Louis McCallum

Introduction to Data Science Portfolio

Week 2 – Acquiring and Cleaning Data

This week we looked at sourcing data from APIs using Python's **requests** package

Met API

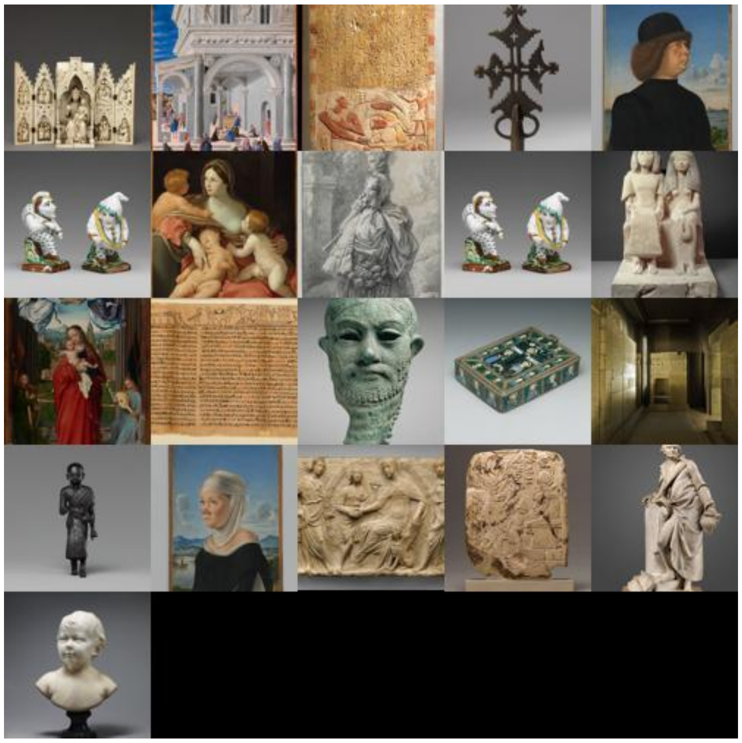
The initial "babies" query returned some images with infants in, but also some that didn't (e.g. this Jesus)

Searching for "rabbits" I got lots of the same images the query before. There is one image below that is new and has animals. This makes me think that there is something not working properly with the query function of the API?

Looking at the API documentation you can specify "title:True" and indeed, only the animal image actually has "rabbit" in the title.

Stopping searching for a timeframe then returned 187 images with rabbit in title! I got the images for the first 25

Unfortunately, still light on rabbits. Things returned include “Book of the Dead of the Priest of Horus, Imhotep (Imuthes)”, not rabbits in the title, as well as other busts and paintings. Clearly the API doesn’t work as documented and it's hard to find out why.

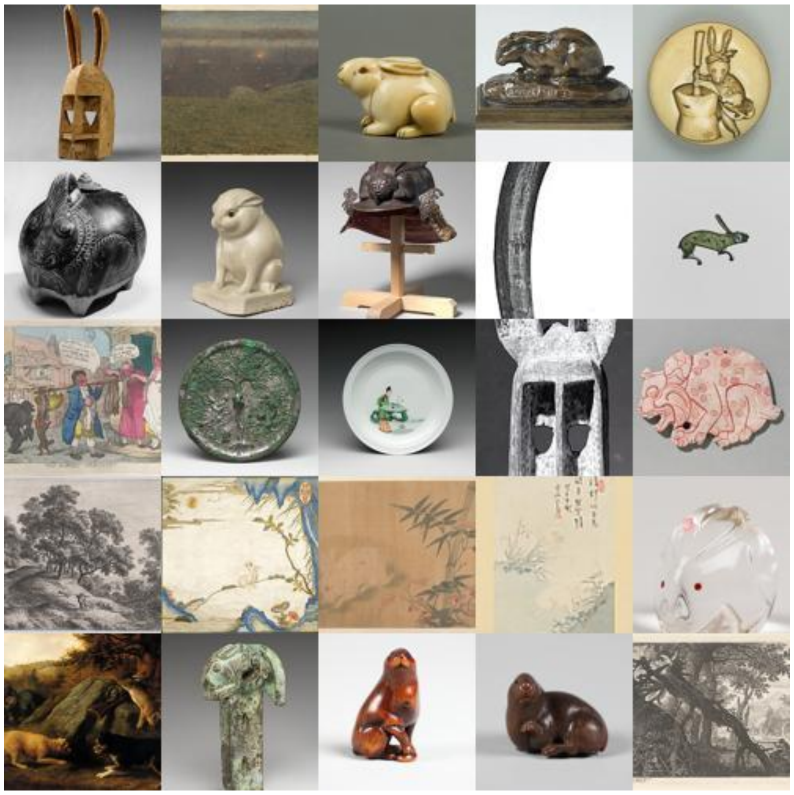


We can see that when you search the actual website for items with rabbit in the title then you get a much better set of rabbity results.

This is the request that gets sent

https://www.metmuseum.org/art/collection/search?showOnly=withImage&searchField=Title&q=rabbit

It returns the html for the actual page. Using this URL and the Beautiful soup library, I managed to parse out the images links and download and get the same results from code. This essentially is more akin to Web scraping than using an API however, in this situation it appeared that the API wasn’t working as well for queries as we wanted to a new approach was needed.



Now I have the actual pipeline I can play around!

I was interested to see what categories of art there were. I found “Ephemera” and that sounded interesting! It seems like its prints and photos?

I searched for “sport” in the description using the URL below

https://www.metmuseum.org/art/collection/search?showOnly=withImage&q=sport&material=Ephemera&searchField=Description&offset=120

When I change the page, I see that the “offset” parameter is added. This means I can use the URL to find which subset of 40 results I get back (there are over 2000!).

Why is it useful to have this as a web scraper as opposed to using the website?

It was useful to use the site to explore quickly and simply with its good interface.

BUT

I had an idea to pull use this image searching to pull in random images from a search and display in an updating grid. Obviously, to run automatically and be nice and flexible, it's not ideal to be manually searching and downloading images from a browser, then storing them elsewhere to be displayed.

If I can access the search results from my code, I can build an interesting site that runs live with little manual work.

Why is it useful to use an API instead of Web scraping?

I have found that web scraping from a live source (e.g. to populate another website) can be a real pain because of CORS requests (e.g. the Met have designed their site so it's hard to pull in a whole webpage from a JavaScript program). This exists for various security reasons.

With an API that's designed to be run programmatically, there are no such restrictions to its less work to integrate into a live project. If I was just running the code once to scrape some data for training, this would be fine, but for running a live project I would be concerned about sustainability (what if they cut me off!).

This shows the tension that exists when an institution / company decides to have an API, it requires some upkeep (e.g. the Met is hosting something, but the query either doesn't work well, or has broken, or in a way that is not documented). But they get to have control over what is shared and how and it means people are less likely to scrape in a less regulated manner.

Grid Site Example

To get around the CORS issues I had to build a Flask App that acted as a proxy that was called from the same origin as the website that was showing the image grid.

The image grid was implemented in JavaScript and

1. Gets the search results page (html) for a random offset (any number below 2000)
2. Extracts the image URLs
3. Displays them in a grid
4. Every 250ms a grid location is switched to a new image
5. Every 10 seconds a new set of 40 results is returned (new random offset)

This switching of offsets tends to get a nice slow change of images because we get a new list of possible images (that are usually related visually or thematically) and gradually they are introduced over the 10 seconds. It's nicer than just changing the whole 25 images in one go!

Code

All code is in the GitHub repository https://git.arts.ac.uk/lmccallum/intro-to-ds-portfolio

Most code is started by prompting ChatGPT then tailoring for specific use cases

Video

Video documentation is also provided in the repo (met-sports.mov)