Leveraging Big Data and Computing Technologies for Improved Digital Accessibility in Cultural Heritage

Avgoustinos Avgousti [1,*,†] Georgios Papaioannou[2,†] and Athanasios Koutoupas[1,†]

[1] *The Cyprus Institute (CyI), Konstantinou Kavafi 20, 2121 Nicosia, Cyprus*
[2] *Ionian University, Dept. of Information Science, Corfu 49100, Greece*

Abstract
This article shows an easy way to integrate Semantic Markup Annotations using the Django Python Framework, Schema.org ontology, or vocabularies, and JSON-LD to increase the online findability of cultural heritage content. By employing those technologies, developers and site builders can improve the online discoverability of cultural heritage digital resources, making them more accessible to scholars and the public. This approach shows structure data's potential to increase cultural heritage (CH) findability and interaction on a global scale.

Keywords
Big Data, Applied Computing, Structured Data, Findability, Django Python, Semantic Markup Annotations, Cultural Heritage

1. Introduction

The internet allows humans to share information, including cultural heritage resources. Online resources help preserve cultural heritage while making it more accessible to a global audience. However, simply having something online does not guarantee that it will be found among the vast ocean of information available on the internet today [1]. Traditional methods of categorization and description are often unable to keep up with the dynamic and interconnected nature of the web. This article demonstrates how Semantic Marup Annotation can improve online cultural heritage resource's findability.

We present a structured approach to making web content machine-readable and more discoverable by search engines and metadata aggregators such as the Europeana digital library by integrating Schema.org and JSON-LD within the Django Python Framework. Our discussion includes practical examples of how cultural institutions, such as museums and archives, can significantly use these technologies to improve their resources' online findability. We look at the specifics of implementing structured data to better connect cultural heritage with the public and researchers, fostering a more informed and engaged global community. This article not only discusses the technical aspects but also considers the broader implications of making cultural heritage more accessible and discoverable online.

MBS2024: 3rd International Conference On Museum Big Data, November 18-19, 2024, Athens, Greece
* Corresponding author.
† These authors contributed equally.
a.avgousti@cyi.ac.cy (A. Avgousti); gpapaioa@ionio.gr (G. Papaioannou); a.koutoupas@cyi.ac.cy (A. Koutoupas)
00000000-0002-9116-8217 (A. Avgousti); 0000-0002-9270-0463 (G. Papaioannou); 0000-0003-4913-2752 (A. Koutoupas)

2. The Django Python Framework

The Django Framework has been used in many cultural heritage projects and for the development of many web applications [2]. Django models are Python classes that represent relational database tables. Django models communicate with the database via an Object-Relational Mapping (ORM), rather than direct SQL queries.

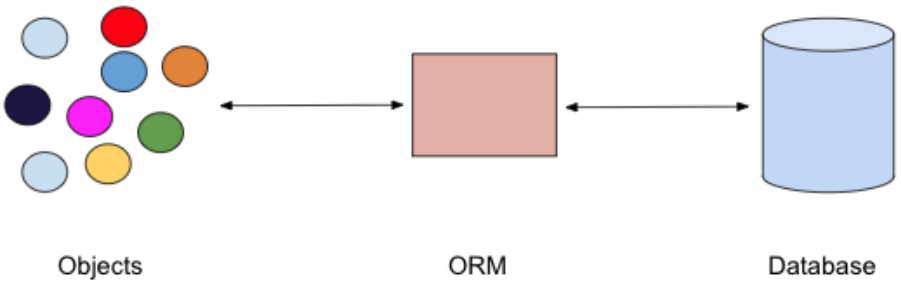

Figure 1: Django Object-Relational Mapping. Graphic by the authors.

Consider the following Django models: "Artist" and "Painting," each of which has a relationship (ForeignKey) with the other. This indicates that a painting is associated with an artist. The following code creates the two tables Artis and Painting in the database.

```
from django.db import models

class Artist(models.Model):
   name = models.CharField(max_length=100)
   nationality = models.CharField(max_length=100)
   birth_date = models.DateField()
   death_date = models.DateField(blank=True, null=True)
   same_as= models.URLField(blank=True)

   def __str__(self):
       return self.name

class Painting(models.Model):
   title = models.CharField(max_length=100)
   artist = models.ForeignKey(Artist, on_delete=models.CASCADE)
   art_form = models.CharField(max_length=100)
   art_medium = models.CharField(max_length=100)
   year_created = models.IntegerField()
   image = models.ImageField(upload_to='paintings/')
   description = models.TextField()
   same_as= models.URLField(blank=True)

   def __str__(self):
       return self. title
```

This will create our Django application; once registered in the admin.py file, we will be able to access it and start publishing data. For the demonstration, we will use public access data from the Metropolitan Museum of New York's online collections (https://www.metmuseum.org/art/collection). The admin.py file would look like this.

```python
from django.contrib import admin
from .models import Artist, Painting

class ArtistAdmin(admin.ModelAdmin):
   list_display = ('name', 'nationality', 'birth_date', 'death_date')
   search_fields = ['name', 'nationality']
   list_filter = ['nationality', 'birth_date']

class PaintingAdmin(admin.ModelAdmin):
   list_display = ('title', 'artist', 'art_medium', 'year_created')
   search_fields = ['title', 'artist__name']
   list_filter = ['art_medium', 'year_created']

admin.site.register(Artist, ArtistAdmin)
admin.site.register(Painting, PaintingAdmin)
```

This will allow us to access the models through the Django administration interface. And we can add our content, which in this case is our painting collection. The images below depict the two models and the connection between them.

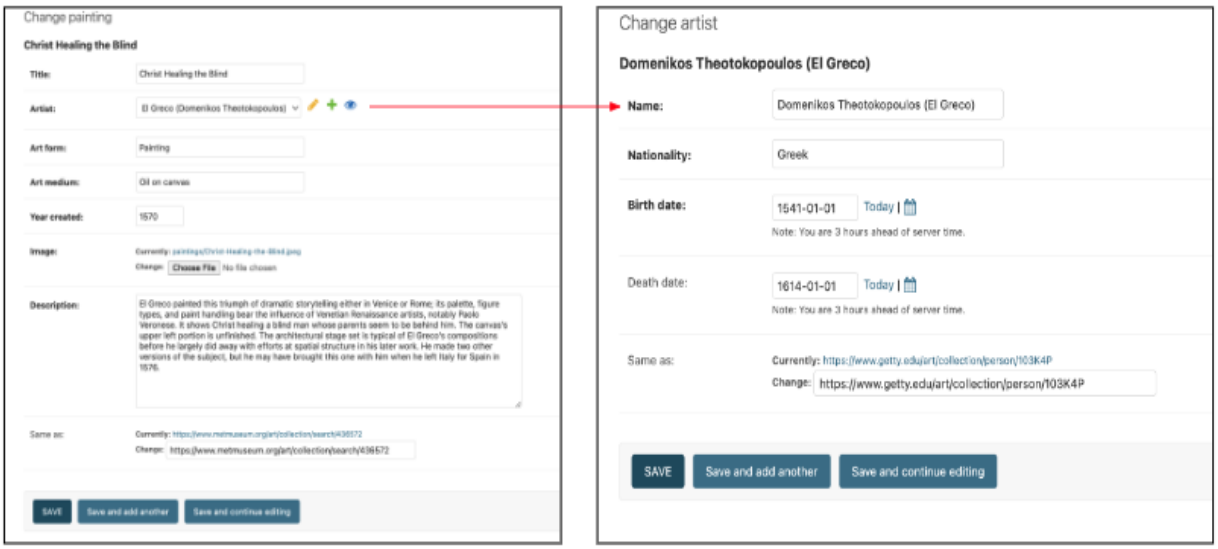

Figure 2: Django admin interface with the relation between the two models. Screenshots by the authors.

To effectively display the data on the front end with Django templates, we must first create a view that retrieves the required data and routes the request to the appropriate HTML template. In Django, this entails defining a view function in the application's views.py file, which serves as the logic layer that controls what users see and interact with.

In our example, the view will retrieve data from the database for a specific painting and artist before rendering an HTML template based on that data. This process is made easier by Django's URL dispatcher, which maps URLs to their corresponding views. Here's how we can set the URL and view.

In our Django application's urls.py file, we must specify a URL pattern that identifies a specific painting by a unique identifier (such as a primary key). This pattern will correspond to our view.

```
    from django.urls import path
    from . import views

    urlpatterns = [
     path('paintings/<int:pk>/', views.painting_and_artist_detail, name='painting-
    and-artist-detail'),
```

The view function, painting_and_artist_detail, will retrieve the painting object from the database. If the painting with the specified primary key does not exist, the page will return a 404 error. Otherwise, it sends the painting data to the template.

```
from django.shortcuts import render, get_object_or_404
from .models import Painting

def painting_and_artist_detail(request, pk):
    painting = get_object_or_404(Painting, pk=pk)
    return render(request, 'painting_and_artist_detail.html', {'painting':
painting})
```

This approach ensures that the front-end template receives all of the required data about the painting and its artist, allowing for a detailed display of cultural heritage items in a user-friendly format. By organizing URL routing and view logic in this way, we create a clear and maintainable structure for web application development.

Following the setup of the view and URL routing, we will now discuss how to pass the fetched data to the front end for display. Using a Django template, a powerful tool for dynamically generating HTML content based on database data.

To successfully create the Django Template and display the painting and artist details, we create an HTML template using Django's templating language to insert data into the HTML structure. This template will be named painting_and_artist_detail.html and will be saved in the Django app's templates directory. Here's how to structure this template to include the necessary details:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>{{ painting.title }}</title>
</head>
<body>
   <h1>{{ painting.title }}</h1>
   <h2>Painting Details:</h2>
   <ul>
       <li>Artist: {{ painting.artist.name }}</li>
       <li>Art Form: {{ painting.art_form }}</li>
       <li>Art Medium: {{ painting.art_medium }}</li>
       <li>Year Created: {{ painting.year_created }}</li>
       <li>Same As:{{ painting.same_as }}</li>
       <br>
       <img src="{{ painting.image.url }}" alt="{{ painting.title }}" style="width:
```

```
400px; height: auto;">
        <br>
        <br>
        <li>Description: {{ painting.description }}</li>
    </ul>
    <h2>Artist Details:</h2>
    <ul>
        <li>Name: {{ painting.artist.name }}</li>
        <li>Nationality: {{ painting.artist.nationality }}</li>

        <li>Birth Date: {{ painting.artist.birth_date }}</li>
        {% if painting.artist.death_date %}
            <li>Death Date: {{ painting.artist.death_date }}</li>
        {% endif %}
        <a href="{{ painting.artist.same_as }}">
            Same As

        </a>
    </ul>
</body>
</html>
```

To display the information stored in the database on a web page, we use a Django template, which is an HTML file enhanced with Django's templating language. This enables dynamic data rendering based on information retrieved from the database using views. The following example shows how to use Django templates to display information about cultural heritage items, such as paintings and their associated artists, that have been entered into a database.

Fig. 3 offers an example of a Django Template for Displaying Database Information:

Figure 3: The front-end template showing the painting and the artist. Screenshot by the authors.

By structuring your Django template in this way, we ensure that the data entered into our database is displayed in a user-friendly and informative manner on your website, thereby increasing the accessibility and engagement of your digital collections.

However, this does not render our content machine-readable.

3. Semantic Markup with Schema.org and Extensions

Schema.org vocabulary or ontology is developed following the Semantic Web principles [4], [5]. Provides a structured vocabulary of data types and properties that can describe various types of web content in a machine-readable format. Such, content includes cultural heritage objects, such as monuments, historic sites, and other related entities. It was launched in 2011 by major search engines (e.g., Google, Yandex, Bing, etc) and is a W3C initiative [6].

Schema.org enables developers to enrich web pages with metadata, making the information available for human and machine consumption instantaneous. Many cultural heritage organizations, including museums, use Schema.org to enable search engines and other machines to better understand the information within their online content, thereby enhancing visibility online. By Incorporating structured data, CH organizations can offer more detailed and accurate information about their digital collections, exhibitions, and events, thereby helping users find the information they seek more effortlessly [8].

The following discussion highlights the Schema.org extensions in the cultural heritage domain.

3.1. BiblioGraph.net

An extension to Schema.org that specifically focuses on providing additional markup tags for describing cultural heritage resources and collections, such as those found in museums, libraries, and other institutions. It provides a set of terms and entities that can be used to describe the specific characteristics and metadata of cultural heritage resources and collections, such as the creator, date of creation, physical format, and more. By using BiblioGraph.net, cultural heritage institutions can provide more detailed and accurate information about their resources and collections, which can help make them more easily discoverable and accessible to the public, and also provide more context and information to researchers and other interested parties (https://bibliograph.github.io/BibloGraph-Frozen/bibliograph.net/).

3.2. Schema Architypes Extensions

The Architypes extension is another example of an extension to Schema.org that focuses on providing additional markup tags for describing cultural heritage resources, specifically archival collections [7]. Architypes is an extension of Schema.org that focuses on providing additional terms and entities for describing the specific characteristics of archival collections, such as the creator, date of creation, physical format, and more. Architypes provides a set of terms and entities that can be used to describe the specific characteristics and metadata of archival collections, such as the creator, date of creation, physical format, and more. By using Architypes, cultural heritage institutions can provide more detailed and accurate information about their archival collections, which can help make them more easily discoverable and accessible to the public and also provide more context and information to researchers and other interested parties [1].

3.3. Schema.org Bibliographic Extension

The Schema.org Bibliographic Extension is another extension of Schema.org that targets the improvement and representation of bibliographic information markup sharing. This extension provides additional terms and entities that can be used to describe bibliographic resources such as books, journals, and other types of publications. This extension allows libraries, universities, and other institutions to share their bibliographic data in a standardized way, making it easier for others to discover and access the resources they hold.

The University of Illinois experimented with a subset of its bibliographic records describing print resources and related holding data, to explore the options and best practices identified to date for expressing library holding data using Schema.org ontology or vocabulary.

Of course, Schema.org has been used in other domains such as life science, an example is BioSchemas which aims to improve content findability in a specific domain, in this case, the life science domain. BioSchemas provides a set of terms and entities that can be used to describe resources in the life science domain, such as experiments, genes, and proteins, making it easier for others to discover and access resources in this field [3].

In general, these extensions are a great way to bring more structure and organization to the information shared on the web, making it easier for machines to understand and find the information they are looking for.

3.4. JSON-LD

JavaScript Object Notation for Linked Data (JSON-LD) is a simple syntax for representing linked data in JSON. By incorporating JSON-LD into Django templates, programmers can embed structured data directly into their web pages without changing the HTML structure. This is a flexible and scalable approach to generating rich metadata.

3.5. Enhancing Cultural Heritage Online Dissemination

The use of Semantic Markup Annotations with Schema.org, integrated into Django HTML templates and JSON-LD, provides cultural heritage institutions with a powerful mechanism for increasing the online visibility and engagement of their collections. This approach allows for the direct embedding of detailed metadata into web pages, making it easier for search engines and other digital services to find, index, and display this content in rich snippets and other enhanced search results [4].

For example, museums can use this technique to provide structured metadata for each artwork, such as title, creation date, medium, and physical dimensions. Similarly, archives can include metadata for historical documents that describe the creators, subjects, relevant dates, and locations associated with each item. Libraries can improve the digital representation of their collections by providing detailed metadata about the authors, titles, genres, and publication dates.

4. Example of Integrating Semantic Markup Annotations

Consider the following scenario: a museum curator or any authorized user wishes to enter and update data for a specific artwork in the system. Upon entry, the Django template generates the

corresponding structured data script using JSON-LD and Schema.org annotations. Here's how you can add such annotations to a Django template:

```
<script type="application/ld+json">
  {
 "@context": "http://schema.org",
 "@type": ["{{painting.art_form }}", "CreativeWork", "VisualArtwork"],
 "name": "{{ painting.title }}",
 "image": "{{ painting.image.url }}",
 "description": "{{ painting.description }}",
 "artMedium": "{{ painting.art_medium }}",
 "artform": "{{ painting.art_form }}",
 "dateCreated": "{{ painting.year_created }}",
 "sameAs": "{{ painting.same_as }}",
  "creator": {
   "@type": "Person",
   "name": "{{ painting.artist.name }}",
   "nationality": "{{ painting.artist.nationality }}",
   "birthDate": "{{ painting.artist.death_date }}",
   "deathDate":"{% if painting.artist.death_date %}{{ painting.artist.death_date
}} {% endif %}",
   "sameAs": "{{ painting.artist.same_as }}"

 }
 }
```

4.1. Dynamic Data Generation

When the template is rendered, it retrieves data from the database and populates the JSON-LD script. This script converts the artwork's details into structured data, making them understandable not only to humans but also to machines.

4.2. Improved Search Engine Interaction

The structured data on this page enables search engines to produce rich search results, which may include images, a thorough description, and direct links. The artwork is much easier to find and user interaction is significantly improved by this visibility.

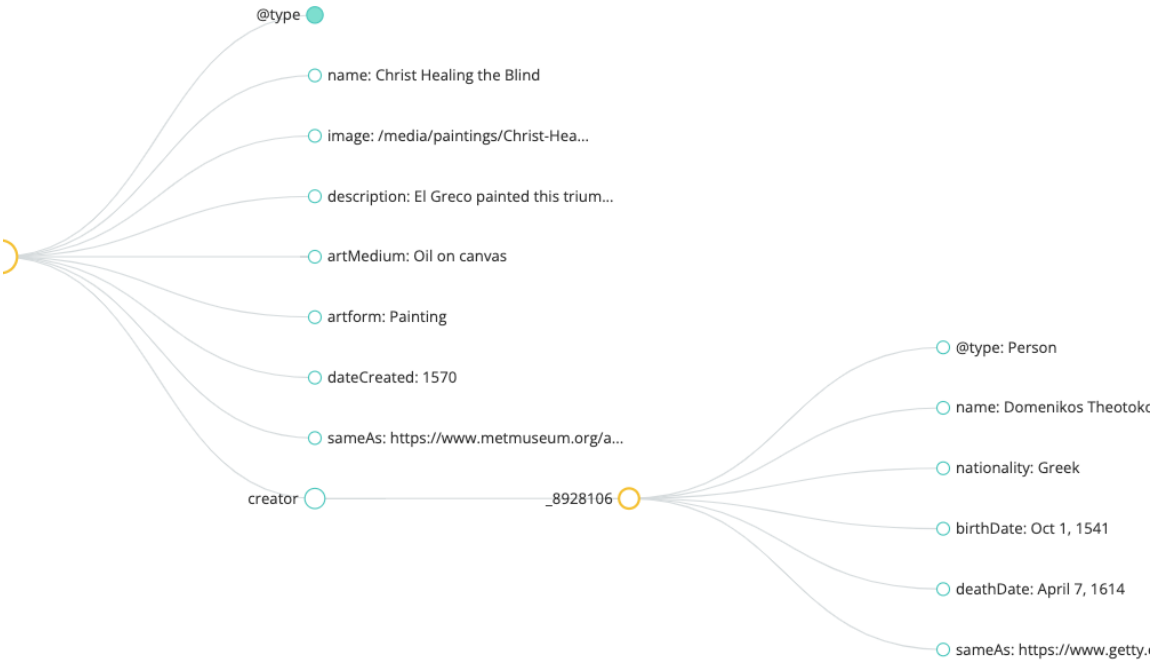

Figure 4: Virtual representation of the JSON-LD code. Graphic by the authors.

The online accessibility and dissemination of collections can be greatly improved by cultural heritage institutions by methodically incorporating Semantic Markup Annotations into their web frameworks. This approach helps to preserve and share a cultural heritage in the digital sphere while also enhancing the user experience by making valuable cultural assets easier to find.

4.3. Benefits and Impact

The integration of Semantic Markup Annotations with Schema.org, as used in Django templates and JSON-LD, offers several significant benefits for the online findability of cultural heritage content. This strategic implementation significantly improves the way cultural artifacts are shared and experienced on the web.

4.4. Improved Visibility and Discoverability

By embedding structured metadata directly into web pages, search engines, and other machines can better index and understand cultural heritage content. This improves visibility in search results, resulting in increased traffic and user engagement. Artworks, artifacts, and historical documents are made more accessible to a larger audience, which includes researchers, educators, and the general public. Using standardized schemas promotes interoperability across systems and platforms. Cultural heritage data structured in this manner can be seamlessly integrated and aggregated across multiple digital libraries, archives, and repositories. This interoperability promotes institutional collaboration and global knowledge sharing. The structured data enables the implementation of sophisticated search, filtering, and recommendation systems on cultural heritage websites. Users can find specific items more easily thanks to improved search capabilities that take advantage of rich metadata. Furthermore, this structure facilitates the creation of personalized experiences,

in which content recommendations are tailored to individual user interests and browsing history. As digital technologies advance, using a structured, well-documented format such as JSON-LD ensures that cultural heritage data is interpretable and usable, protecting it from technological obsolescence. This preservation is critical to ensuring the continuity of cultural knowledge over time. These technologies also improve educational opportunities by increasing the discoverability and accessibility of cultural heritage. They make it possible for learning experiences to be more dynamic and interactive, enabling the public and students to interact creatively and historically. This fosters a more knowledgeable and sophisticated society by deepening cultural understanding and appreciation.

5. Conclusion

Schema.org and Semantic Markup Annotations combined, as in Django templates and JSON-LD, provide a powerful means of promoting cultural heritage online. This method greatly improves the discoverability and interoperability of digital collections by structuring and organizing content into a machine-readable format. Above all, it improves the user experience by increasing the content's accessibility and interest.

Additionally, the preservation of cultural heritage information in a format that can be understood and used by future technologies is facilitated by the use of structured data. This is essential to ensuring that cultural to remains relevant in the quickly changing digital landscape. Additionally, it encourages the creation of intelligent web services that can better interact with this data, resulting in fresh perspectives and improved learning opportunities. Examples of these services include semantic search engines and AI-driven analysis tools.

Cultural heritage institutions can significantly contribute to public education and the advancement of cultural appreciation by cultivating a more engaged and educational online presence. By doing this, they help to preserve our collective heritage for future generations by fostering a greater appreciation and understanding of our diverse histories and traditions on a global scale.

To summarize, the integration of Schema.org Semantic Markup Annotations into web development processes represents a significant breakthrough in the preservation and sharing of cultural heritage. This methodology addresses the cultural imperative to preserve our history's findability, accessibility, and interpretability.

Declaration on Generative AI

During the preparation of this work, the authors used X-GPT-4 in order to: Grammar and spelling check. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication’s content.

References

[1] R. Wallis, A. Isaac, V. Charles, and H. Manguinhas, “Recommendations for the application of Schema.org to aggregated Cultural Heritage metadata to increase relevance and visibility to

search engines: the case of Europeana," The Code4Lib Journal, no. 36, Apr. 2017, Accessed: May 13, 2024. [Online]. Available: https://journal.code4lib.org/articles/12330.

[2] A. Garaba, "Medieval municipal buildings: development of a digital atlas to support historical research," 2020. [Online]. Available: https://webthesis.biblio.polito.it/14420/1/tesi.pdf

[3] A. J. G. Gray, C. Goble, R. C. Jimenez, and Bioschemas Community, "Bioschemas: From Potato Salad to Protein Annotation: 16th International Semantic Web Conference," Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks, 2017.

[4] A. Avgousti and G. Papaioannou, "The Current State and Challenges in Democratizing Small Museums' Collections Online," Information Technology and Libraries, vol. 42, no. 1, Art. no. 1, Mar. 2023, doi: 10.6017/ital.v42i1.14099.

[5] A. Avgousti, G. Papaioannou, and S. Hermon, "Enhancing Online Accessibility of Digitized Artifacts from Small Museum Collections in Cyprus: An Empirical Evaluation of the CyprusArk Solution," J. Comput. Cult. Herit., vol. 17, no. 3, p. 34:1-34:24, Apr. 2024, doi: 10.1145/3648229.

[6] R. V. Guha, "schema blog," Official blog for schema.org. Accessed: Apr. 29, 2023. [Online]. Available: http://blog.schema.org/2015/05/schema.html

[7] M. A. Matienzo, E. R. Roke, and S. Carlson, "Creating a Linked Data-Friendly Metadata Application Profile for Archival Description," International Conference on Dublin Core and Metadata Applications, vol. 0, pp. 112–116, 2017, Accessed: Jun. 14, 2021. [Online]. Available: https://dcpapers.dublincore.org/pubs/article/view/3860

[8] D. B. Lowe, J. Creel, E. German, D. Hahn, and J. Huff, "Introducing SAGE: An Open-Source Solution for Customizable Discovery Across Collections," The Code4Lib Journal, no. 52, Sep. 2021, Accessed: May 15, 2024. [Online]. Available: https://journal.code4lib.org/articles/15740.