LODAC Visualisation Whitepaper

Matt Thompson, Sysemia Ltd

October 29, 2013

Contents

1 Introduction

This paper is compiled from the work I did as part of a summer internship at the NII (National Institute of Informatics) in Tokyo, Japan. My goal in this internship was to learn the fundamentals of dealing with linked open data and how to apply them in useful ways.

Working with Sysemia Ltd in Bristol and the University of Bath, the purpose of my EngD is to use linked data to create better and more engaging museum exhibits for participants. Since this is the first year of the research portion of the degree, I am yet to finalise a research question, so this internship will also hopefully give me a better idea of some potential research areas.

Sysemia has interests in taking its experience in knowledge management and expanding into the realm of digital media in cultural institutions by consulting exhibition spaces on the use of digital devices and systems. The service is based on specialised knowledge in the exhibition sector as well as the development of digital knowledge webs, enabling curators to structure exhibition content data and visitors to interact with the exhibition space via accessing the knowledge web via digital devices.

The LODAC project is precisely the type of dataset that Sysemia will be working with. It contains information about Japanese museum and art gallery exhibits as well as the artists that created them and the museums where they are on display. This dataset was created by crawling the web sites of Japanese museums.

My initial goals for the internship were to find ways to visualise and browse the dataset. The end result would be a web service that could be used for people to discover new artists. They would be able to do this by finding artists that they like, then seeing which artists are similar to the one they have found, based on a visualisation or explicit recommendation done by the system.

2 Creating an art browser

My first project was to create a very simple site that browsed through the LODAC artworks based on their location and medium. I called the site "Bunka-kun".

2.1 Querying the LODAC database

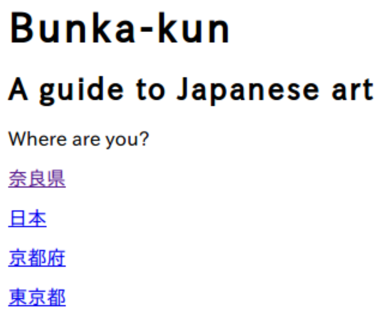
The site works by asking the user a series of questions and sending a SPARQL query to the LODAC server based on their answer. The back-end code was written in Clojure [1] and uses a domain specific language to form the SPARQL queries [5], then rendering an HTML page using Hiccup [3] based on the LODAC endpoint's response.

3 Visualising the links between artists

The "Bunka-kun" art browser was an interesting way to find certain types of art in a certain location, but I was more interested in the artists themselves and the links between them. What would be a good way to visualise the communities of artists and the links between them? The next project was to create a force-directed graph visualisation that clustered together different kinds of artists, called 'Artvis'.

3.1 Getting the data

The easiest way to find out information about the artists was to get their category information from DBPedia. This was done by first finding which artists existed in both DBPedia and



Bunka-kun

A guide to Japanese art

Where are you?

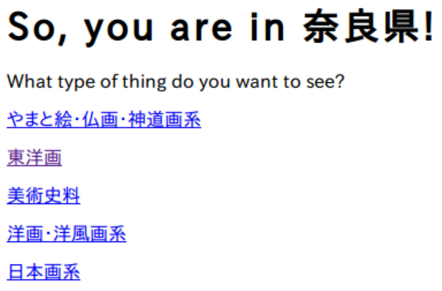
奈良県

日本

京都府

東京都

(a) Asking user for location



So, you are in 奈良県!

What type of thing do you want to see?

やまと絵・仏画・神道画系

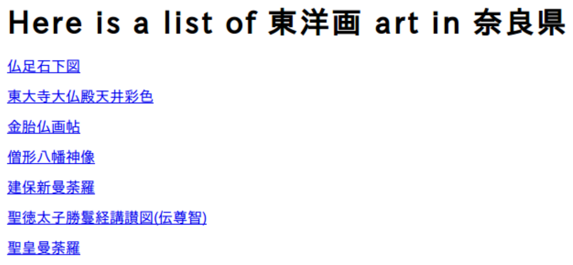
東洋画

美術史料

洋画・洋風画系

日本画系

(b) Asking user for type of art



Here is a list of 東洋画 art in 奈良県

仏足石下図

東大寺大仏殿天井彩色

金胎仏画帖

僧形八幡神像

建保新曼荼羅

聖徳太子勝鬘経講讃図(伝尊智)

聖皇曼荼羅

(c) Displaying a list of links to the LODAC pages for the matching artifacts

Figure 1: The "Bunka-kun" art browser

LODAC, simply by comparing the strings of the artists' names. This resulted in a total of about 200 artists with entries in both DBPedia and LODAC.

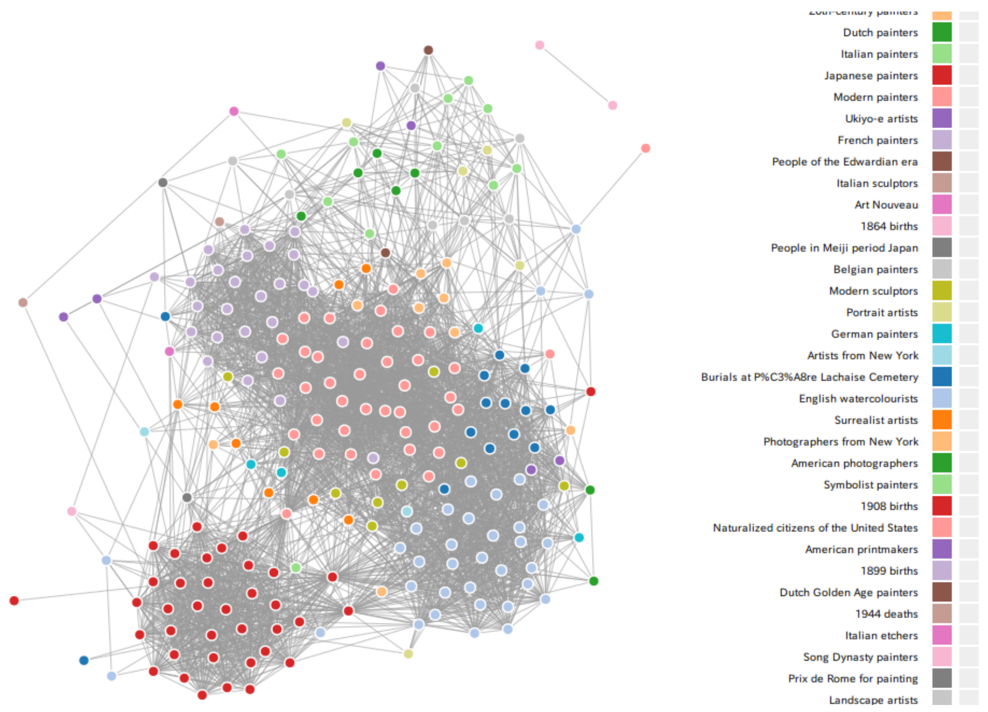
3.2 Making a force-directed graph

The next thing I made was a graph-based visualisation of the artists. The nodes of the graph were coloured and clustered according to the DBPedia category that each artist belonged to. The data used was gathered by querying the DBPedia SPARQL endpoint based on artists that exist in the LODAC database. This was then compiled into a JSON data format to be read by the Javascript D3 (Data Driven Documents) visualisation library [2] for display on the site.

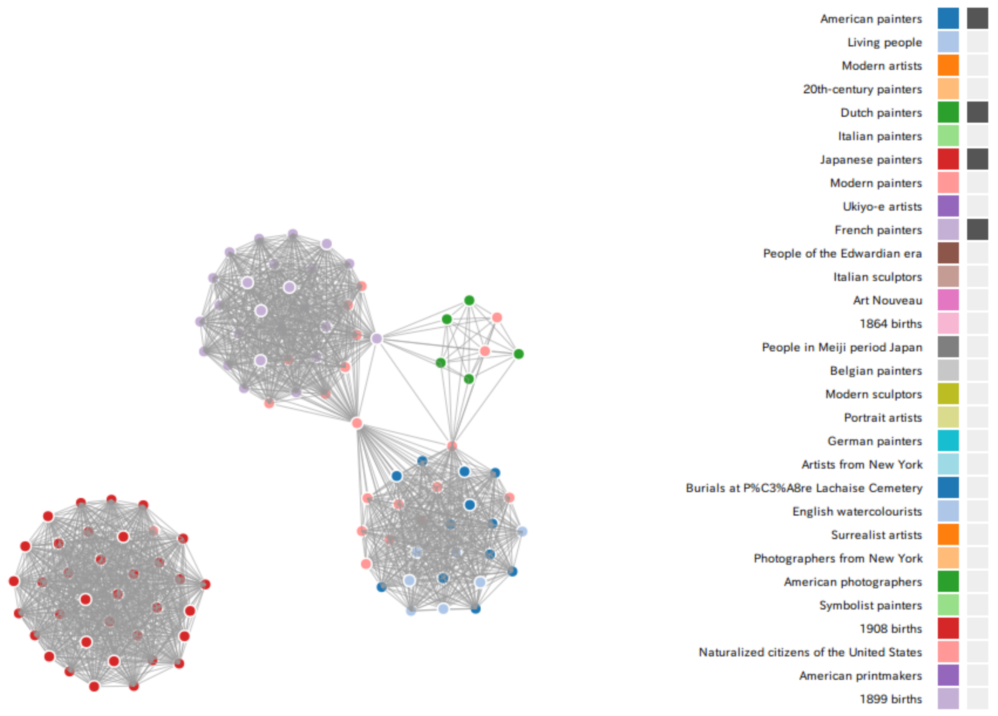
Some custom code to filter the artists by category was then added. The visualisation first shows all artists that it knows of. These artists are displayed as nodes on a force-directed graph, linked together if they share a category in the DBPedia database. This kind of visualisation naturally clusters the nodes together into communities that share the same categories. Each node was assigned a colour based on the category it shares with the greatest number of other artists. For example, say an artist belongs to the three categories of Japanese painters, Impressionists and Sculptors. If there are more sculptor artists in the dataset than Japanese painters or Impressionists, then the artist's node is assigned the colour that corresponds to sculptors. Subsets of artists can be selected by clicking on checkboxes next to a legend to the right of the visualisation.

In figure 2b, the user has clicked on the 'American painters', 'Dutch painters', 'Japanese painters' and 'French painters' checkboxes, therefore only the artists that belong to these categories appear in the visualisation.

Hovering the mouse over an artist's node displays the name of the artist and their most



(a) Showing all nodes on a graph. Hovering the mouse over a node shows the artist's name and main category.



(b) Artvis with the 'American painters', 'Dutch painters', 'Japanese painters' and 'French painters' categories selected.

Figure 2: The "Artvis" graph visualisation

popular category.

This type of visualisation is very convenient for seeing which artists bridge different communities. As shown in figure 2b, one artist, Yves Tanguy, is connected to both French Painters and American Painters.

The visualisation is available online at `http://cblop.com/artvis`.

4 Automatic taxonomy generation

The main problem with the Artvis visualisation was that the categories of artists appeared as a big, arbitrarily-ordered list. This is not ideal for browsing through different types of artists. Ideally, the user would be able to browse through a hierarchical taxonomy of categories of artists, drilling down from vague categories such as 'Japanese artists' down to more specific ones such as 'Edo period Ukiyo-e painters'.

Unfortunately, such a taxonomy does not exist for Wikipedia's or DBPedia's artist categories. This was an opportunity to see if one could be formed automatically from available data.

4.1 Gathering more data

For the Artvis visualisation, only artists from English DBPedia that also occurred in the LODAC ontology were used. This was a very small number of artists, only around 200.

In order to automatically create a taxonomy, more data from more artists would be required. I used the same basic string matching tools to link 172 artists from Japanese DBPedia, 280 from English DBPedia (using the sameAs attribute to find the Japanese equivalent from Japanese DBPedia) and 791 from Keio University's Wikipedia ontology at [4].

4.2 Heymann taxonomy generation algorithm

For the first attempt at taxonomy generation, I implemented a version of Heymann's automatic taxonomy generation algorithm [6]. Algorithm 1 shows the pseudocode for this algorithm.

The algorithm requires some kind of way of computing the similarity between two tags. In my implementation, I used a very simple score. The score for each pair of tags is how many times they appear together in an artist's list of DBPedia categories.

An example visualisation of the generated taxonomy is shown in figure 3. The full hierarchy can be browsed online at `http://cblop.com/tax_cat`.

4.3 Hierarchical clustering

Next, I tried a more low-level approach using hierarchical clustering to group the artists together. Using this method, the taxonomy was derived by looking at the most common categories that belonged to each cluster of artists and putting those categories together within the taxonomy.

Although this does not provide a hierarchical taxonomy, it is possible to derive this kind of taxonomy by cutting off the tree at certain points.

The dendrogram in figure 4 shows the clustering of artists. The diagram is coloured to show how the artists are clustered at a reasonably low level, with a threshold value of 6. Looking at the artist names in the diagram, it is clear that the names of non-Japanese artists in Katakana are grouped together as one would expect.

Algorithm 1 An extensible greedy algorithm for hierarchical taxonomy generation from social tagging systems using graph centrality in a similarity graph of tags.

```
Require: $L_{generality}$ is a list of tags $t_i, ...t_j$ in descending order of their centrality in the similarity graph.
Require: Several functions are assumed: $s(t_i, t_j)$ computes the similarity (using cosine similarity, for example) between $t_i$ and $t_j$. $getVertices(G)$ returns all vertices in the given graph, $G$.
Require: $taxThreshold$ is a parameter for the threshold at which a tag becomes a child of a related parent rather than of the root.
1: $G_{taxonomy} \leftarrow \langle \emptyset, root \rangle$
2: for $i = 1...|L_{generality}|$ do
3:   $t_i \leftarrow L_{generality}[i]$
4:   $maxCandidateVal \leftarrow 0$
5:   for all $t_j \in getVertices(G_{taxonomy})$ do
6:     if $s(t_i, t_j) > maxCandidateVal$ then
7:       $maxCandidateVal \leftarrow s(t_i, t_j)$
8:       $maxCandidate \leftarrow t_j$
9:     end if
10:  end for
11:  if $maxCandidateVal > taxThreshold$ then
12:    $G_{taxonomy} \leftarrow G_{taxonomy} \cup \langle maxCandidate, t_i \rangle$
13:  else
14:    $G_{taxonomy} \leftarrow G_{taxonomy} \cup \langle root, t_i \rangle$
15:  end if
16: end for
```

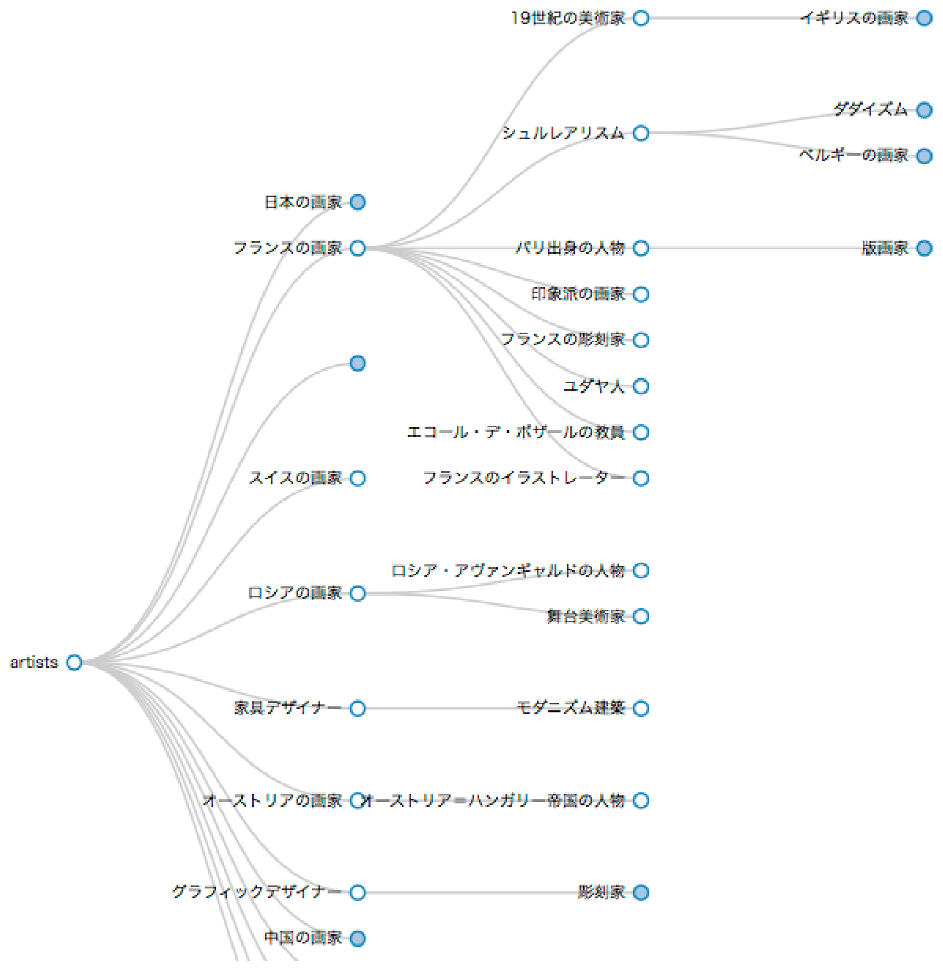


Figure 3: Taxonomy generated by using Heymann's automatic taxonomy-generating algorithm

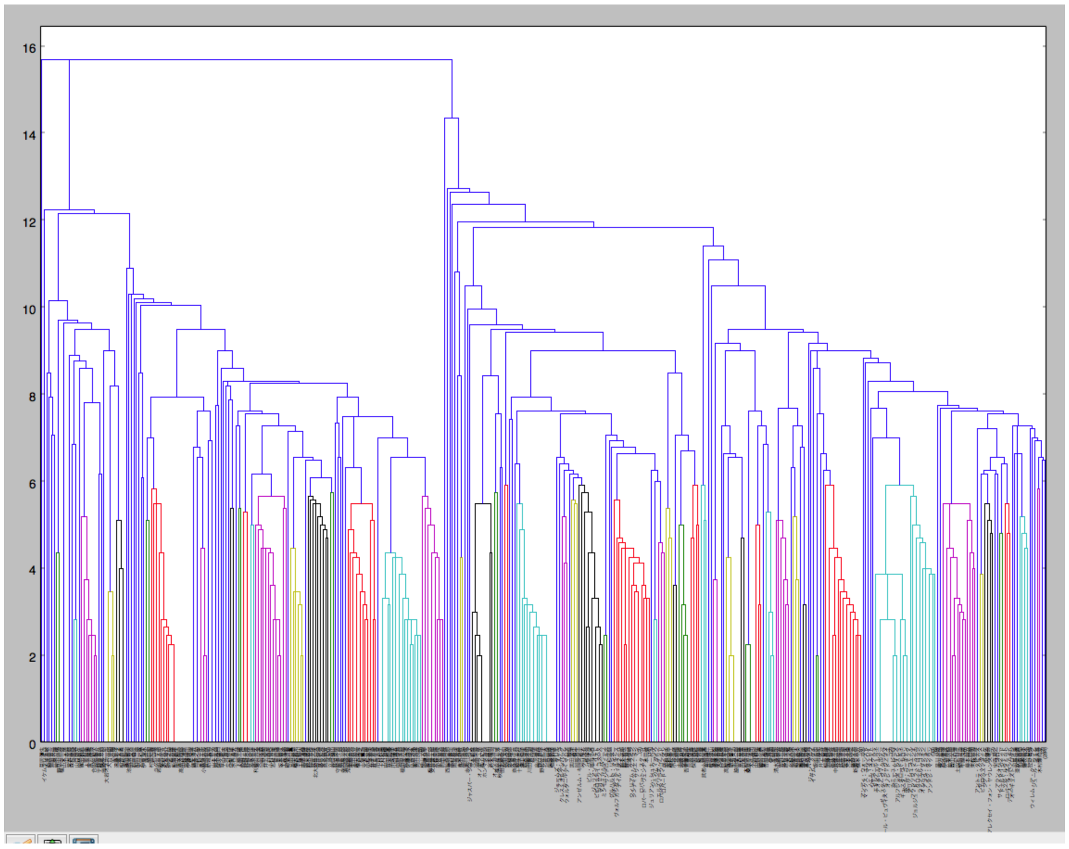


Figure 4: Dendrogram of the hierarchical cluster of artists, coloured with a threshold value of 6

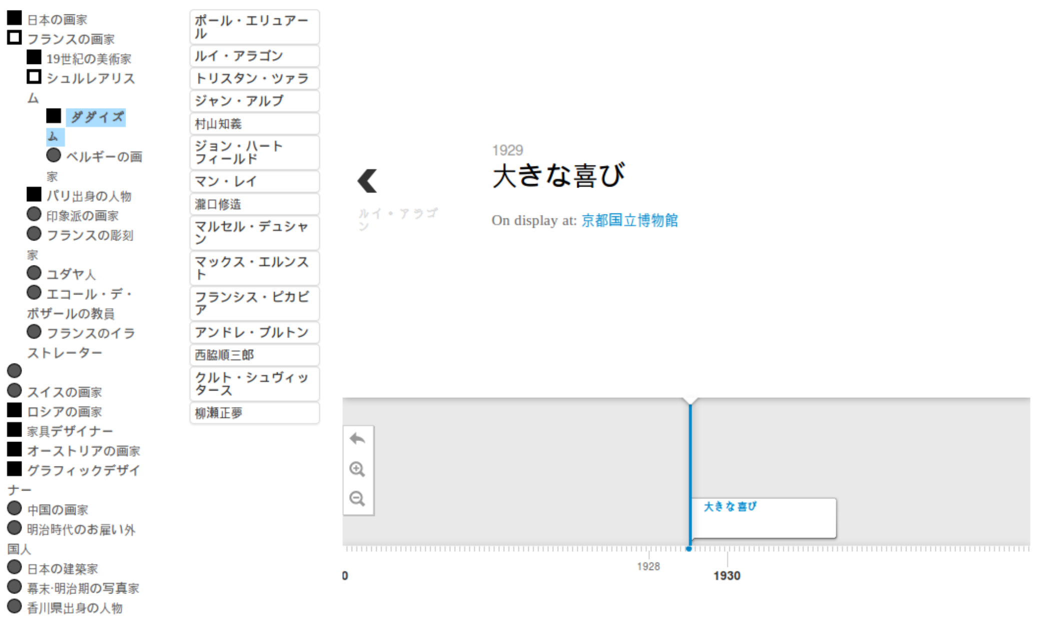


Figure 5: Screenshot of the artist timeline browser

5 Art browser app

5.1 Motivation

For the newcomer to Japanese art, it is difficult to discover new artists. I wanted to make a system where a user could browse through a taxonomy of artists to find ones that they know and have the system recommend new artists based on their selection. They would also be able to see other artists in the same level on the taxonomy to find artists within the same genre as ones that they like.

Also, it would be good to see all the artworks for a given artist and which museums in Japan have them on display. I decided to show all the artworks on a timeline to do this. A user browses a list of artworks that are chronologically organised for each artist.

5.2 Technology used

Using the taxonomy generated using Heymann's algorithm, a tree of artist categories is shown on the right. I used the Angular.js Javascript library to allow realtime filtering of the artists based on the selected categories.

Once an artist is selected, their timeline of artworks appears on the right. I used the Timeline.js library to read in a JSON file for each artist that holds a list of their artworks along with the dates they were created and where they can be found in Japan.

A screenshot of the resulting application can be seen in figure 5, and online at `http://cblop.com/timeline`.

6 Conclusions and future work

6.1 Facet-based browsing

Ideally, a better way of browsing throught the categories of artists would be used.

Although the hierarchies that were generated are informative, they are not enough on their own to facilitate a useful way of browsing through the artists. The problem is that the higher levels of the hierarchy tend to be nationalities. What if the user wants to find all impressionists, not just those that are French? Under the automatically generated taxonomy, this would not be possible.

With this in mind, a facet-based way of browsing through the artists should be explored. This would enable the user to browse taxonomies of artists based on (for example):

- Nationality
- Period
- Materials used
- Movement/genre

The user would be able to select one way of browsing the artists, then browse them based on the facet. Each facet wolud display a different taxonomy to the user.

Once the user clicks on a tag in the taxonomy, it is added to a list used to filter the artists that are displayed.

Also, a search feature should be added so that the user can simply type in the name of an artist or tag, and the esults would be displayed on the right.

References

[1] The clojure programming language. `http://clojure.org`.

[2] Data driven documents (d3), a javascript visualisation library. `http://d3js.org`.

[3] Hiccup html generation library for clojure. `https://github.com/weavejester/hiccup`.

[4] Keio wikipedia ontoloy. `http://www.wikipediaontology.org`.

[5] Matsu, a sparql dsl for clojure. `https://github.com/boutros/matsu`.

[6] Paul Heymann and Hector Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. 2006.