Cultural Heritage ASessor (CHAS)

CHAS: Cultural Heritage ASsessor

Author: Ruben Schalk & Joop Vanderheiden
Affiliation: Rijksdienst voor het Cultureel Erfgoed
Date: 2023-10-12 14:37:22

Content

Background

Many museums, libraries, archives and other cultural heritage institutions are working on digitally disclosing their collections. In The Netherlands, publishing these data in the linked data format is the format of choice . By using linked data, collections can be shared in an inter-operable manner. What is more important, when using linked data connections can be made with other data, live on the web. In this way, paintings from a specific artist can be connected to another painting by the same artists in another museum, even when the first museum is unaware of its existence. For users this is great: building applications on top of linked data collections allows them to discover related objects across institutions across the world.

Great as this sounds, publishing linked data and using its fullest potential comes with challenges. First, most collections management systems don't yet publish linked data. Second, and most importantly, linked data only becomes truly linked if the metadata uses URI's from publicly available thesauri or other term lists. For example, if museum A only denotes Van Gogh as `Gogh, Vincent van`, and museum B only uses `Vincent van Gogh`, no connection can be made between related objects. However, if both museums also provide a URI (more or less a URL) from a thesauri to describe Van Gogh - for instance https://data.rkd.nl/artists/32439 - then we can use this URI to 'travel' between objects of the same artist (and all related information) when the collection is published as linked data. The same goes for other key variables, such as materials, genres, categories, etc. Ideally, GLAMs should incorporate URI's from thesauri like the Cultural Heritage Thesaurus, curated bibliographies or others to describe their objects.

CHAS explained

As long as publishing thesauri-enriched linked data is not yet the industry standard, GLAMs benefit from tools, tips and tricks to help them get started. This is what we aim to do with the pipeline presented here. We have called it the Cultural Heritage ASsessor (or CHAS). CHAS presents a combination of scripts, templates and SPARQL queries that help to:

- convert records to linked data;
- assess to what degree these records can be enriched with thesauri or other term list, *without* first harmonizing the data.

Together, these steps assist GLAMs to publish linked data, and assess how easily their data can (or cannot be) enriched with URI's from public thesauri. In the remainder of this explanation we'll use two sample datasets to illustrate the process, provided by the Dutch Cultural Heritage Agency (RCE).

If you want to try out the pipeline for your own collection, consult the README of the CHAS repo and use the CHAS_report file to generate a report on your dataset. All required applications are available as open source (apart from Excel, but there are alternatives…). Note that some experience with programming, SPARQL and/or data science may be required.

From CMS to csv

We start with an export from a collections management system, which is a csv file in our example. Should you have an XML file, you can convert this to csv using software like Excel or Openrefine, or online converters like convertio. The choice to start with a csv is intentional: it forces us to keep the datamodel as 'flat' as possible, which helps to maintain an easy overview of the data. This suits our purpose of assessing collection data. When 'officially' publishing collections as linked data it is likely that the datamodel becomes more layered to keep as much context as possible.

The export csv contains several hundreds objects with a colonial connection from the RCE Art Collection.

In Excel or Openrefine we first select only the columns we want to keep. Many fields are not required for our exercise and only make the process more cumbersome. For example, our dataset originally has hundreds of (mostly empty) columns. We only keep those that have information that can be mapped to thesauri: material, subject, creator, technique and location. We delete the unwanted columns, rename the others to something we can understand, and save the edited csv. You can download the edited sample csv from the CHAS repository. We have kept an identifier columns to allow merging of edits with the original records. The first rows now look like this (without the lengthy column 'description'):

priref	associated_subject_1	associated_subject_2	associated_subject_3	associated_subject_4	material_1	material_2	object_category	object_name	object_number	creator	production_place	acquisition_place	technique
13279	sculptuur	beeldhouwkunst	masker	dier	hout	verf	ethnografica	figuurplastiek	AB15389	onbekend	Kameroen	Rotterdam	
13281	sculptuur	beeldhouwkunst	masker	dier	hout	touw	ethnografica	figuurplastiek	AB15390	Igbo, Volk van	Kameroen	Rotterdam	

priref	associated_subject_1	associated_subject_2	associated_subject_3	associated_subject_4	material_1	material_2	object_category	object_name	object_number	creator	production_place	acquisition_place	technique
13282	sculptuur	beeldhouwkunst	vrouw		hout	verf	ethnografica	figuurplastiek	AB15391	Senufo, Volk van	Ivoorkust	Rotterdam	gepatineerd
13284	sculptuur	beeldhouwkunst	man	masker	staal	hout	ethnografica	figuurplastiek	AB15393	Igbo, Volk van	Nigeria	Rotterdam	
13285	sculptuur	beeldhouwkunst	masker		hout	mensenhaar	ethnografica	figuurplastiek	AB15394	Beneden-Sepik, Volk van	Papoea-Nieuw-Guinea	Rotterdam	
13333	sculptuur	beeldhouwkunst	fragment	archeologica	aardewerk	pigment	ethnografica	figuurplastiek	AB15425	Teotihuacan, Volk van	Mexico	Rotterdam	

From csv to linked data

Different options are available to convert the edited csv to linked data. It can be done using Python libraries, Openrefine (with RDF extension), the LD Wizard, or COW: CSV on the web converter.

Because we have modest requirements, we'll use LD Wizard. It's very simple: upload the csv, select which vocabularies you want to use for the predicates (columns), edit your preferences (such as the base URI), and simply click 'Next' to convert to linked data. If you require more advanced data manipulation, e.g. adding additional relations, string manipulations, etc., then you can download and edit the COW-metadata file, and run the conversion again through either using COW or LD Wizard. A wiki how to use COW can be found at the COW Github repo.

To keep the data simple, we try to map all variables to related properties in Dublin Core or Dublin Core Terms, except for `object_name` and `technique`, for which we use `schema:artform`. All this can be changed easily according to your preferences. Every object is defined as an instance of `edm:ProvidedCHO`. A colonial object now looks like this when expressed in linked data format (.nt):

Subject	Predicate	Object
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	https://linkeddata.cultureelerfgoed.nl/europeana/colonial/def/priref	"36682" .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/subject	"Oosters tapijt" .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/subject	"tapijt" .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/description	"Tabriz-tapijt, van wol en katoen" .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/publisher	"Rijksdienst voor het Cultureel Erfgoed" .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/spatial	"AMERSFOORT" .

Subject	Predicate	Object
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/spatial	“DOORN” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/spatial	“Iran-Noordwest” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/medium	“wol (textiel)” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/medium	“katoen” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/type	“toegepaste kunst” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	https://schema.org/artform	“geknoopt” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	https://schema.org/artform	“Tabriz-tapijt” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://purl.org/dc/terms/creator	“onbekend” .
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/B3135	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.europeana.eu/schemas/edm/ProvidedCHO .

The converted and mapped sample dataset has been uploaded to the Linked Data environment of the Dutch Cultural Heritage Agency. This allows to add a SPARQL service that we can use to connect the colonial dataset to thesauri and other reference datasets as long as it can be accessed online, preferably using a SPARQL endpoint.

If you don’t have a triplestore at hand (which is quite likely), there are user-friendly open-source alternatives available such as GraphDB desktop version, or free online triplestores such as TriplyDB.

Overview of terms

Now the fun part can begin. How good is the quality of our collection data? First we want to see what our terms actually look like. Are they already to some degree standardized? We’ll retrieve all the relevant terms from our dataset using a SPARQL query; a method that we’ll continue to use throughout this demo. The results are limited to 20, but you can set a higher figure. We see that top-20 terms in our colonial dataset are relatively standardized, but with some exceptions like ‘wol (textiel)’ and the compounded ‘Oosters tapijt’.

```
url <- "https://api.linkeddata.cultureelerfgoed.nl/datasets/ruben-schalk/colonial/services/europeana-co
sparql_query <- "PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema: <https://schema.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

            SELECT ?obj (count(?obj) as ?n) WHERE {

                    ?sub dct:medium|dct:subject|schema:artform ?obj .


                  }
ORDER BY DESC (?n)
LIMIT 20"


response <- POST(url, body = list(query = sparql_query))
content <- content(response, "parsed")
df <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))
print(df)
```

```
##                obj  n
## 1          textiel 63
## 2  Oosters tapijt 60
## 3           tapijt 60
## 4   wol (textiel) 60
## 5   beeldhouwkunst 50
## 6         geknoopt 49
## 7        sculptuur 46
## 8   figuurplastiek 35
## 9      miscellanea 35
## 10       aardewerk 31
## 11     archeologica 30
## 12        fragment 30
## 13         pigment 30
## 14          zilver 25
## 15            hout 22
## 16          katoen 22
## 17          kralen 21
## 18         geweven 20
## 19           zijde 17
## 20             tas 14
```

Connecting to thesauri and reference data

Can we improve the usability of the collection by adding URI's from thesauri? Let's find out.

Cultural Heritage Thesaurus

We'll first try to connect the `dct` and `schema:artform` fields to labels in the Cultural Heritage Thesaurus (CHT). Many Dutch GLAMs use this thesaurus already, so adding some of these URI's helps to connect our data to theirs. The code below shows a SPARQL query that fetches keywords from the colonial dataset and checks whether these occur (in exactly the same spelling) in the SKOS `prefLabel`, `hiddenLabel`, or `altlabel` of CHT concepts.

The same principle can be repeated for other thesauri, by simply adapting the URL behind `SERVICE` to the corresponding SPARQL endpoint of another thesaurus. (Note: the GRAPH clause is required only for the CHT to exclude some reference graphs).

```
url <- "https://api.linkeddata.cultureelerfgoed.nl/datasets/ruben-schalk/colonial/services/europeana-col
sparql_query <- "PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema: <https://schema.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

                 SELECT (count(distinct(?obj)) as ?terms) (count(distinct(?uri_cht)) as ?cht_matches) WH

                    ?sub dct:medium|dct:subject|schema:artform ?obj .

                    BIND(strlang(?obj, 'nl') AS ?obj_nl)

                SERVICE <https://api.linkeddata.cultureelerfgoed.nl/datasets/rce/Cultuurhistorische-Thes

                      GRAPH <https://data.cultureelerfgoed.nl/term/id/cht/thesaurus> {

                        optional{   ?uri_cht skos:prefLabel|skos:altLabel|skos:hiddenLabel ?obj_nl . }

    }}
                  }
    LIMIT 10"


response <- POST(url, body = list(query = sparql_query))
content <- content(response, "parsed")
df <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))
print(df)
```

```
##   terms cht_matches
## 1   180         122
```

We see that from the selected keyword fields (180 in total), 122 have an exact match with a concept in the thesaurus. Not a bad score, but also one that can be improved by data harmonization.

Instead of a count of matches, we can also add the retrieved CHT concepts with their URI's to the linked data. One way to do so would be to again find matches through SPARQL, use the `CONSTRUCT` clause to generate new triples, and then upload these to the linked data. That query would look something like this:

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema: <https://schema.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

CONSTRUCT  {
  ?sub dct:subject ?uri_cht .

}
WHERE {

  ?sub dct:subject ?obj .

  BIND(strlang(?obj, "nl") AS ?obj_nl)

SERVICE <https://api.linkeddata.cultureelerfgoed.nl/datasets/rce/Cultuurhistorische-Thesaurus-CHT/servi

    GRAPH <https://data.cultureelerfgoed.nl/term/id/cht/thesaurus> {

         ?uri_cht skos:prefLabel|skos:altLabel|skos:hiddenLabel ?obj_nl .

    }}
} LIMIT 10
```

For example, whereas earlier the triple for an object would have been:

Subject	Predicate	Object
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/AB15398	dct:subject	"archeologica"

Now the URI of the concept 'archeologica' from the Cultural Heritage Thesaurus can be added:

Subject	Predicate	Object
https://linkeddata.cultureelerfgoed.nl/cho-kennis/colonial/id/AB15398	dct:subject	https://data.cultureelerfgoed.nl/term/id/cht/b21e06cc-eac4-47d0-bdc6-399ee89747eb

This can be repeated for the other predicates for which we found concepts in the thesaurus.

Colonial History thesaurus

The 'Koloniaal Verleden' term list contains terms that can be associated with a colonial heritage. Think of terms like 'ivory', 'spices', 'sugar' or 'textiles'. It provides a nice example to show that we only need to adapt the query modestly when we want to connect our sample data to another thesaurus that uses the same datamodel (`SKOS` in this case). To do this, we've only removed the `GRAPH` clause (not needed for this thesaurus), adapted the endpoint URL, and renamed the variables for clarification:

```
url <- "https://api.linkeddata.cultureelerfgoed.nl/datasets/ruben-schalk/colonial/services/europeana-co
sparql_query <- "PREFIX dct: <http://purl.org/dc/terms/>
                  PREFIX schema: <https://schema.org/>
                  PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
                  PREFIX dc: <http://purl.org/dc/elements/1.1/>
                  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
                  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

                 SELECT (count(distinct(?obj)) as ?terms) (count(distinct(?uri_koloniaal)) as ?koloniaal

                    ?sub dct:medium|dct:subject|schema:artform ?obj .

                    BIND(strlang(?obj, 'nl') AS ?obj_nl)

                SERVICE <https://api.linkeddata.cultureelerfgoed.nl/datasets/rce/Koloniaal-Verleden/ser


                        optional{   ?uri_koloniaal skos:prefLabel|skos:altLabel|skos:hiddenLabel ?obj_n
                }
                      } LIMIT 10"


response <- POST(url, body = list(query = sparql_query))
content <- content(response, "parsed")
df <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))
print(df)
```

```
##   terms koloniaal_matches
## 1   180                 9
```

Interestingly, we can only match nine keywords to this thesaurus, even though the sample data should be exclusively about colonial objects. The identification of whether these objects had a potential colonial affiliation was done manually, which likely explains this.

Geonames

Next, we'll try to harmonize spatial keywords. Geonames has been made available as linked data for Dutch, Belgian and German placenames. Unfortunately, we still lack a truly global overview of current and historical placenames, but this is a start. The query below matches distinct terms from the colonial dataset to this Geonames subset. Out of the 55 unique spatial keywords in the data, 10 have a match with a Geonames entity.

```
url <- "https://api.linkeddata.cultureelerfgoed.nl/queries/ruben-schalk/colonial-geonames/run"
sparql_query <- "PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema: <https://schema.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skosxl: <http://www.w3.org/2008/05/skos-xl#>
PREFIX aat: <http://vocab.getty.edu/aat/>
PREFIX gvp: <http://vocab.getty.edu/ontology#>
PREFIX gn: <https://www.geonames.org/ontology#>
```

```
SELECT (count(distinct(?obj)) as ?colonial_spatial) (count(distinct(?uri)) as ?geonames_match) WHERE {
  ?sub dct:spatial ?obj .

  SERVICE <https://demo.netwerkdigitaalerfgoed.nl/geonames/sparql> {
    optional { ?uri gn:name|gn:alternateName ?obj }
  }
}
LIMIT 10"

response <- POST(url, body = list(query = sparql_query))
content <- content(response, "parsed")
df <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))
print(df)
```

```
##   colonial_spatial geonames_match
## 1               55             10
```

Cultural Heritage Thesaurus: test 2 #

Now that the idea is hopefully clear, we'll move to another dataset: a sample of the art collection kept by the Dutch Cultural Heritage Agency. The nice thing about the code chunks is that we can just copy a query from above. We only need to adapt the `url` variable so that it calls the SPARQL endpoint of this dataset. Because the LOD conversion is done in the same manner, the predicates are the same as in the colonial dataset. Let's test the terms from this collection against the CHT:

```
url <- "https://api.linkeddata.cultureelerfgoed.nl/datasets/ruben-schalk/rce-art-sample/services/rce-ar
sparql_query <- "PREFIX dct: <http://purl.org/dc/terms/>
PREFIX schema: <https://schema.org/>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

                 SELECT (count(distinct(?obj)) as ?terms) (count(distinct(?uri_cht)) as ?cht_matches) W

                    ?sub dct:medium|dct:subject|schema:artform ?obj .

                    BIND(strlang(?obj, 'nl') AS ?obj_nl)

                SERVICE <https://api.linkeddata.cultureelerfgoed.nl/datasets/rce/Cultuurhistorische-The

                      GRAPH <https://data.cultureelerfgoed.nl/term/id/cht/thesaurus> {

                        optional{   ?uri_cht skos:prefLabel|skos:altLabel|skos:hiddenLabel ?obj_nl . }

    }}
                  }
    LIMIT 10"


response <- POST(url, body = list(query = sparql_query))
content <- content(response, "parsed")
```

```r
df <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))
print(df)
```

```
##   terms cht_matches
## 1   279         192
```

That's quite a nice score: about 68 per cent of terms can be matched directly against the CHT.

Contested Terms: Words Matter Knowledge Graph

Now that the replication seems to work, a final assessment is to map descriptions of objects against a knowledge graph of contested terms. Using expert-advice, a group of researchers recently published a list of terms that could be considered as contested in descriptions of cultural heritage. See their paper here: https://ir.cwi.nl/pub/33129/. A copy of this graph can be queried at the linked data environment of the Dutch Cultural Heritage Agency: https://linkeddata.cultureelerfgoed.nl/rce/Words-Matter-Knowledge-Graph.

In the context of decolonizing collections, it would be useful to get a quick overview of potentially contested terms in a collection, without having to go through all the descriptions manually. That is what our final query aims to do. It retrieves the descriptions from the art collection sample, tries to match these against the Words Matter knowledge graph, and returns the potential contested terms for each object description.

This code is more comprehensive, but if you know your way around R (or a colleague does), you can simply copy and paste the code below, and again adapt the `url` variable to your own dataset.

```r
### import descriptions as text from collection ###

url <- "https://api.linkeddata.cultureelerfgoed.nl/datasets/ruben-schalk/rce-art-sample/services/rce-ar
query <- "PREFIX dct: <http://purl.org/dc/terms/>
          PREFIX schema: <https://schema.org/>
          PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
          PREFIX dc: <http://purl.org/dc/elements/1.1/>
          PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
          PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

          SELECT ?identifier ?description WHERE {

           ?identifier dct:title ?description .


          }
          LIMIT 500"

response <- POST(url, body = list(query = query))
content <- content(response, "parsed")
colonial <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))

### import terms from word matter knowledge graph ###

url_wordsmatter <- "https://api.linkeddata.cultureelerfgoed.nl/datasets/rce/Words-Matter-Knowledge-Grap
query <- "PREFIX dct: <http://purl.org/dc/terms/>
          PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
          PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
          SELECT DISTINCT ?obj WHERE {
            ?sub <http://www.w3.org/2008/05/skos-xl#literalForm> ?obj .

            FILTER(LANG(?obj) = 'nl')
            } "

response <- POST(url_wordsmatter, body = list(query = query))
content <- content(response, "parsed")
wordsm <- as.data.frame(do.call(rbind, lapply(content, as.data.frame)))

### Find terms from description in collection in words matter graph ###


# Function to check if any word from word_df appears in the text_variable of text_df

check_words_in_text <- function(word_df, text_df, word_column, text_variable) {
  # Create an empty logical vector to store results
  match_results <- logical(nrow(text_df))

  # Create an empty list to store the matched words for each row
  matched_words_list <- vector("list", length = nrow(text_df))

  # Loop through the words in word_df and check if they appear in any row of text_df's text_variable
  for (i in seq_len(nrow(text_df))) {
    text <- text_df[[text_variable]][i]
    words <- stri_extract_all_regex(text, paste0("\\b", word_df[[word_column]], "\\b"), opts_regex = li
    matched_words <- unique(na.omit(unlist(words)))
    if (length(matched_words) > 0) {
      match_results[i] <- TRUE
      matched_words_list[[i]] <- matched_words
    }
  }

  # Convert the list of matched words to a character vector
  matched_words <- sapply(matched_words_list, function(x) {
    if (length(x) == 0) return("")
    paste(x, collapse = ", ")
  })

  return(list(match_results, matched_words))
}

# Call the function to find matches between 'obj' in wordsm and 'description' in colonial
results <- check_words_in_text(wordsm, colonial, "obj", "description")

# Extract the match_results and matched_words from the results list
matches <- results[[1]]
matched_words <- results[[2]]

# Add the match_results and matched_words as new columns to colonial
colonial$matches <- matches
colonial$matched_words <- matched_words
setDT(colonial)
```

```
#print(colonial[matched_words != "", ]) # unselected for now because we show the results in markdown in
```

Identifier	Description	Matches	Matched Words
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_131257	West-Indisch huis	TRUE	Indisch
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_131258	West-Indisch huis	TRUE	Indisch
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_131260	West-Indisch huis	TRUE	Indisch
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_131264	West-Indisch huis	TRUE	Indisch
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_133248	Batavia	TRUE	Batavia
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_133250	Batavia	TRUE	Batavia
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_133251	Batavia	TRUE	Batavia
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_53533	Emilia van Nassau-Beverweerd (1635-1688), met een bediende	TRUE	bediende
https://linkeddata.cultureelerfgoed.nl/cho-kennis/rce-art-sample/id/rce_58790	Wit huis bij maanlicht St. Alban de Montbel	TRUE	Wit

We see that some terms match with the Words Matter graph, such as, 'wit' ('white'), and 'bediende' ('servant'). This does not mean they are contested per se (for this you should consult the Words Matter graph in detail), but that some caution may be required in using these terms. How to deal with terms like these is eventually up to yourself. Nevertheless, this code chunk and the SPARQL query can save a lot of time in checking their metadata for potentially contested terms.

More Thesauri

There are more thesauri available to which you can connect your linked data collection. The principle is the same as above: find the endpoint URL, and (if needed) adapt the SPARQL query to find matched terms between your collection and the thesaurus.

Below is an overview of thesauri with their SPARQL endpoints. You can also find these at the Termennetwerk website, but be sure to then only copy the part of the URL after 'reconcile/'.

SPARQL endpoints for thesauri (in Dutch):

Thesaurus	Provider	Endpoint
Adamlink: straten in Amsterdam	Adamlink	Link
Archeologisch Basisregister	RCE	Link
Art & Architecture Thesaurus	Getty	Link
Brabantse gebouwen	Erfgoed Brabant	Link
Brinkman trefwoordenthesaurus	KB	Link
Cultuurhistorische Thesaurus	RCE	Link
EuroVoc	European Union	Link
GeoNames (NL, BE, GER)	Geonames	Link
Goudse straten	Gouda Time Machine	Link
GTAA: classificatie	Beeld & Geluid	Link
GTAA: genres	Beeld & Geluid	Link
GTAA: geografische namen	Beeld & Geluid	Link
GTAA: namen	Beeld & Geluid	Link
GTAA: onderwerpen	Beeld & Geluid	Link
GTAA: onderwerpen beeld-geluid	Beeld & Geluid	Link
GTAA: persoonsnamen	Beeld & Geluid	Link
Homosaurus	IHLIA	Link
Iconclass	Henri van de Waal	Link
Indisch Erfgoed Thesaurus	IHC	Link
Koloniaal Verleden	RCE	Link
Muziek: genres en stijlen	Muziekweb	Link
Muziek: personen en groepen	Muziekweb	Link
Muziekschatten: onderwerpen	SOM	Link
Muziekschatten: personen	SOM	Link
Nederlandse Thesaurus van Auteursnamen	KB	Link
RKDartists	RKD	Link
STCN: drukkers	KB	Link
Thesaurus Nationaal Museum van Wereldculturen	NMVW	Link
Uitvoeringsmedium	Podiumkunst.net	Link
Wikidata	Wikidata	Link
WO2-biografieën	NIOD	Link
WO2-thesaurus	NIOD	Link

Conclusion

We hope the principle is clear. Once you have a linked data version of your collection, it's possible to use the SPARQL queries showed here to asses your metadata by connecting to different thesauri and term lists. You can also run these queries outside of R Studio: in a SPARQL editor such as Yasgui, or within your own triplestore environment. To accommodate this, the queries are also available separately at CHAS repository/queries.

Future steps

We aim to also provide this pipeline in Python. What we have not yet aimed for is to perform fuzzy matching or reconciliation on terms or creator names in a collection. This can already be done using Openrefine in combination with the Network of Terms Reconciliation service. However, integrating Network of Terms reconciliation with our R/Python pipeline may be useful if you start the process from linked data instead of a csv/XML.

Disclaimer

The pipeline provided here is experimental in nature. It means that it is still under development and may not be free from errors or issues. While we have made every effort to ensure its accuracy, there could be unforeseen results. Its use is entirely at your discretion and risk. It is advisable to thoroughly evaluate its performance and suitability for your specific needs before relying on it extensively.

Your feedback is important to us. If you encounter any issues, have suggestions for improvements, or wish to share your experiences, please do not hesitate to reach out to us. Your input will help refine and enhance this pipeline.

Question and remarks: thesauri@cultureelerfgoed.nl.

Licence

MIT Licence