several core "must have" requirements, the most discriminating of which was that the final system must be able to perform both graph searches to leverage the relationships in the data and textual keyword queries that treated the information as JSON records in a more traditional (and therefore familiar and intuitive) fashion. Secondly, it must be performant enough to generate facets across large result sets, including using values obtained through following relationships in the graph.

The requirement list also included the ease of ingesting, indexing, transforming, securing and retrieving records; support for tokenization and stemming of text in multiple languages, assigning and customizing relevancy scores, geospatial data, and the generation of facets; support for boolean, fielded, range (e.g., date or numeric) and hierarchical queries; support for geospatial and graph queries; the use of standards, APIs and code libraries, along with the ready availability of documentation and vendor support for the product.

In assessing the market, it became obvious that there were two categories of systems which would both require significant investment of time and resources, and that making the wrong decision would be catastrophic for the project unless it was caught early enough to swiftly change direction.

Multi-modal Solution vs System Integration

The first category was a single multi-modal system that could process both graph and document style queries simultaneously. It was immediately clear that there were no open-source options in this category, and only a very small number of vendors with realistic solutions. The alternative was to find the most capable set of products to manage records, document searching and graph searching, and then integrate those tools in a more distributed architecture, as in **Figure 2**. There is abundant availability of open-source or cloud-based tools for this second category; however, the time needed for the integration would reduce the amount of effort that could be applied to the discovery platform.

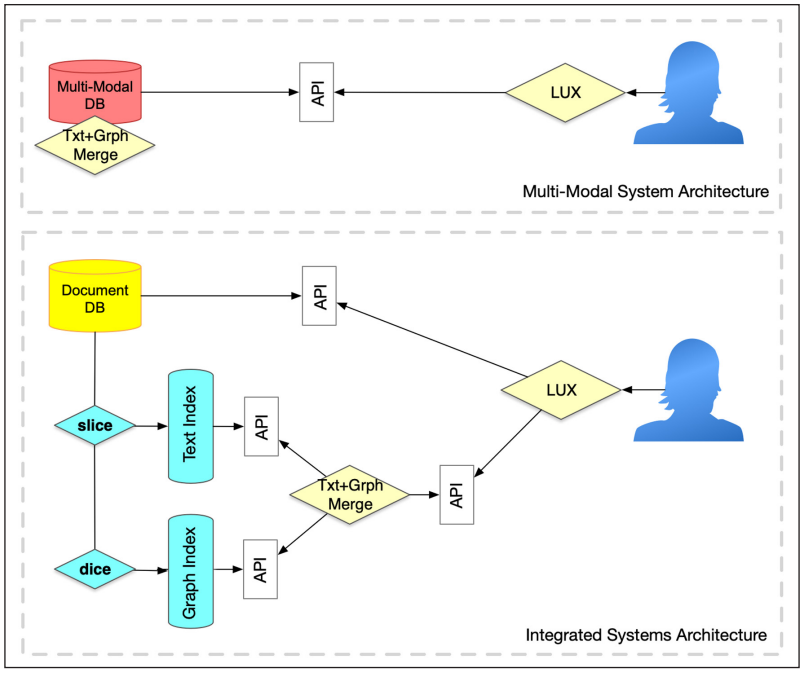

Figure 2: Multi-Modal versus Integrated Systems Architectures.

Through the assessment process, combinations of Amazon Web Services tools (Amazon, 2024), such as Neptune, Elastic, and DynamoDB, and more openly available products were considered. In both cases, the amount of effort that would have been required to even attempt to have an integrated query that was performant enough to meet the baseline requirements was so extensive that they were discarded from consideration. After also discarding products that claimed performant multi-modal capabilities but in practice could not deal with the data, the final choice came down to two products: a true multi-modal system, and an externally built integration of a highly performant graph engine with the Solr text indexing system.

After extensive testing with real data at scale, both products were highly capable and were approximately the same total cost of ownership, taking into consideration licensing and hardware requirements. The deciding factors were the longevity of the companies involved (one was a startup, the other well-established), the availability of support and documentation, and testimonials from existing users of the platforms. A further consideration was that the selected system could also function as just a document database, thus giving the project a fallback path if the graph functionality turned out to be too complex or too slow in the available time for implementation. These secondary factors tipped otherwise very balanced scales in favor of MarkLogic (Progress, 2024).

Solution Architecture

The overall architecture for acquiring, processing and loading data was informed by the need for a single system to manage the information, that would need to be loaded with the final records, rather than being able to construct them on the fly from a triplestore.

To remain synchronized with changes in the distributed systems of record, each of the participating units implements the IIIF Change Discovery API (IIIF, 2021), which is in turn built upon the W3C's Activity Streams 2.0 specification (W3C, 2017). This pattern was selected as image content was already available via the more well established IIIF specifications, and thus adopting another API from the same suite was both technically and politically easy. The Change Discovery API describes the update events that have occurred to the records in chronological order, such that a harvester can walk from the most recent change backwards in time until the point at which it last processed the stream, retrieving new and updated records over the web and deleting records that should no longer exist. The units also produce large dump files with all of the records to avoid millions of HTTP requests at the initial load time. These are not referenced from the activity stream, but the value of this functionality will be provided to the IIIF consortium as feedback on the specification to be considered for inclusion in a future version.

The external data sources do not provide the Change Discovery API for their content, with the exception of the Getty Vocabularies; instead they are downloaded as dump files, if available, and loaded wholesale into record caching infrastructure for ease of access, or downloaded when needed during the subsequent processing. Dump files are greatly preferable, to avoid millions of requests over the network to retrieve individual records from around the world, even if only a small fraction of the dataset is used. These records are then mapped and transformed into Linked Art, as described in the previous section.

After all the records are synchronized to a single virtual machine, they are processed in parallel; the algorithm uses only the information from a single record at once, and therefore the order in which the records are encountered does not matter. While this introduces challenges, it is important for the scalability of the reconciliation and enrichment pipeline, as with 41 million records in the resulting system, it would take more than a month of processing using a single process at a time.

The processing pipeline uses only open-source products, including PostgreSQL for caching the harvested, intermediate and final representations of the records and Redis for extremely fast access to a key/value store, necessary for managing the concordances between the 100 million+ URIs involved. The processing is implemented in Python, as a familiar language for data engineers with an abundance of existing libraries and tools. It has three core phases:

1. The records are reconciled by initially considering URIs that are claimed as identifying the same entity, for example an equivalence between the same concept in two different authority systems. If there aren't any such URIs provided or discoverable, it attempts to exactly match the name of the record against the closest domain and entity type authorities in turn. For example, a person from a library record would first be matched against the Library of Congress Name Authority File, and then if there was not a match, proceeding to other sources. The matched records are then processed in the same way to collect more and more equivalents through the network of linked open data.
2. Once reconciled and the identifier set has been established, all of the records have their identifiers and outbound links to other records updated to the internal URI used by LUX.
3. Finally, all of the records with the same internal URI are merged, some final tidying is performed and the dataset is exported to load into MarkLogic where it is ready for use.

System Optimization

In early phases of the implementation, the processing pipeline materialized all the RDF triples in the JSON-LD records into Marklogic to ensure all of the information was available for graph queries. It quickly became apparent that, at the scale of data in LUX, we needed to optimize the indexing for the interactions we knew were necessary, rather than the more traditional approach of indexing everything in the triplestore.

The first change was to create artificial triples for indexing purposes in the data processing pipeline. For example, the relationship between an object and the person that created it could run through three triples (object created_by/part/carried_out_by person) rather than just one, which meant that the system needed to perform unnecessary joins across large intermediate result sets. To alleviate this, we added additional "shortcut" triples to prevent the need for the joins, in this case lux:agentOfProduction. This also allowed us to deal gracefully with slightly different data structures, including whether the person carried out the production as a whole, or only part of it.

Once we had established the set of queries, we also built a custom generator for the triples rather than relying on the JSON-LD expansion algorithm. This proved to be many times faster, and let us materialize only the triples that we needed to power the searches, reducing the dataset's size on disk by approximately half. While the unnecessary triples did not affect performance at run time, they require significant time to generate and load, and then disk space to store. External uses of the same dataset would not be affected, as every triple is present in the publicly available JSON-LD records, and thus we have both semantic precision when needed, and performance for our applications.

The queries were initially developed using SPARQL (W3C, 2013); however, with performance testing, we found that the MarkLogic query language called CTS was many times faster to execute. By aligning the document identifier used for record-based searching with the URI of the entity described in the document used in the graph queries, we could then join graph and document queries together very efficiently. This allows us to use the right tool for the job: document search for relevance and text queries, and graph search for connections between entities in the dataset. MarkLogic also has another API called Optic; however, at the time of implementation it was not as performant for our use cases as CTS.

Evaluation

By thinking outside of the traditional linked data box of SPARQL and triplestores, and using the right tool for the job in terms of documents and graphs, we were able to create a highly scalable and performant system. Furthermore, by adopting a reasonably-priced

commercial product, we could focus limited developer resources on understanding and solving our specific problems, rather than building an integration architecture which would need constant maintenance. We feel that linked data technology is ready for the cultural heritage sector today and is well-suited to dealing with the Linked Art data model; however, care must be taken to make the choice that is appropriate for the project or product.

Cross-Collection Discovery

The best data model, cleanest and most precise data, and most sophisticated technology platform would all be worthless without an intuitive and usable web application to allow users to interact with the knowledge maintained by the system. Building a discovery platform on top of a graph rather than a traditional relational or document-based system enabled us to rethink received wisdom around implementation and user experience patterns to produce something innovative. This section will not deal with the LUX user experience or interface design directly, but instead focus on the overall lessons learned from doing that work.

User Experience Paradigm

So far, the discussion has been couched in technical terms of searching and queries; however, as Roy Tennant has famously said, 'only librarians like to search, everyone else likes to find' (Tennant 2001). The user experience paradigm can easily shift to finding or discovery rather than repeatedly searching, given the use of linked data. This can be managed across collections through the adoption of a single, appropriate conceptual model, used to consistently expose the relationships between entities, rather than merely presenting the name of the entity as an unactionable string in the user interface, or treating it as a search term and taking the user to yet another list of results.

By presenting a record about the entity, for example the Person record for Roy Tennant as quoted above and captured in **Figure 3**, the user can explore the knowledge graph in a mediated and controlled way. The user interface presents the information about the person, along with the relationships with the items in the collections (for example the works that the person has written) and other entities (such as their nationality or place of birth and their occupations). Further, the graph can be leveraged to discover hidden relationships by determining frequent co-authors, the subjects that the author commonly writes about, or places that they write about or publish at. This allows the user to easily focus in on what they are looking for, and at the same time gain more contextual information about the people, places, events and concepts in the cultural heritage landscape.

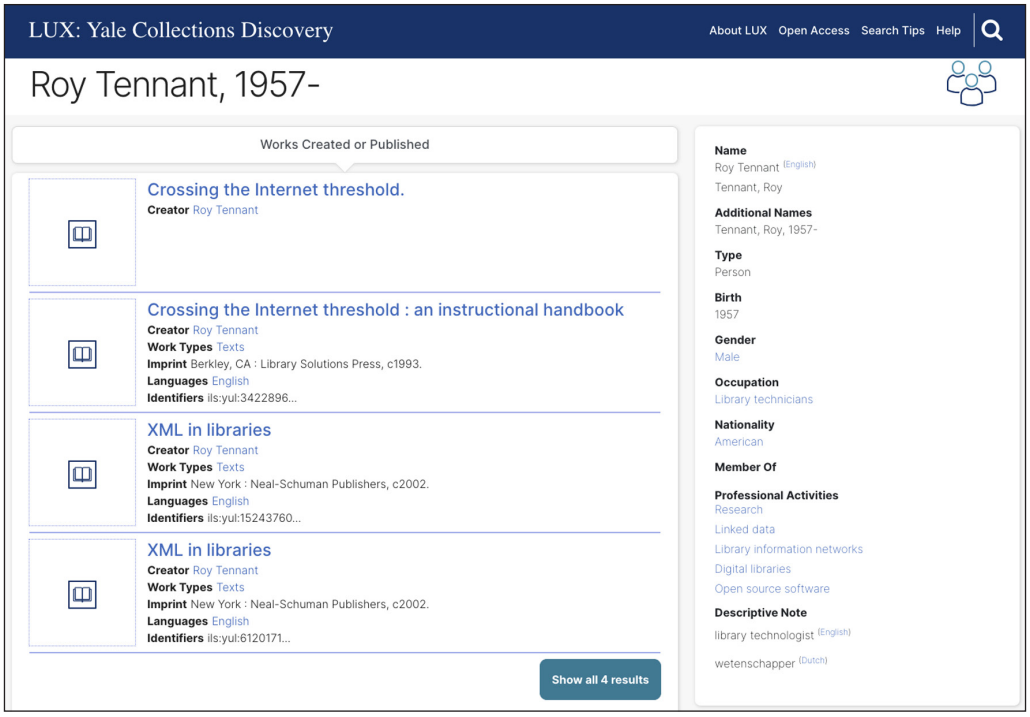

Figure 3: Screenshot of the LUX view for the record about Roy Tennant.

Given the new functionality available, the decision was made to try to keep the user interface as clean and familiar as possible for people accustomed to using museum, archival or library search systems. This also helps to bridge the conceptual gaps between domains—searching for a natural history specimen is typically not performed in the same way as searching for a book, a painting or within an archival hierarchy, yet all of these are included within the cross-collection dataset. This meant that the front-end application required extensive design work, continuous implementation and refinement effort, and has been tested with a variety of users.

Building an Application on Linked Data

There were two primary lessons that were learned through the development of a front-end application interacting directly with linked data. These are the necessity of the record as a construct, and that web caching infrastructure can support the burden of interlinked records without duplicating information across those records, leveraging the architecture of the web.

One of the tendencies of Linked Data aficionados is to emphasize that the record is no longer necessary, and there is only a single graph of all knowledge. However, nothing could be further from the usability-required truth that everyone, from content

managers to data and software engineers to end users, thinks in terms of records or at the very least in terms of distinct entities which might as well be represented as records. In order to build an application within a reasonable amount of time, a reasonable budget and without sending software engineers back to university for graduate degrees in graph query optimization, maintaining the record construct was essential. Furthermore, the computational costs of repeatedly constructing the same sub-graphs needed to render the information to the end user, or to compute the counts for facets on result sets, is overwhelming. One solution is to introduce a cache for those computed sub-graphs, materializing them into a discrete, serialized chunk of JSON-LD, which is functionally equivalent to a pre-constructed record.

The architectural design of the interactions between the web application and the knowledge graph was also illuminating in terms of what was intuitively felt to be feasible by the software engineers, and what turned out in practice to work well. There was a great desire to deliver all and only the information necessary from the dataset to the front end via a custom-built API, with the rationale being that it would be too slow to deliver all of the information about a particular entity, and too hard for the front-end application to extract the information from the deeply-nested record structure. However, the prior decision to use React (a development library that creates a single page application which loads once and then retrieves information via javascript calls to render) combined with the use of multiple layers of web caches meant that the performance was always well within pre-established acceptable bounds of rendering the layout in under a second, and the details in under three seconds. The web caches are easy to maintain and optimize, as the searches are structured in a systematic way, and the interactions with the data are via the static JSON-LD representations, rather than dynamically constructed responses. As React and the browser maintain internal caches as well, the application can extract the aspects of the record that are needed without requesting another representation. For example, the same cached response will be used to generate the name of the link in one page, the entry in a results list and the full record view if the user clicks through to it. As the graph is richly connected, the same records are likely to be used many times in a single user session. As such, the user's experience is improved by this REST-oriented paradigm, and not degraded in any way.

The implementation of the front-end application was made easier by intentional designs in the data structures of the knowledge graph. In particular, the consistent use of patterns within the data, as described in section 2, allowed javascript components to be built that extracted, interpreted and rendered the pattern. These are then composed into higher level components and reused to produce the different page layouts. This consistency and rigor in the data has downstream benefits in terms of development

time, maintenance, the ability to change the application without completely refactoring it, and the relative independence of the front end from the backend dataset or query engine. This lowers the total cost of ownership over time dramatically, and at the same time allows easy reuse of either the data or the code. For example, we have used the same front-end application over top of a completely static site with no query functionality at all, and while it (of course) relies on starting with a link and following the relationships in the graph, the code did not need to change to present a useful interface. We have also used the same front-end application code with some extremely minor cosmetic changes to create a user interface for the discovery of research datasets, managed using the same data structures within a MarkLogic back-end. Thus, the same system gracefully degrades in the absence of back-end functionality, and can be reused without refactoring for new domains, providing strong evidence of a highly sustainable codebase.

Hypermedia API

The "follow your nose" approach of knowledge discovery works well if everything is linked both to and from everything else. However, in a highly connected knowledge graph like LUX, having every relationship maintained bidirectionally would be an enormous overhead. For example, the concept of "books" would need to have an explicit reference to each of the millions of books in the library, and adding any new record would require updating every record that it references with the corresponding back link. Instead, LUX follows the Linked Art API's recommendation of linking from the child to the parent, or from the many to the few. This recommendation means that there will not be records with thousands or even millions of links in the "book" or "specimen" cases, but instead each of the millions of books and specimens each links once to the respective concept. For example, a photo album with 100 pages, each of which is described separately, would be the target of the links from those pages, rather than linking to them directly. However, when you are on the photo album page, the artist page or the state page, the user needs to be able to discover those referring records without waiting beyond the acceptable page load times.

The solution adopted is to search for the records that refer to the current one. Given an index of the references, as maintained by the triplestore aspect of the back-end product, this is extremely fast and adds almost no overhead to the system for direct relationships. The page then renders the first few results of the search and links to the full search results page for the user to transition between the record and the set of records that refer to it. Initially, this was accomplished by writing the queries into the page; however, this proved to be brittle in the face of a changing back-end query syntax,

changing data model, or other iterative work. Much of the impact of the separation of the application and the data, as described in the previous section, was undone by tightly coupling them again via the queries.

To tease the systems apart again while maintaining the functionality, we use the Hypertext Application Language (HAL) (Kelly, 2023) to provide named search links that the front end can follow and receive a standardized response, rather than having to construct that link itself. The links are kept separate from the semantic knowledge using the standard-defined_links property. The links are added at request time by the middle tier system, which tests whether there will be at least one record that matches, and thus if the front end sees the link in the response, it knows that retrieving the representation will provide additional information. As this is part of the response, the query links get cached and thus the one-time cost of performing these tests is quickly mitigated.

This indirection through the named link provides several benefits. Firstly, it separates the back-end query technology from the front-end, thereby decoupling the two systems. The same code would work in exactly the same way with an entirely different query language, because the front-end doesn't need to construct or manipulate it, only follow the provided link. It also separates the data model specifics from the front-end, allowing the two to iterate more independently. We could change the semantic definition of a particular query to include further cases, and the front-end code would never notice the change. Finally, it creates a better division of skills needed between front- and back-end engineers. The data and software engineers that deal with provisioning the search indexes can configure and name the queries more easily than a front-end engineer can create them, especially in the face of changing back-end capabilities and data patterns.

The separation also works because the response follows the paged collection pattern from the W3C Activity Streams standard. The specification defines collections of items, coalesced into pages, which are doubly linked via next and previous links, allowing a consuming application to walk forwards or backwards through conveniently sized subsets of the overall collection. It was easy to implement for search results, facet results and the more complex related list responses that also have specific labels for each result. This gives a standard structure for the producing and consuming implementations to work with, rather than needing to reimplement technology-specific formats for different uses.

Evaluation

The patterns and solutions established have been tested in a practical sense by the ongoing efforts to improve LUX, and to reapply the paradigm in other domains within Yale. With a different dataset using Linked Art as the baseline, it was less than a person-day of engineering effort to stand up a new system to manage and render that data, including with several new indexes and features. This was accomplished by reusing the existing components with some newly configured labels and adding a handful of new HAL links, by a relatively junior software engineer.

The query structure has changed after the HAL links were added to move the scope of the search (objects versus people, for example) into the URI from a query parameter. The front-end application did not need to update the record pages at all, despite using a dozen or more queries, as it simply followed the new HAL links.

The performance of the system is improved by the aggressive use of web caching, ensuring that the request only gets all the way to the database when it is really something that has not been done in the last week. Performance testing of the system without the cache allows for 150 concurrent queries, but in regular operation we have never come close to this number as the vast majority are handled by the different caching layers. This approach relies on the consistent identification of records and using a single static representation of them, rather than dynamic queries constructing application specific structures. While there is currently only one front-end application, in the not-too-distant future there will be more that use exactly the same back-end infrastructure, and that will also prove (or disprove) the value of the consistency.

Conclusions

We have endeavored to demonstrate the feasibility and value of an approach that rethinks the notion of cross-collection discovery in the context of a knowledge graph, built using standards such as Linked Art, IIIF, Activity Streams, the Hypertext Application Language and more. In particular, the foundational ontology design of Linked Art allows the multiple domains within natural and cultural heritage to happily co-exist and be richly interrelated without significant work to re-describe the source materials.

The usability of Linked Art as both a consistent and coherent data model and an easy to understand and implement API was critical to the success of LUX. Consistency in the data enabled consistency and simplicity in the implementations that produce, transform and interact with the resulting knowledge graph. The API structure based on records was also essential, as it eased the burden of the implementation and allowed functionality such as faceting to be computationally tractable using standard

techniques and tools. The use of a multi-modal database that supports both graph- and record-based search simultaneously allowed the right tool to be used for different aspects of the desired functionality. Finally, the use of web caching infrastructure and the indirection provided by HAL links for named searches helped to ensure that the front-end was performant, easy to build and adapt, and cheap to maintain and reuse.

Yale hopes that by adopting open standards and publishing the LUX codebase as open-source software, other organizations will be able to learn from, use and contribute to a practical and sustainable cultural heritage knowledge graph that could span across institutions and domains. If other organizations were to also publish their data according to Linked Art, and made harvestable via IIIF's Change Discovery API, then organizations that wish to integrate and reuse that knowledge could do so by following the same approach that Yale has with LUX. This would generate a valuable network of knowledge without requiring a single, centralized and ultimately unsustainable database; the incentive to participate in the network is to have a local cross-collection discovery platform. Keeping this local discovery platform updated with accurate information then increases the value of the network as an ecosystem, and allows other organizations to select which systems to use for enriching their own records rather than being forced to use self-proclaimed authority systems. Given the experience with LUX, this future which is both close and bright promises to provide unprecedented access to the collections of the world's museums, libraries, and archives.

Acknowledgements

The work described was partially funded by an Andrew W. Mellon Foundation grant to Yale University.

Competing Interests

The author has no competing interests to declare.

References

Amazon 2024 *Cloud Computing Services*. https://aws.amazon.com/ [Last Accessed 26 March 2024].

American Art Collaborative (AAC) 2017 *The American Art Collaborative*. https://americanart.si.edu/about/american-art-collaborative [Last Accessed 26 March 2024].

Apache Foundation 2024 *Welcome to Apache Solr* https://solr.apache.org/ [Last Accessed 26 March 2024].

Bellinger, M 2011 'Model for Campus-Wide Content Strategy', paper presented at *Coalition for Networked Information Fall Forum 2009*, *Washington DC, USA*. https://www.cni.org/wp-content/uploads/2011/08/cni_yale_bellinger.pdf [Last Accessed 26 March 2024].

CIDOC 2024 *The CIDOC Conceptual Reference Model*. https://cidoc-crm.org/ [Last Accessed 26 March 2024].

Farallon Geographics 2024 *Arches Project*. https://www.archesproject.org/ [Last Accessed 26 March 2024].

International Federation of Library Associations and Institutions (IFLA) 2018 *IFLA Library Reference Model: A Conceptual Model for Bibliographic Information* https://repository.ifla.org/handle/123456789/40 [Last Accessed 26 March 2024].

International Image Interoperability Framework Consortium (IIIF) 2021 *IIIF Change Discovery API 1.0* https://iiif.io/api/discovery/1.0/ [Last Accessed 26 March 2024].

Kelly, M 2023 *JSON Hypertext Application Language*. https://datatracker.ietf.org/doc/html/draft-kelly-json-hal-11 [Last Accessed 26 March 2024].

Library of Congress (LOC) 2024 *Bibliographic Framework Initiative*. https://www.loc.gov/bibframe/ [Last Accessed 26 March 2024].

LUX 2024 *The Artist's Garden in Giverny by Claude Monet*. https://lux.collections.yale.edu/view/object/e6fa0e33-1575-42f5-b00f-d1494bc0e9d9 [Last Accessed 26 March 2024].

Open Archives Initiative (OAI) 2015 *The Open Archives Initiative Protocol for Metadata Harvesting*. https://www.openarchives.org/OAI/openarchivesprotocol.html [Last Accessed 26 March 2024].

Progress 2024 *Database Platform to Simplify Complex Data: Marklogic*. https://www.progress.com/marklogic [Last Accessed 26 March 2024].

Raemy, J and **Sanderson**, **R** 2024 Analysis of the Usability of Automatically Enriched Cultural Heritage Data. In: Moral-Andrés, F. Merino-Gómez, E. Reviriego, R. (eds.) *Decoding Cultural Heritage: A Critical Dissection and Taxonomy of Human Creativity through Digital Tools*, Springer, forthcoming. https://arxiv.org/abs/2309.16635 [Last Accessed 26 March 2024].

Sanderson, R 2018 'Shout It Out: LOUD', keynote presented at *Europeana Tech*. Rotterdam, 15 May 2018. https://www.youtube.com/watch?v=r4afi8mGVAY [Last Accessed 26 March 2024].

Sanderson, R, Anderson, E, Beaudet, D, Bruseker, G, Dang, T, Isaac, A, McQuaid, J, Lill, J, Midavaine, B, Norling, S, Page, K, Palmer, R, Raemy, J, Sissman, D. 2024 *Linked Art*. https://linked.art/ [Last Accessed 26 March 2024].

Tennant, R 2001 *Cross-Database Search: One-Stop Shopping*. https://www.libraryjournal.com/story/cross-database-search-one-stop-shopping [Last Accessed 26 March 2024].

World Wide Web Consortium (W3C) 2008 *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. https://www.w3.org/TR/xml/ [Last Accessed 26 March 2024].

World Wide Web Consortium (W3C) 2013 *SPARQL 1.1 Query Language*. https://www.w3.org/TR/sparql11-query/ [Last Accessed 26 March 2024].

World Wide Web Consortium (W3C) 2017 *Activity Streams 2.0*. https://www.w3.org/TR/activitystreams-core/ [Last Accessed 26 March 2024].

World Wide Web Consortium (W3C) 2020 *JSON-LD 1.1 A JSON-based Serialization for Linked Data*. https://www.w3.org/TR/json-ld11/ [Last Accessed 26 March 2024].

Yale Center for British Art (YCBA) 2024 *Collections Overview*. https://britishart.yale.edu/collections-overview [Last Accessed 26 March 2024].

Yale Peabody Museum (YPM) 2024 *The Collections*. https://peabody.yale.edu/explore/collections [Last Accessed 26 March 2024].

Yale University Library (YUL) 2024 *Yale Library Homepage*. https://library.yale.edu/ [Last Accessed 26 March 2024].