DataDeps.jl

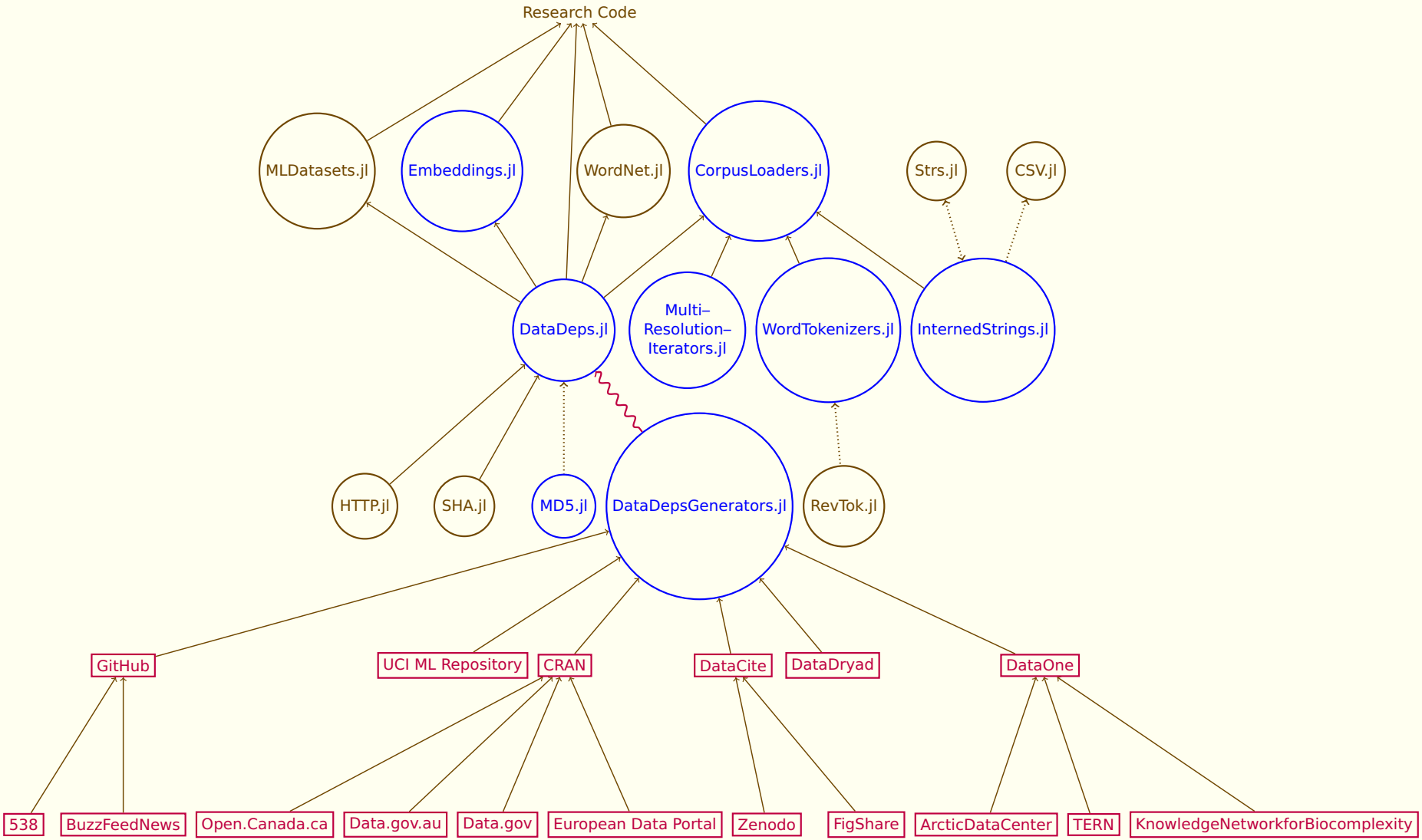
and other foundational tools for data driven research (Especially NLP)

Lyndon White

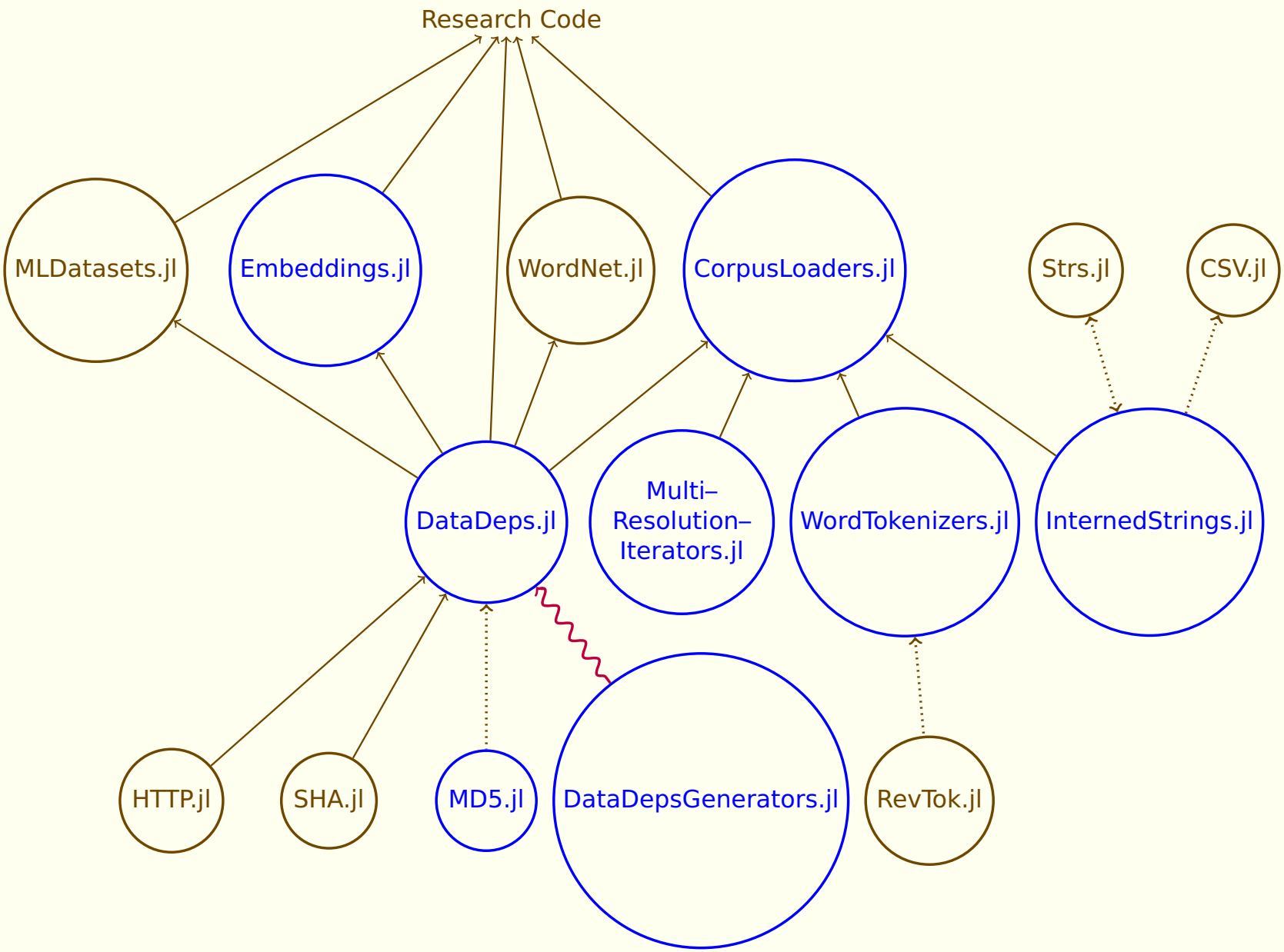
School of Electical, Electronic and Computer Engineering
The University of Western Australia

THE UNIVERSITY OF WESTERN AUSTRALIA

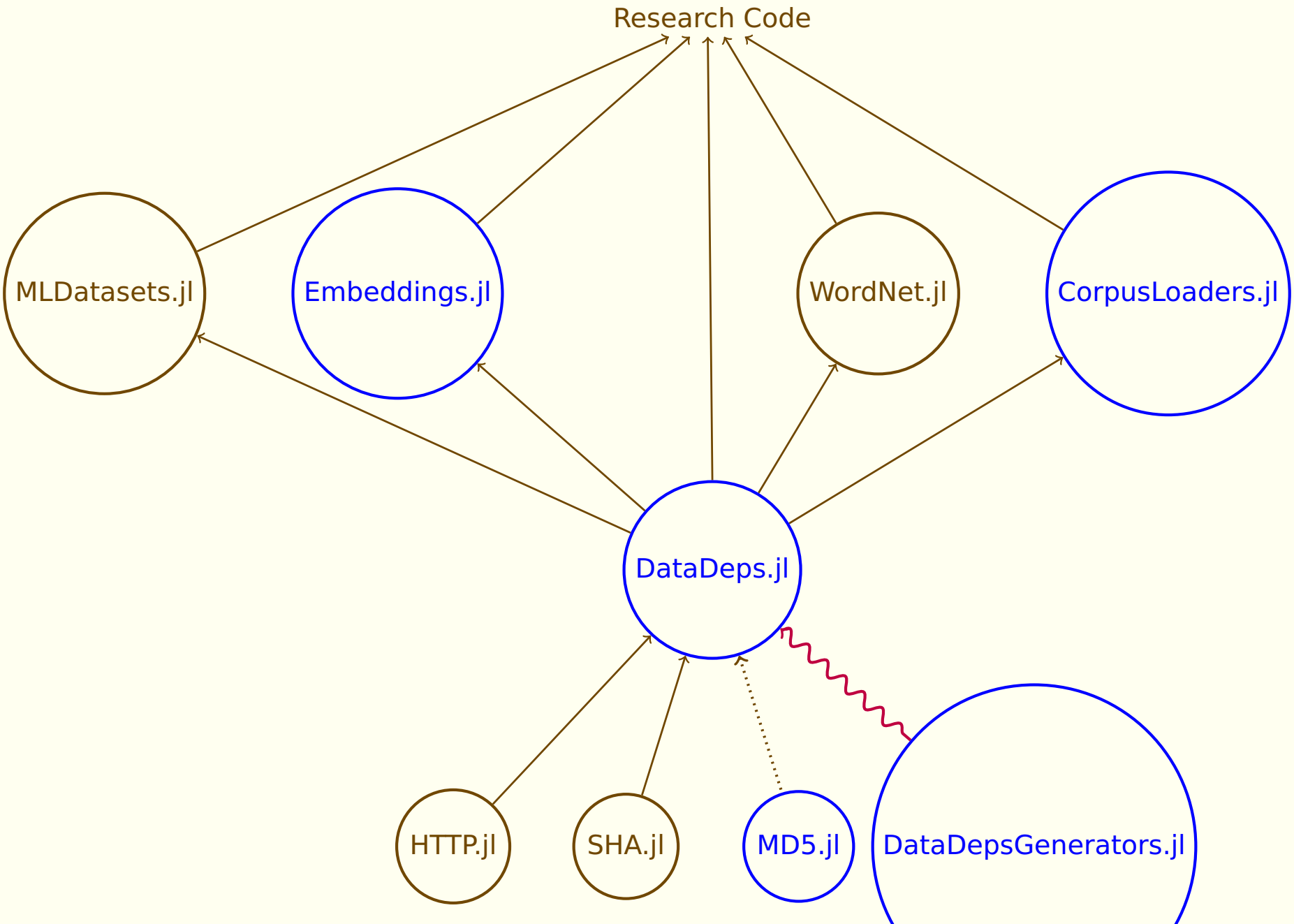
Big Picture


Research Code
MLDatasets.jl
Embeddings.jl
WordNet.jl
CorpusLoaders.jl
Strs.jl
CSV.jl
DataDeps.jl
Multi–
Resolution–
Iterators.jl
WordTokenizers.jl
InternedStrings.jl
HTTP.jl
SHA.jl
MD5.jl
DataDepsGenerators.jl
RevTok.jl
GitHub
UCI ML Repository
CRAN
DataCite
DataDryad
DataOne
538
BuzzFeedNews
Open.Canada.ca
Data.gov.au
Data.gov
European Data Portal
Zenodo
FigShare
ArcticDataCenter
TERN
KnowledgeNetworkforBiocomplexity

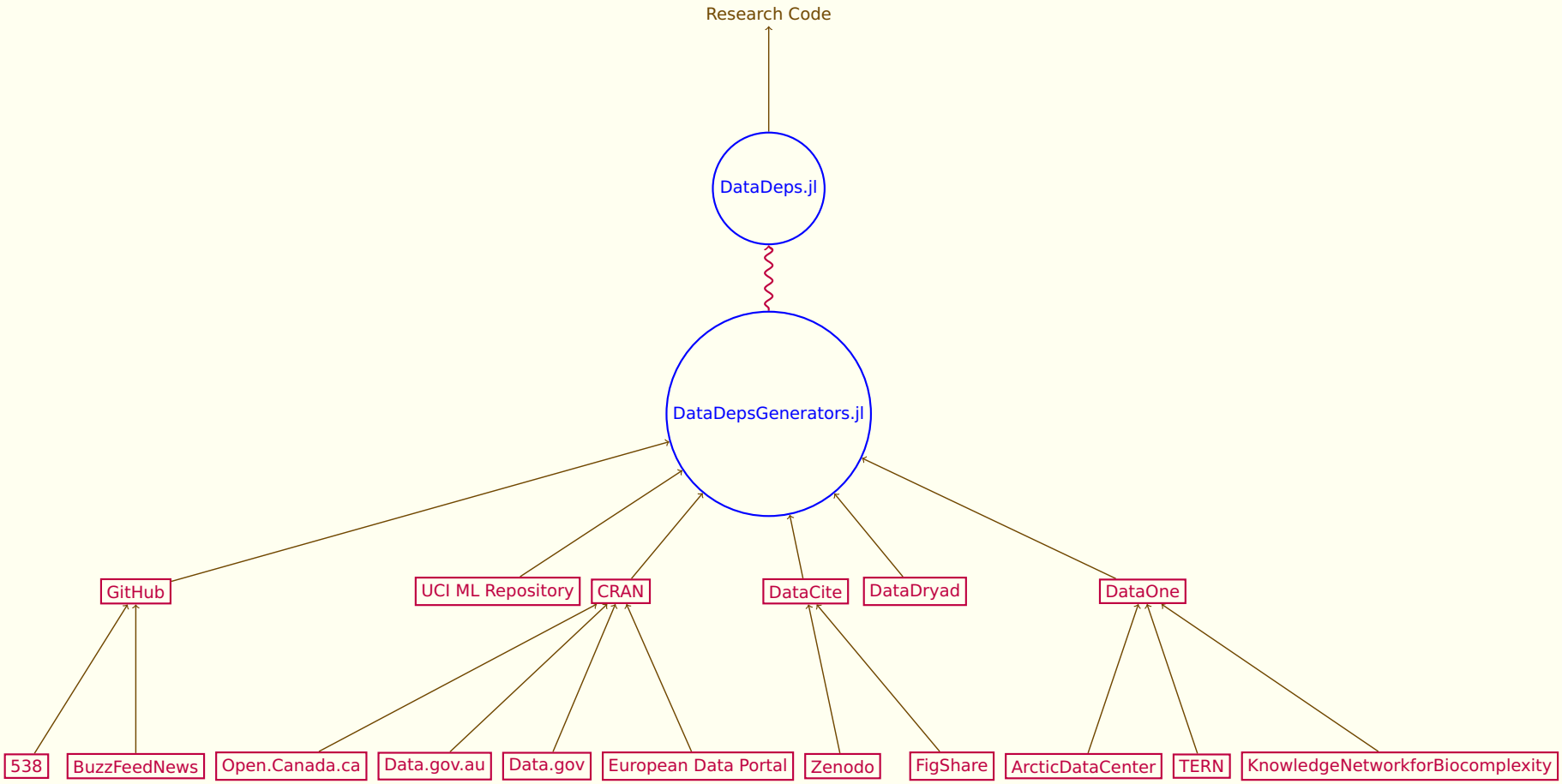
Julia Packages



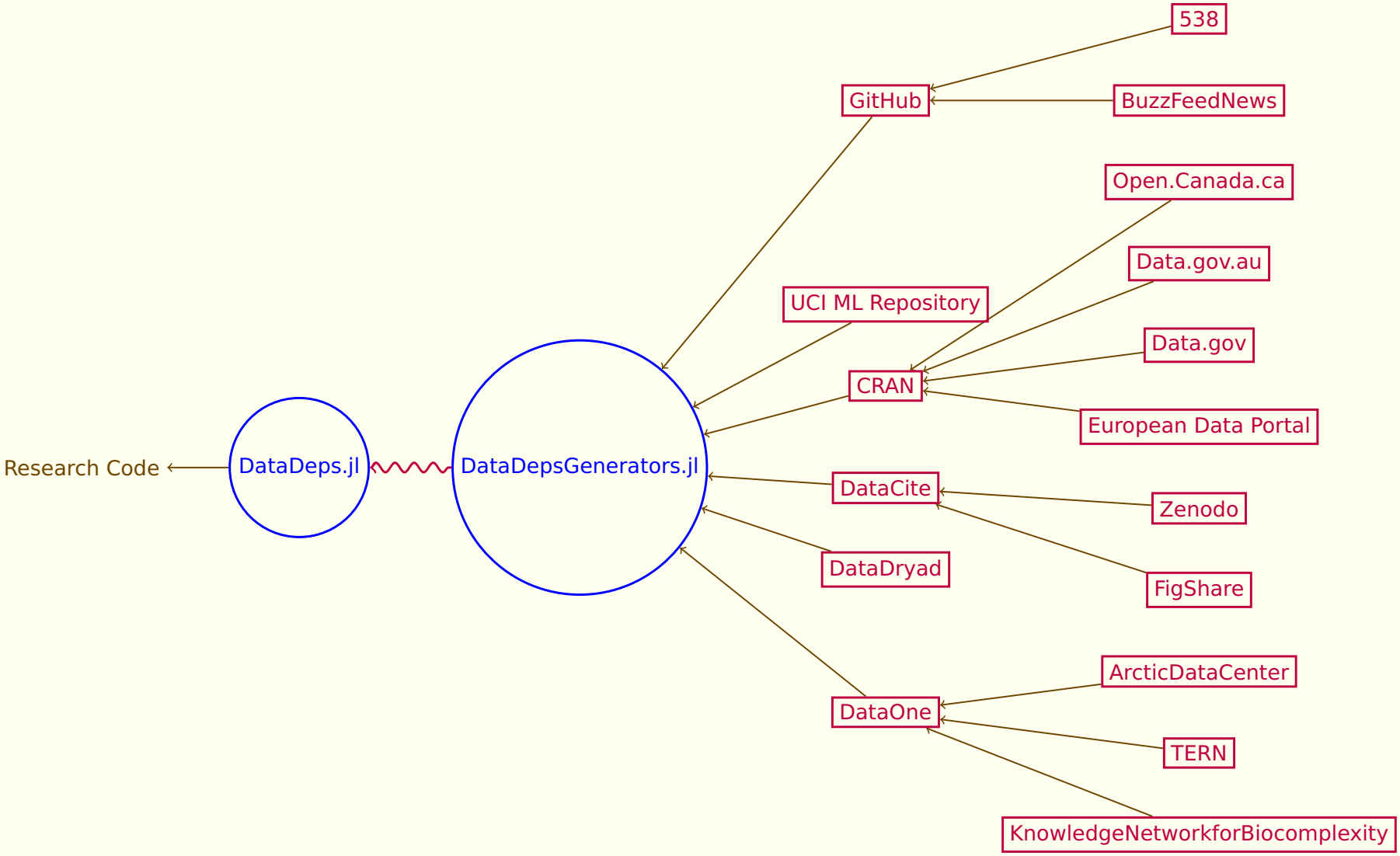
DataDeps.jl


Research Code
MLDatasets.jl
Embeddings.jl
WordNet.jl
CorpusLoaders.jl
DataDeps.jl
HTTP.jl
SHA.jl
MD5.jl
DataDepsGenerators.jl

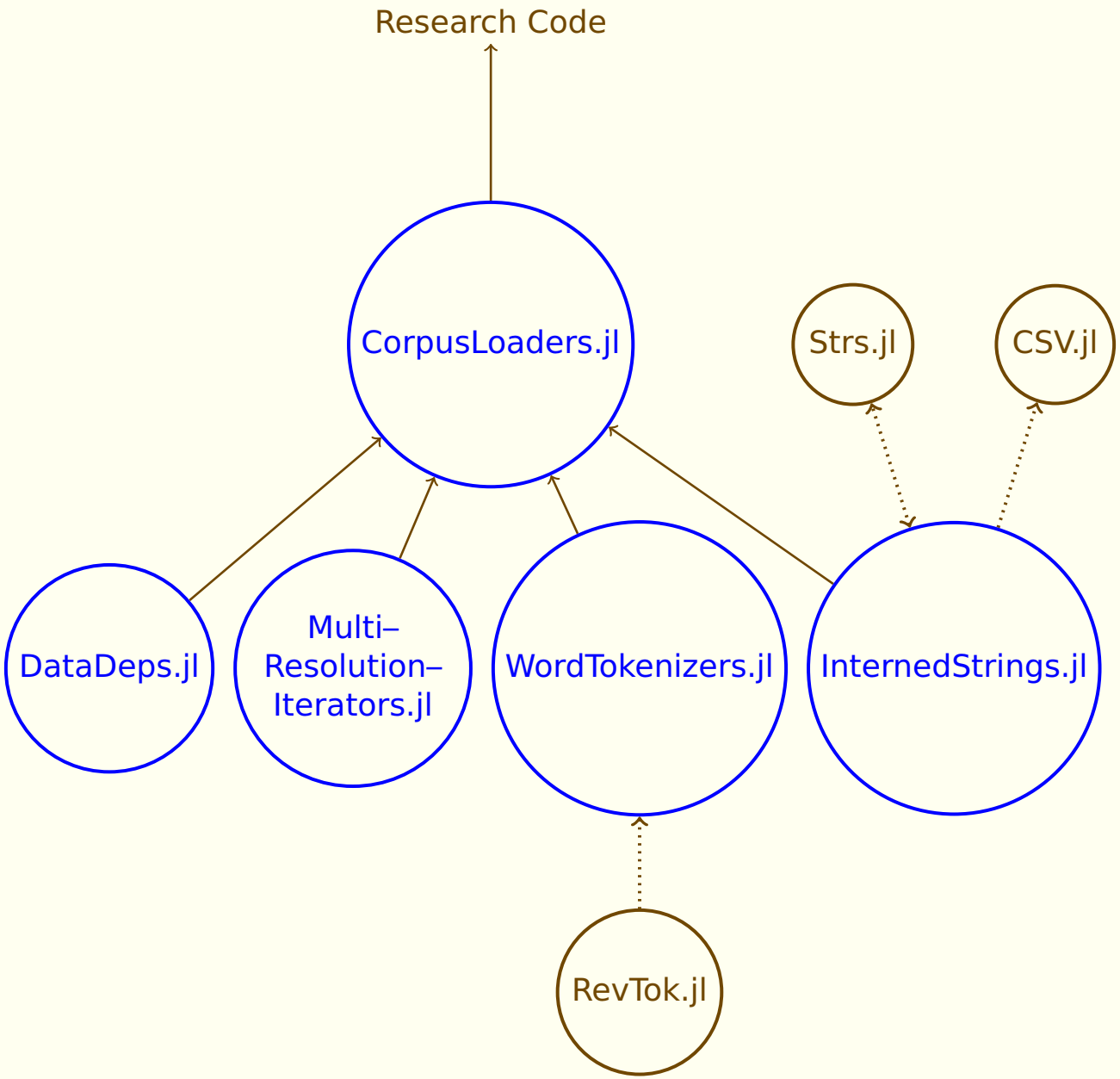
DataDepsGenerators.jl



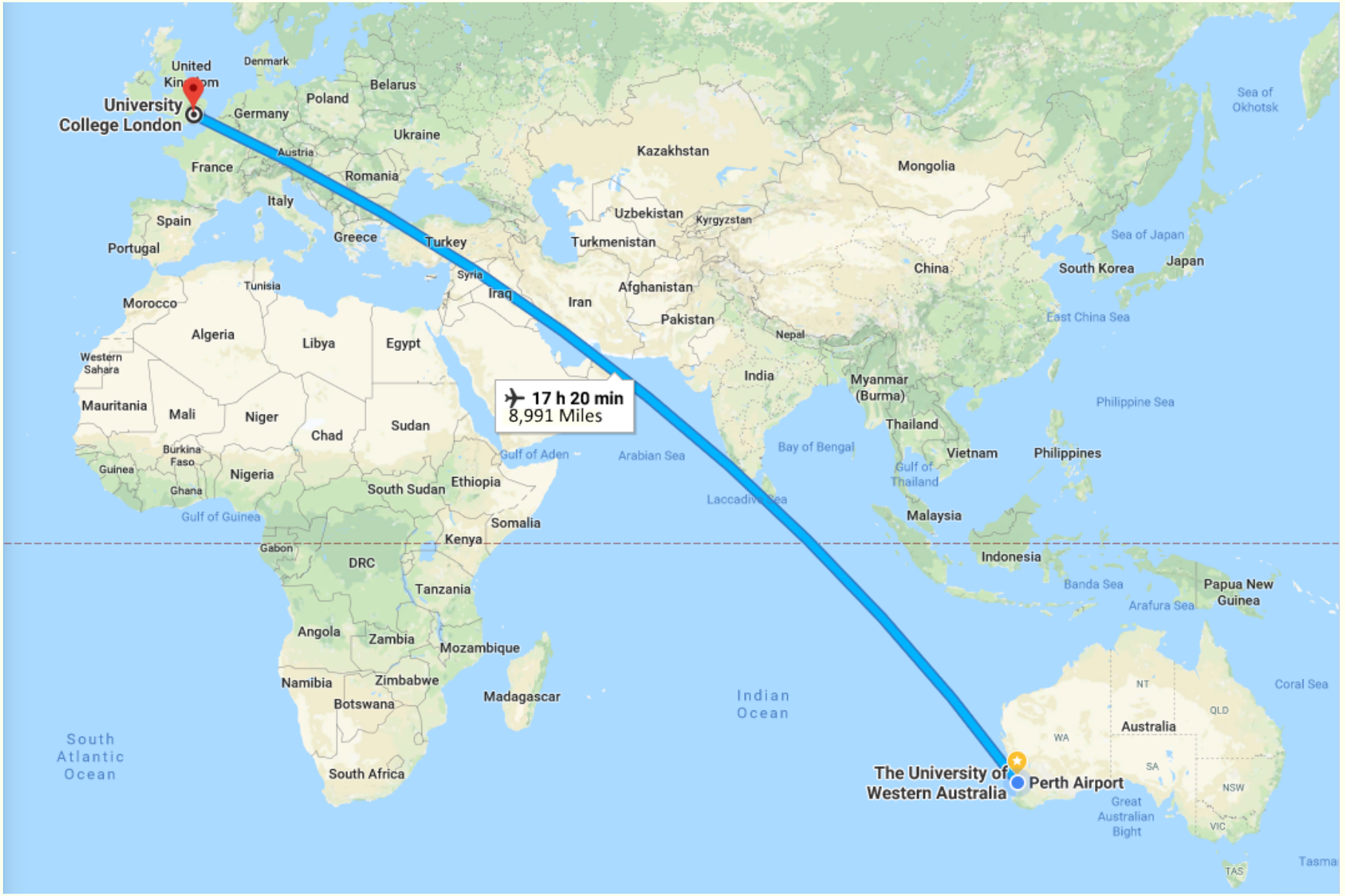
DataDepsGenerators.jl



CorpusLoaders.jl

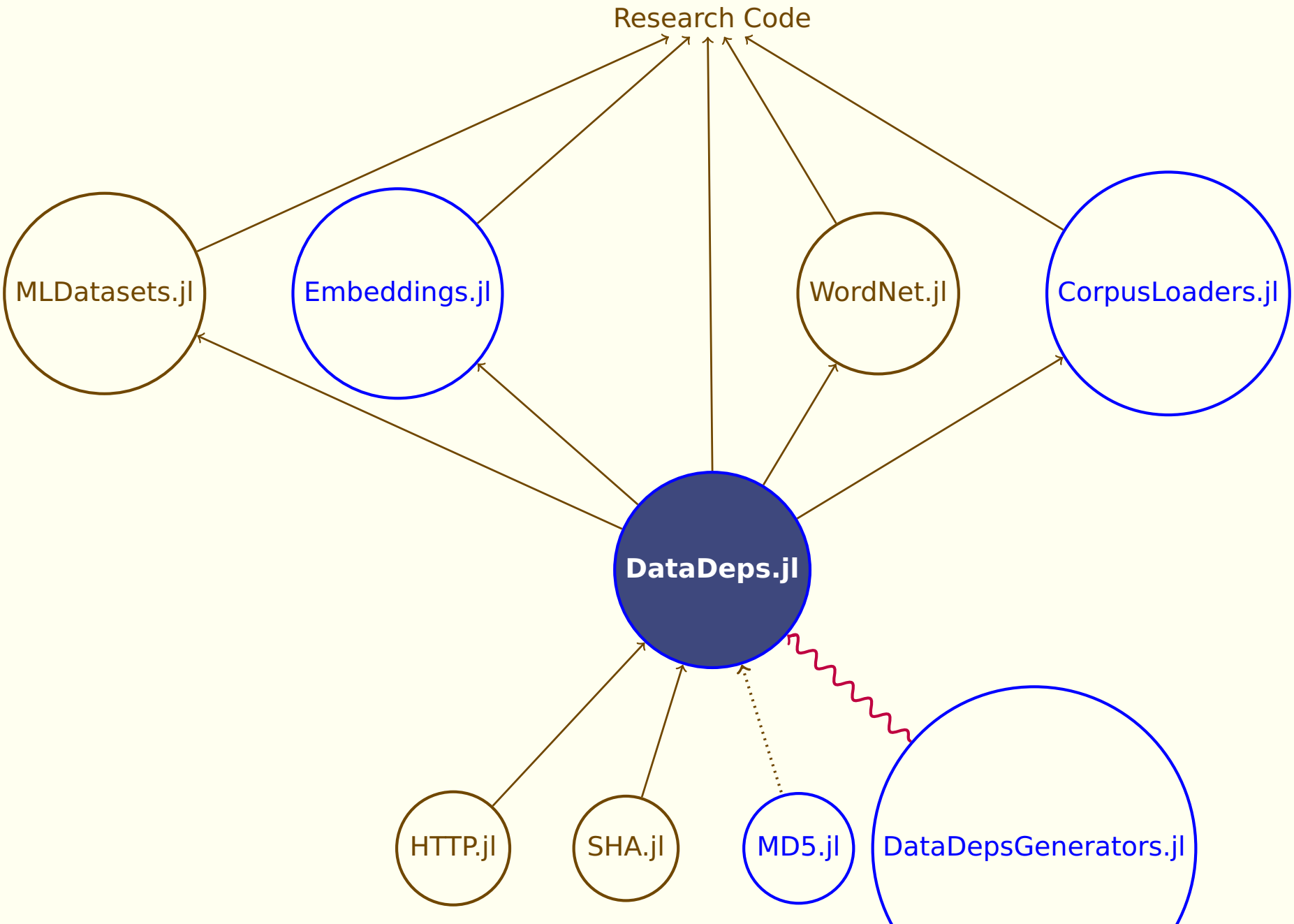


Australia, it is quiet far away



Australia, really it is quiet far away

DataDeps.jl


Research Code
MLDatasets.jl
Embeddings.jl
WordNet.jl
CorpusLoaders.jl
DataDeps.jl
HTTP.jl
SHA.jl
MD5.jl
DataDepsGenerators.jl

3 Key Questions about Data: #1 Storage Location

- Where do I put it? Should it be on the local disk (small) or the network file-store (slow)?
- If I move it, am I going to have to reconfigure things?
- What if I put it all in the git repo?

3 Key **Answers** about Data: #1 Storage Location

- Where do I put it? Should it be on the local disk (small) or the network file-store (slow)?
  - Anywhere on the DataDeps Load Path will work
- If I move it, am I going to have to reconfigure things?
  - Not if you are using a datadep path
- What if I put it all in the git repo?
  - git does not like large binary files.
    50KB `.csv` is ok; 50MB `.jld` is not.

3 Key Questions about Data: #2 Redistribution

- I don’t own this data
- am I allowed to redistribute it?
- How will I give credit, and ensure the users know who the original creator was?

3 Key **Answers** about Data: #2 Redistribution

- I don’t own this data
  - but it is publicly accessible, right?
- am I allowed to redistribute it?
  - IANAL, but linking should be fine.
- How will I give credit, and ensure the users know who the original creator was?
  - Tell them with a prompt when it is downloaded

3 Key Questions about Data: #3 Replication

- How can I be sure that someone running my code has the same data?
- What if they download the wrong data, or extract it incorrectly?
- What if it gets corrupted or has been modified and I am unaware?

3 Key **Answers** about Data: #3 Replication

- How can I be sure that someone running my code has the same data?
  - Make automatic data fetching part of the code
- What if they download the wrong data, or extract it incorrectly?
  - They can’t make mistakes if it is automated
- What if it gets corrupted or has been modified and I am unaware?
  - Filehashs, SHA256, MD5 etc.

DataDeps.jl is best for Data with the following properties.

DataDeps.jl
Static
Public
File-Based
Any Format
Any Size

DataDeps.jl is best for Data with the following properties.

DataDeps.jl	Git
Static	Dynamic
Public	Public or Private
File-Based	File-Based
Any Format	Plain Text Formats
Any Size	<50MB

`ManualDataDep`, using DataDeps.jl with git and other uses

- A manual datadep registration consists just of a name and a message.
- It lets you use `datadep"Paths"` for locating data without any automation
  - If locating the file fails, it prompts the user with the message to install it.

`ManualDataDep`, using DataDeps.jl with git and other uses

- A manual datadep registration consists just of a name and a message.
- It lets you use `datadep"Paths"` for locating data without any automation
  - If locating the file fails, it prompts the user with the message to install it.
- Note that for every package the directory `<PKG>/deps/data` is on the datadeps load path
  - So for files that a good fit for git you can include them in the Repo.

`ManualDataDep`, using DataDeps.jl with git and other uses

- A manual datadep registration consists just of a name and a message.
- It lets you use `datadep"Paths"` for locating data without any automation
  - If locating the file fails, it prompts the user with the message to install it.
- Note that for every package the directory `<PKG>/deps/data` is on the datadeps load path
  - So for files that a good fit for git you can include them in the Repo.
- Note also this lets you work with dynamic data and private data
  - as the syncing mechanism is external to DataDeps.jl

Theoretical, handling private data, with specific `fetch_method` method

```
using HTTP, OAUTH2
baseurl="https://.../files"; authurl="https://.../auth";
appid="..."; appsecret="..."
function authed_download(fileid, local)
    authtok = getauthtok(authurl
        ENV["GD_ID"], appid, appsecret)
    url = joinpath(baseurl, fileid
    HTTP.download(url, local, ["Authorization"=>authtok])
end

register(DataDep("Secret Science Data",
"The data from my google drive of sciency stuff"
"12fileidencodedinbase64whywecanusepaths21ab", # fileid
"d41d8cd98f00b204e9800998ecf8427e"; # hash
fetch_method = authed_download
))
```

Vabdewakke's 6 Degree's of Replicability

1. The results cannot seem to be reproduced.
2. The results could be reproduced by, requiring extreme effort.
3. The results can be reproduced, requiring considerable effort.
4. The results can be easily reproduced with at most **15 minutes** of user effort, requiring some proprietary source packages (MATLAB, etc.).
5. The results can be easily reproduced with at most **15 min** of user effort, requiring only standard, freely available tools (C compiler, etc.).

Vandewalle, Kovacevic, and Vetterli (2009),
"Reproducible research in signal processing"

What happens when I try and reproduce someone's research code?

1min	Find the website from the paper, and download the code
2min	Read enough of the README to get rough bearings
1min	Find out where to get the data from and download the data
2min	Try and remember how to use `tar -xzfvalphabetsoup` etc.
2min	Workout how to tell script where data is
2min	Setup any software dependencies etc.
3min	Run the code and make sure it isn't crashing etc.
2min	Interpret the output

You can't trust hardcoded paths; but they are nice to work with.

- Ideally we'ld just use hard-coded, absolute paths
- Absolute paths work with all applications
- Hard-coding the paths in code means less typing
- But they break if anything is moved.
- Making the path be passed in as an argument to the script solves this
  - but now user has to be typing it in to run it.
  - So harder to use.
  - You could include a bash-script that invokes it with the path, but now you're just hard coding it somewhere else

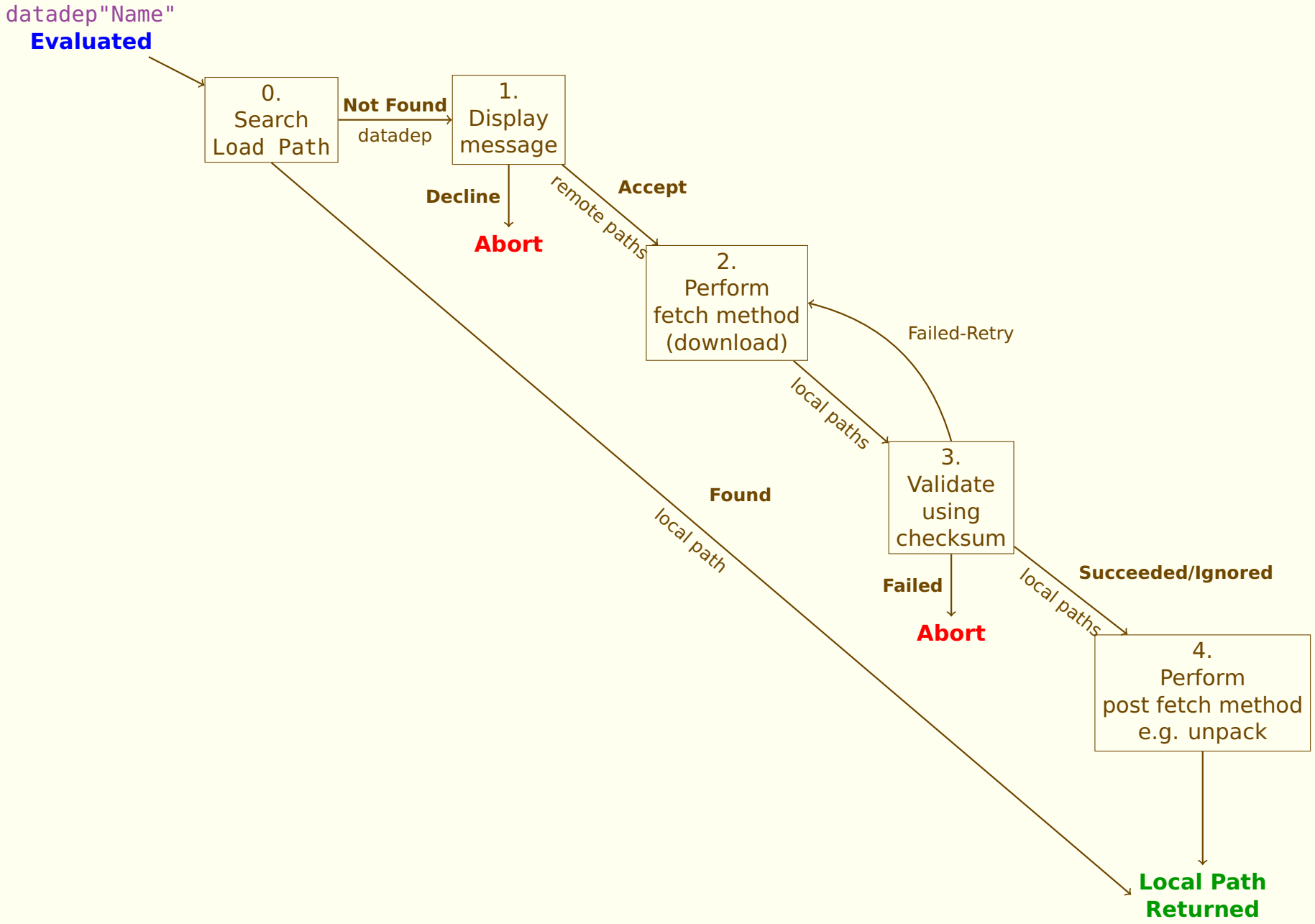
You could making the path be passed in as an argument. But...

- Now user has to be typing it in to run it.
- So it is harder to use.
- You could include a bash-script that invokes it with the path, but now you’re just hard-coding it somewhere else

`datadep"Census 2018/populations.csv"`
A path you can trust

- Always resolves to an absolute path to that file
- Even if that means it has to download it first
- But before resorting to downloading checks a large number of places
  - `<PKG>/deps/data`,
  - `~/.julia/datadeps`,
  - `/usr/share/datadeps`, etc.
- You know that if you use a datadep path it will resolve to a file that exists.

What happens when you uses a datadep?



Current Usages of DataDeps.jl

MLDatasets.jl

- Provides easy access to a bunch of ML datasets
- `xs, ys = MNIST.traindata()`
- Gives you regular julia arrays

CorpusLoaders.jl

- Provides easy access to linguistic corpora
- `corpus_gen = load(WikiCorpus())`
- gives you a multi-resolution iterator

Current Usages of DataDeps.jl

Embeddings.jl

- Provides access to hundreds of pretrained word embedding models.
- `load_embeddings(FastText_Text{:fr})`
- gives you a table of French word embeddings.

WordNet.jl

- Look up lexical relations and definitions.
- `lemma = db['a', "glad"]`
- `antonyms(db, synsets(db, lemma)[1])`

DataDep Registration Block

```
register(DataDep("DataDepName",
"""
Free Text Field Displayed to user before download.
Use to give credit, and tell people about licensing.
Or other messages.
""",
"Download URL",
"file hash (will be printed if not provided)";
post_fetch_method = function to run on downloaded files
))
```

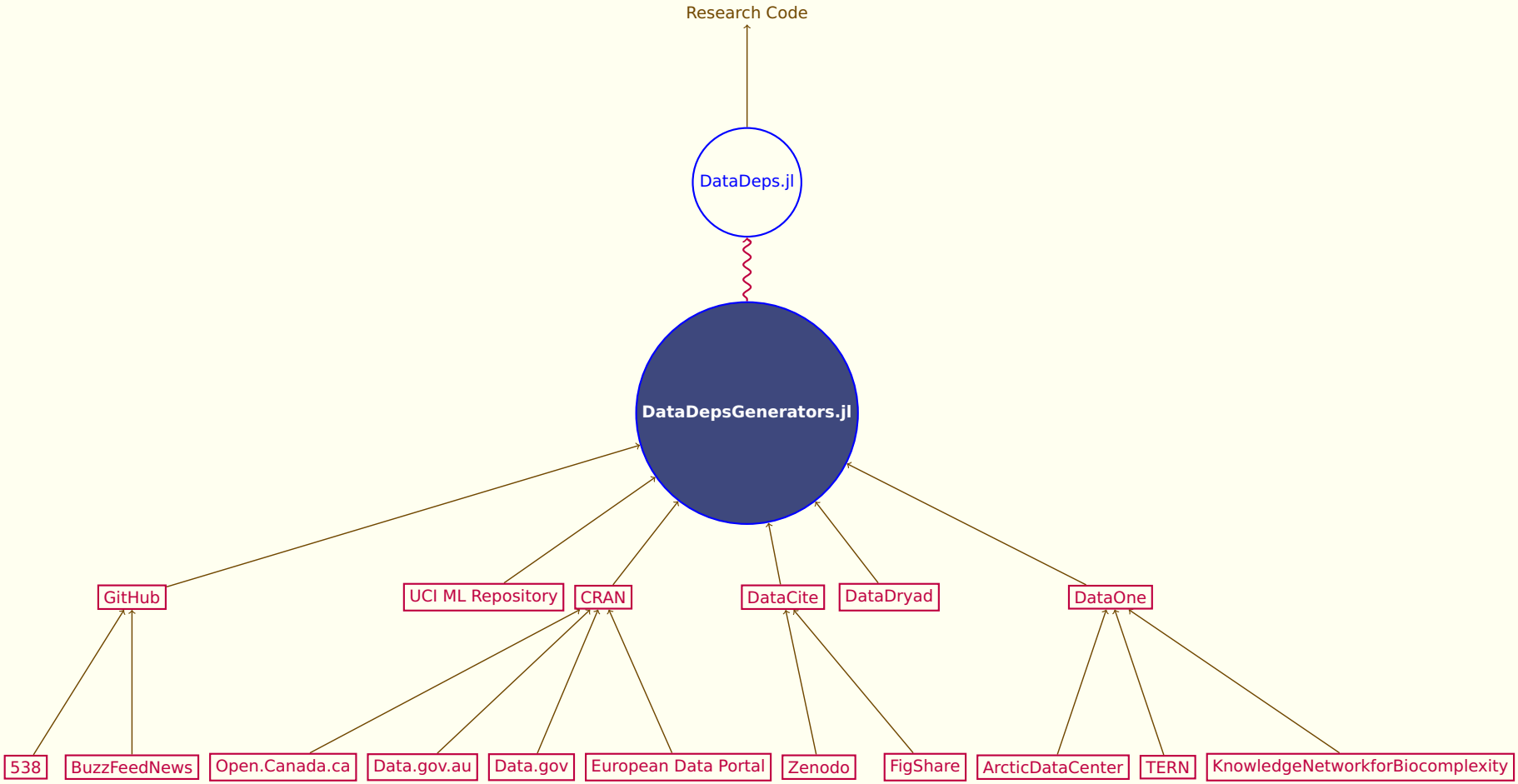
Registration Block Example

```
register(DataDep("WordNet 3.0",
"""
Dataset:  WordNet 3.0
Website:  https://wordnet.princeton.edu/wordnet

George A. Miller (1995).
WordNet:  A Lexical Database for English.
Communications of the ACM Vol.  38, No.  11:  39-41.

License:
This software and database is being provided to you,
the LICENSEE, by Princeton University under
the following license...
""",
"http://wordnetcode.princeton.edu/3.0/WNdb-3.0.tar.gz",
"658b1ba191f5f98c2e9bae3e25...";
post_fetch_method = unpack
))
```

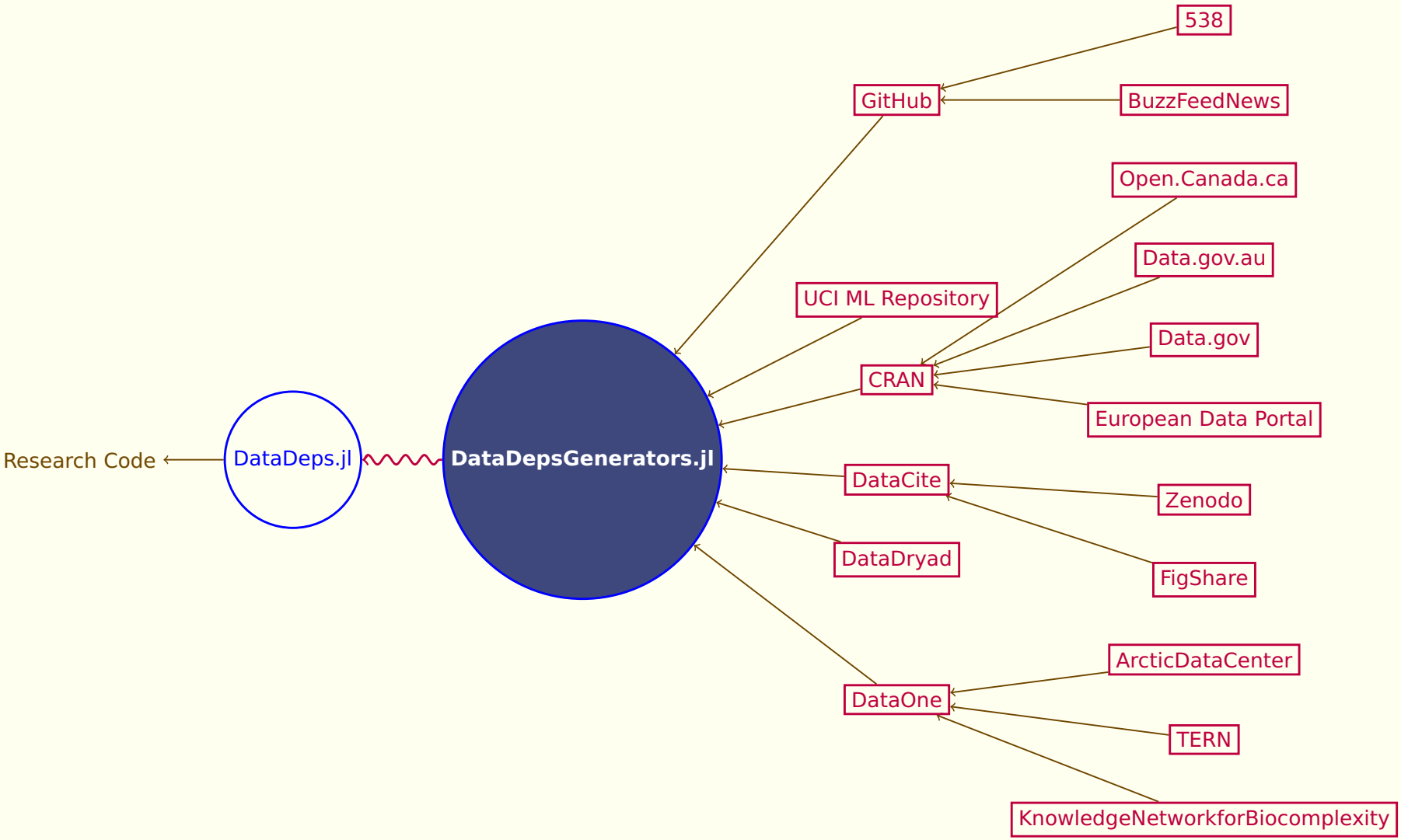
Registration Block: Recursive Example

```
using MD5

register(DataDep("DataDepNameRec",
"""
Warning these files are all together 39.8GB
""",
["http://example.com/readme.txt",
    ["http://example.com/data1.zip",
     "http://example.com/data2.tar.gz",
    ]
],
(md5, "d41d8cd98f00b204e9800998ecf8427e")
post_fetch_method = [identity, unpack]
))
```

DataDepsGenerators.jl



DataDepsGenerators.jl

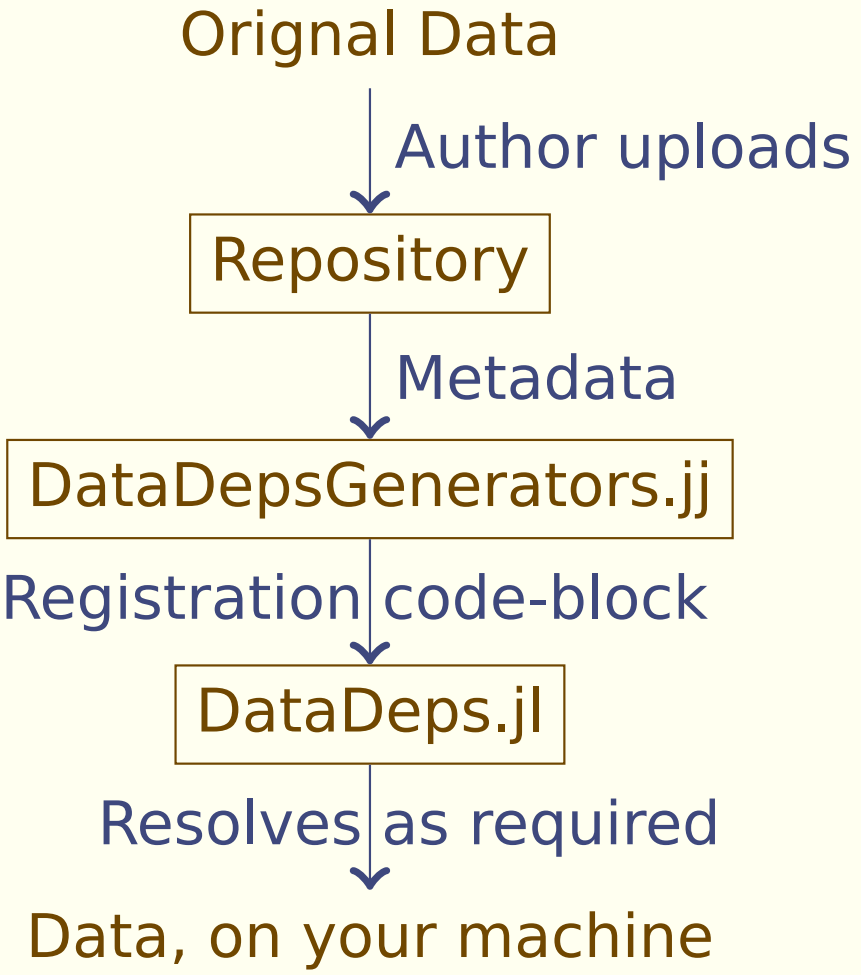


Developers still have to write registration blocks

- DataDeps.jl shifts the work from manually to automatic
- But it still has to be done at once.
- Writing a registration block normally means copy-and-pasting from a website.
  - Even copy pasting a dozen URLs is annoying
  - Enough information for data providence needs more

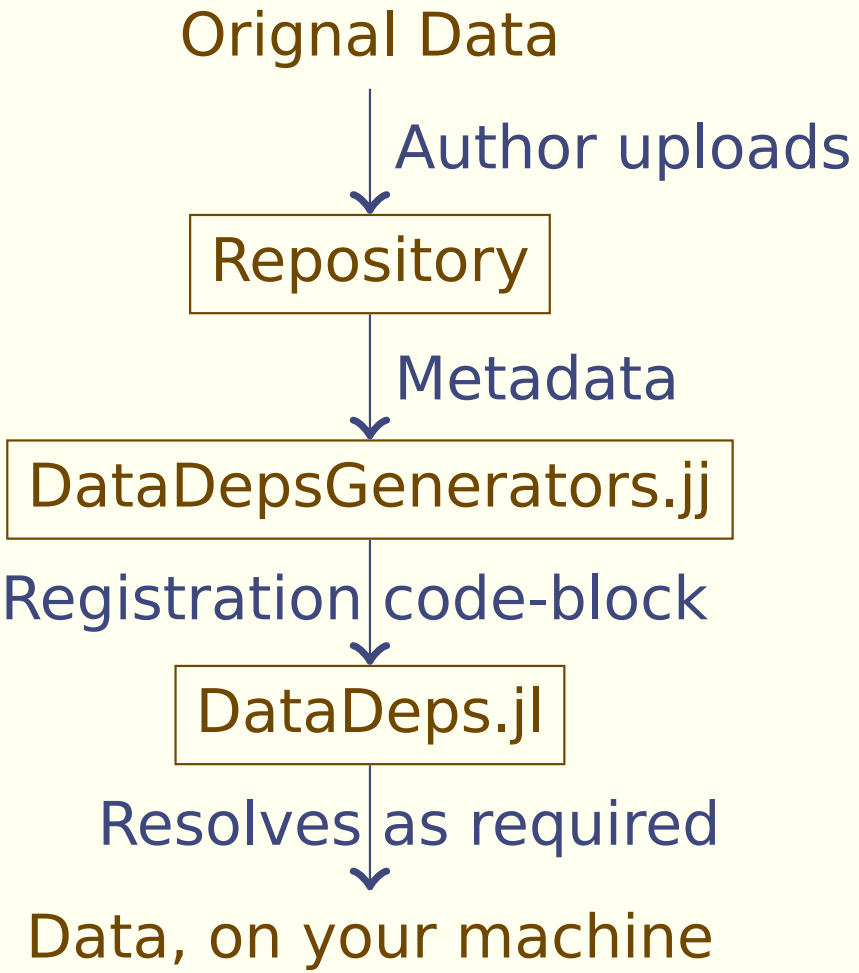
For published data this information is available from some API

- Title
- Author Name
- Publication Date
- Licensing info
- Description
- Citation for the linked journal paper
- Download URLs
- File Hashes



DataDepsGenerators creates static code

- This avoids propagating DataDepsGenerator's many heavy dependencies.
- Avoids issue unstability of repo APIs/websites



Many datasets have multiple possible APIs

For example something that points at Figshare

```
generate("https://doi.org/10.23640/07243.5525476.v1")
```

- It includes a DOI
  - DataCite API
  - JSON-LD DOI Content Negotiation service
- It is a URL
  - Embedded JSON-LD element
  - FigShare API
- Many others will be tried and will error out

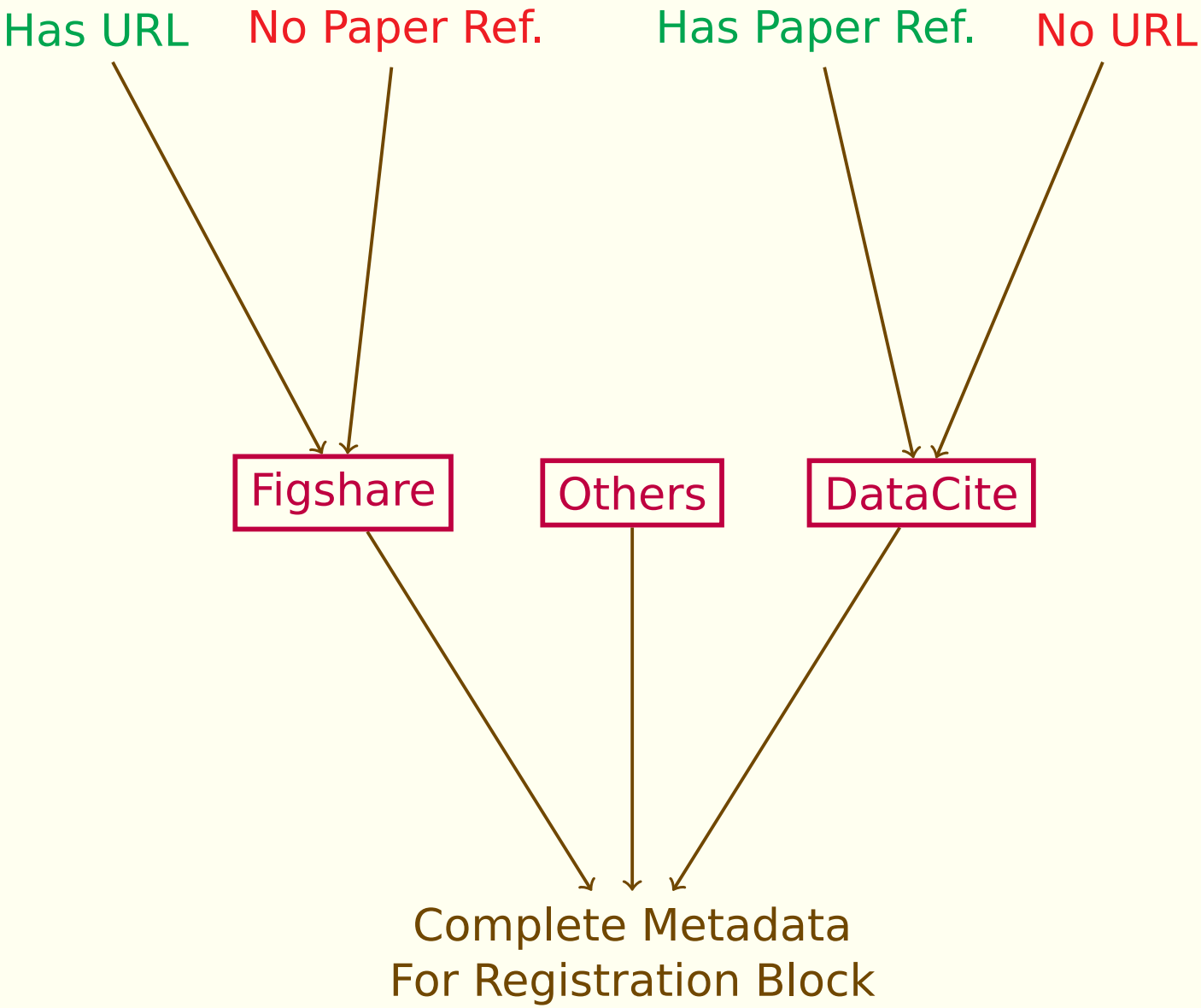
Many datasets have multiple possible APIs

For example something points at DataDryad

```
generate(
"http://datadryad.org/resource/doi:10.5061/dryad.gj651")
```

- DataCite API
- JSON-LD DOI Content Negotiation service
- Embedded JSON-LD element
- DataDryad Webscraper
- DataOneV1 API
- Many others will be tried and will error out

DataDepsGenerators combines all the Metadata Sources



GitHub

- 85×10^6 repositories
  - Some of them are data, (most are not)
  - Not a great place for data, but commonly used
- BuzzFeedNews:
  - 1 Repo per dataset
- fivethirtyeight:
  - 1 shared Repo with a folder for each dataset
- We generate URLs pointing at `cdn.rawgit.com`
  - This is backed by StackPath CDN
  - It is generated to point at the latest commit at generation time

DataOne

- Data Observation Network for Earth
  - ~40 Earth and Environment Science repositories
  - 1.2×10^6 Data Files
  - Seems to have iffy metadata, varying between nodes.

Others

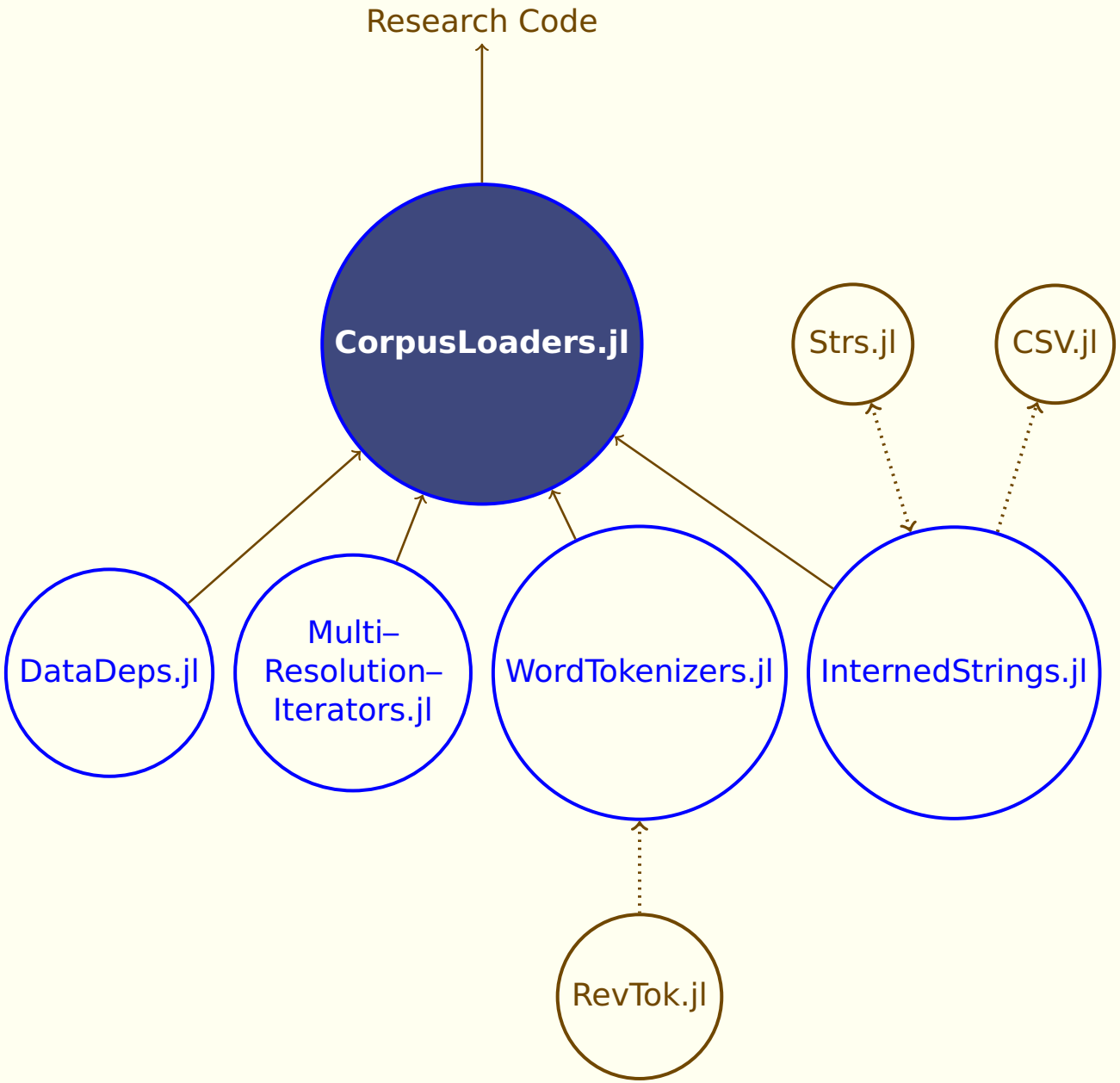
- DataDryad
  - Ecological research data
  - 7.1×10^4 Data Files
  - 2.2×10^4 Data Packages
- Figshare
  - Mostly Figures and Datasets
  - 8×10^5 Files
- UCI ML Repository
  - 437 Datasets
  - Very commonly used in benchmarking basic ML
  - Awful website, zero API

DataCite

- 11×10^6 DOIs issued
  - Mostly to datasets, though they count also figures as data
  - If you have a DOI and it points at data, it is probably from DataCite
- Problem is all their metadata is missing download URLs
  - So user has to add them manually after
- This is effectively the final fallback for anything with a DOI.

We are now going to take a very short break

CorpusLoaders.jl

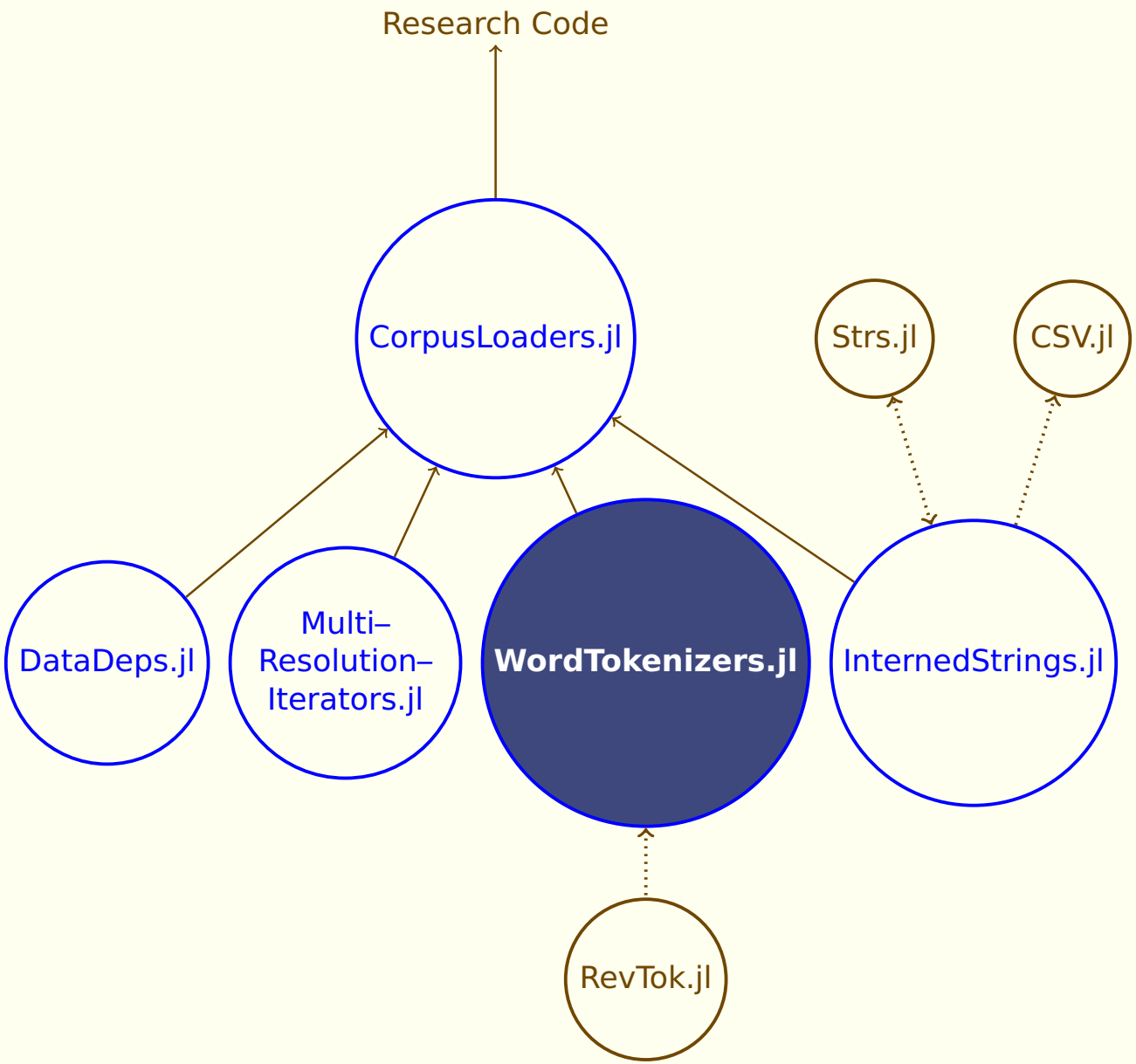


WordTokenizers.jl

- Not much to say.
- Honestly not a lot in it itself.
  - Just a couple of parsers for a few formats
- Mostly interesting as a vessel for driving its dependencies.

- Notable differences from MLDatasets.jl
  - Everything is lazy-loaded from disk via `Channels`.
  - Everything is Multi-Resolution

CorpusLoaders.jl



WordTokenizers.jl

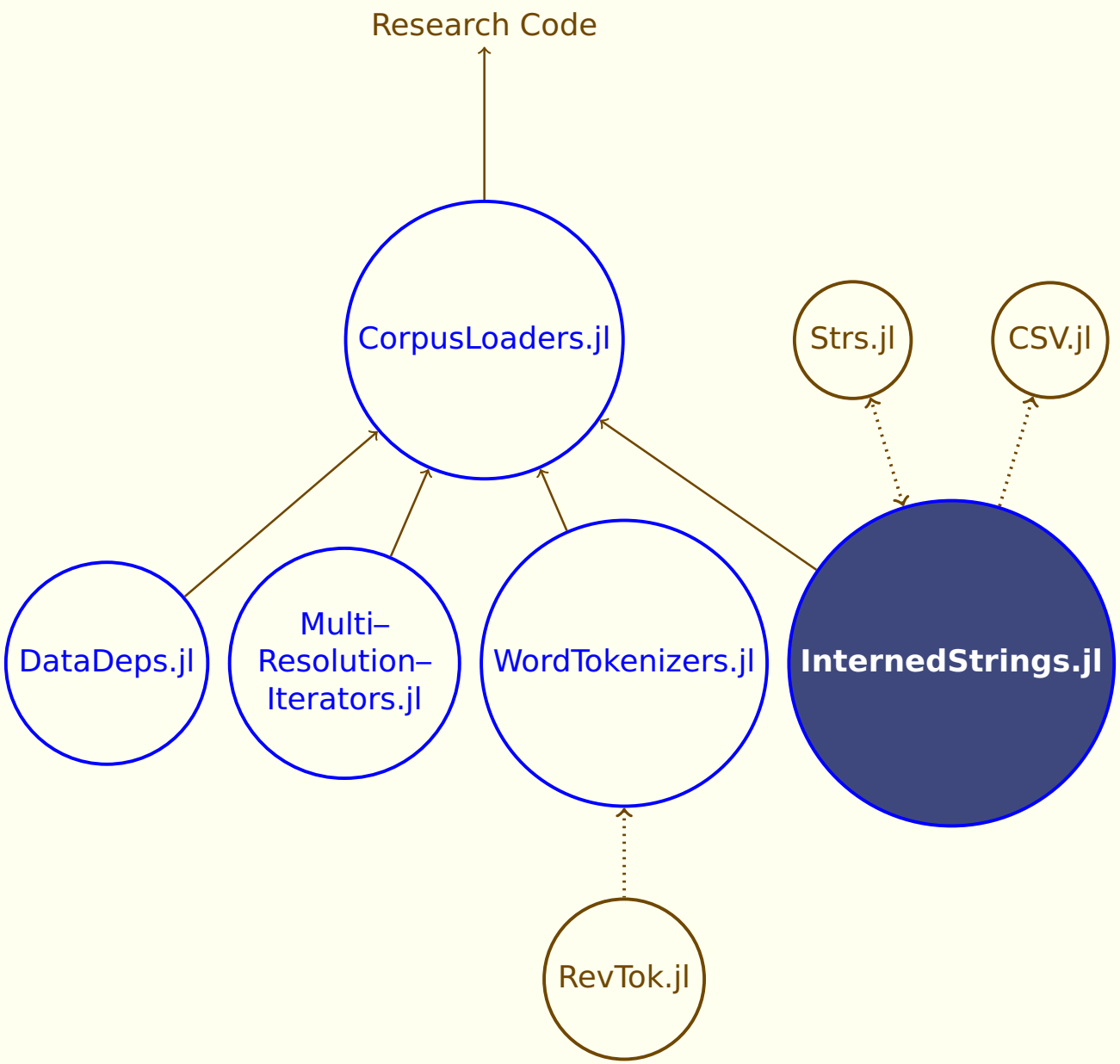
Configurable `tokenizer` and `sentence_segmenter`.
Abuses `eval` and `#265` so that you can change the tokenizer being used globally.
Also compatible with externally defined tokenizers like RevTok.jl.

Nabbed the original Penn Tokenizer sed-script.
Wrote some code that converts basic sed language into julia AST.
Ported some of NLTK's tokenizers into sed.

Rule-based sentence splitter based on Sampo Pyysalo & Yoshimasa Tsuruoka's perl script.

Regex is just really good at working with English.

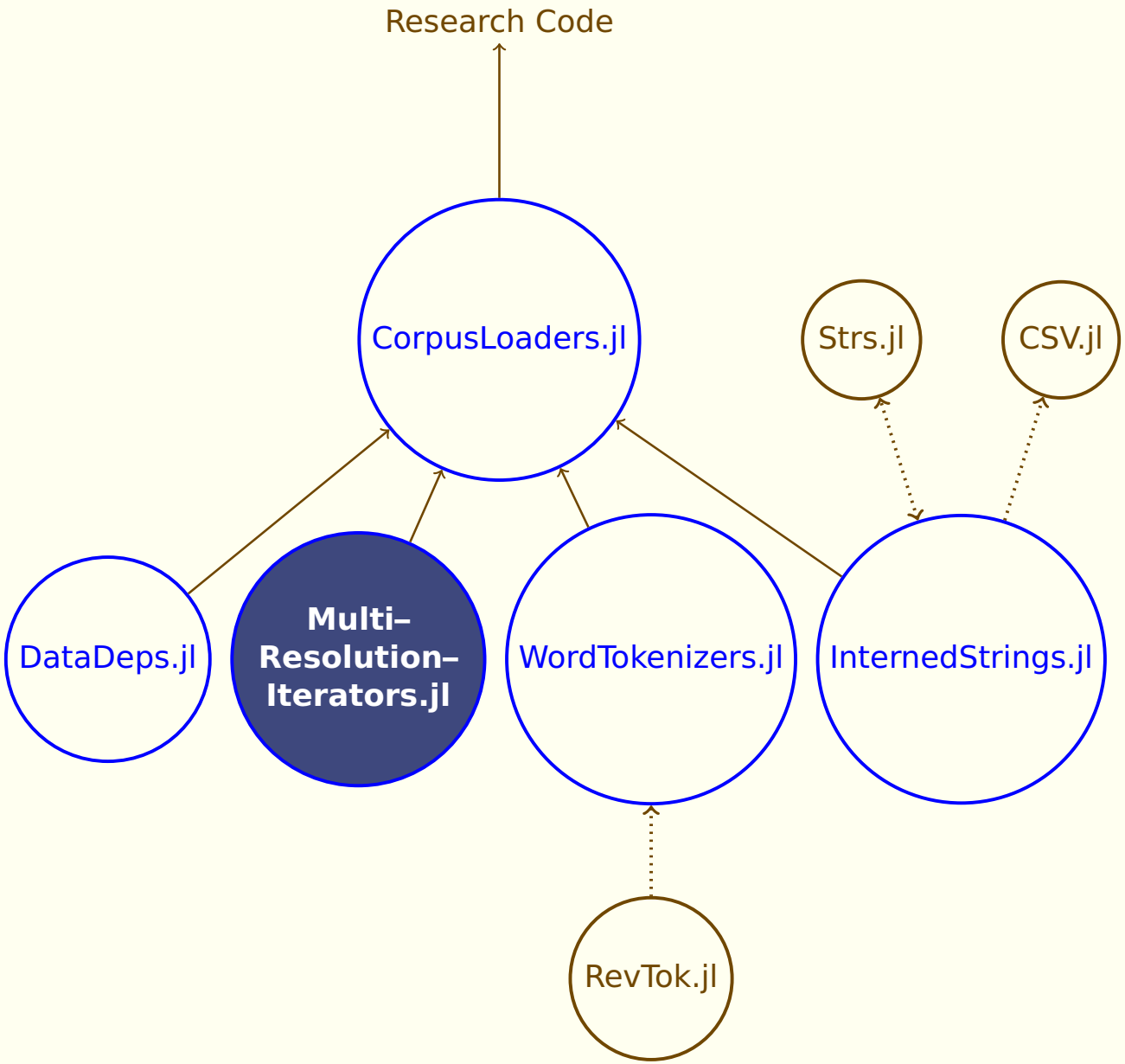
CorpusLoaders.jl



InternedStrings.jl: All these duplicate strings are using all my memory

- Strings are immutable, so we only need one copy of each
- We can maintain a pool of `WeakRef`s to each string allocated.
- `str = intern(str)`
  - Add the `str` to the pool if not already
  - replace `str` with an element of the pool
- Because the pool only has `WeakRef`s strings can still be garbage collected.

CorpusLoaders.jl



MultiResolutionIterators.jl: The structure of a Corpus

- Corpus
- made up of: Documents
- made up of: Paragraphs
- made up of: Sentences
- made up of: Words or Tokens
- made up of: Letters or Characters

MultiResolutionIterators.jl: Not everyone wants every level of structure

Full corpus structure is
Documents ►Paragraphs ►Sentences ►Words ►Letters

A Corpus Linguist may want
a stream of: Sentences ►Words

An Information Retrieval researcher may want
a stream of: Documents ►Words

A char-RNN language modeller might just want
a stream of Letters

MultiResolutionIterators.jl: example

```
julia> animal_info = [
[   ["Turtles", "are", "reptiles", "."],
    ["They", "have", "shells", "."],
    ["They", "live", "in", "the", "water", "."]],
[   ["Cats", "are", "mammals", "."],
    ["They", "live", "on", "the", "internet", "."]]
]
2-element Array{Array{Array{String,1},1},1}

julia>  struct AnimalTextIndexer end;
julia>  levelname_map(::AnimalTextIndexer) = [
    :documents=>1,
    :sentences=>2,
    :words=>3, :tokens=>3,
    :characters=>4, :letters=>4
]
```

Information Retrieval wants Documents ►Words

```
julia> flatten_levels(animal_info, 2)
```

~ Or ~

```
julia> flatten_levels(animal_info,
           lvls(indexer, :sentences))
```

~ Or ~

```
julia> flatten_levels(animal_info,
           (!lvls)(indexer, :documents, :words))

julia> full_consolidate(ans)

2-element Array{Array{String,1},1}:
String["Turtles", "are", ".", ...,  "the", "water", "."]
String["Cats", "are", ..., "the", "internet", "."]
```

Char-RNN Topic Modelling wants Documents ►Words

```
julia> join_levels(animal_info,
            lvls(indexer,Dict(
                :words=>" ",
                :sentences=>" ")
             )
        )

julia> full_consolidate(ans)

2-element Array{String,1}:
"Turtles are reptiles ... They live in the water ."
"Cats are mammals . They live on the internet ."
```

MultiResolutionIterators.jl

- Mix and match operations as needed
- Easy to define any iterator function to work at levels
- Overload `apply` to customize behaviour with particular iterator-types.

E.g. could make `MyVector` eager and type preserving via `apply(f, v::MyVector) = MyVector(collect(f(v)))`

What have we learned?

- Australia: Quiet distant, not near London

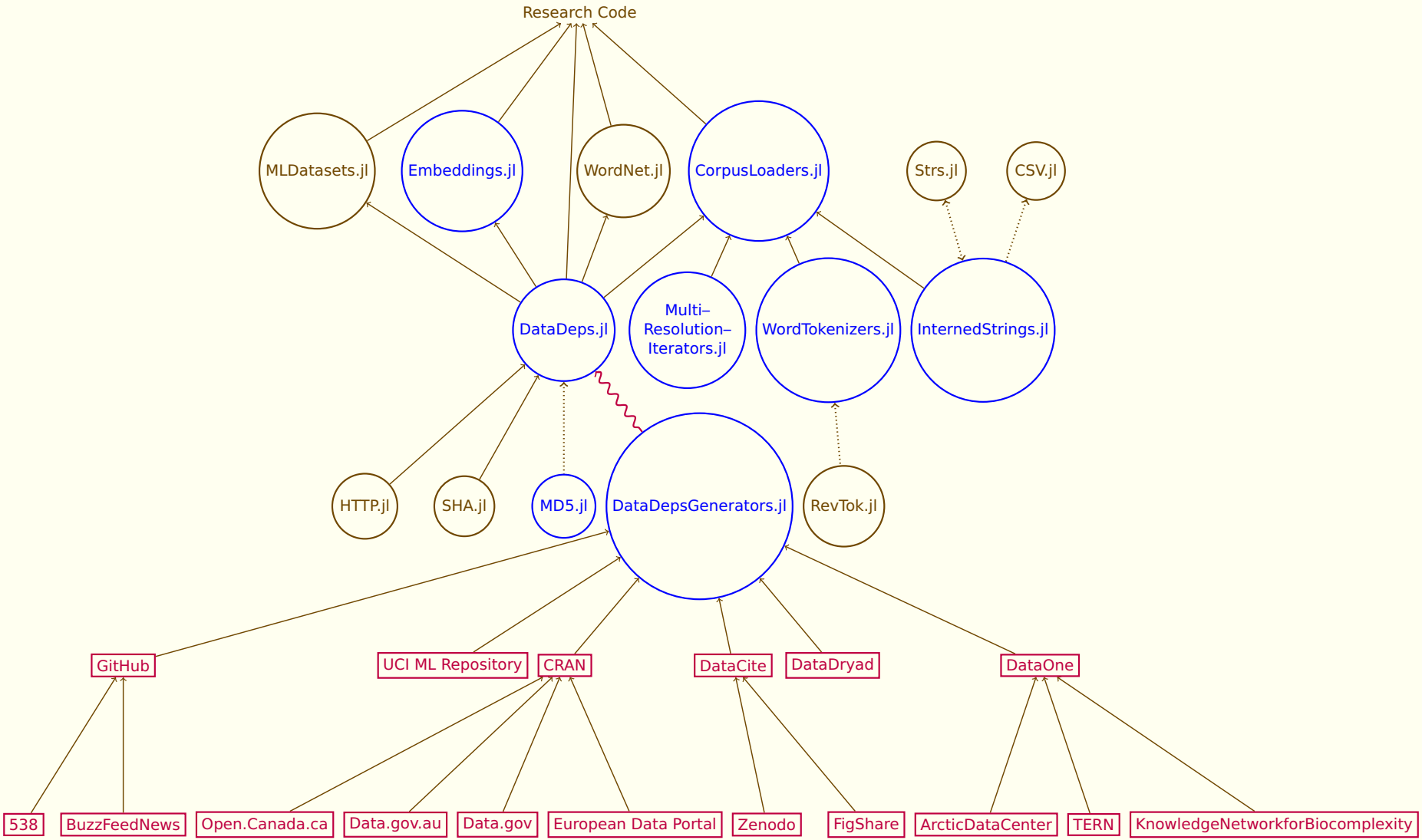
What have we learned?

- Australia: Quiet distant, not near London
- Data is important
- Simplify your data management with DataDeps.jl
- Simplify your DataDeps creation with DataDepsGenerators.jl

What have we learned?

- Australia: Quiet distant, not near London
- Data is important
- Simplify your data management with DataDeps.jl
- Simplify your DataDeps creation with DataDepsGenerators.jl
- Data is useful for NLP
- Packages exist to make working with NLP shaped data easier.

Big Picture


Research Code
MLDatasets.jl
Embeddings.jl
WordNet.jl
CorpusLoaders.jl
Strs.jl
CSV.jl
DataDeps.jl
Multi–
Resolution–
Iterators.jl
WordTokenizers.jl
InternedStrings.jl
HTTP.jl
SHA.jl
MD5.jl
DataDepsGenerators.jl
RevTok.jl
GitHub
UCI ML Repository
CRAN
DataCite
DataDryad
DataOne
538
BuzzFeedNews
Open.Canada.ca
Data.gov.au
Data.gov
European Data Portal
Zenodo
FigShare
ArcticDataCenter
TERN
KnowledgeNetworkforBiocomplexity

Appendix/ Spare Rants

Why DOI's alone are not the answer to broken dataset links:

- When we say DOI's are persistent we mean they never expire.
  - Unlike web URL, you do not have to pay to renew.
  - It won't be sold to someone else. It will always refer to your object.
- DOI's point to landing pages, not data/paper downloads
  - DOI's have attached metadata, but not direct links to the data (not even DataCite DOIs (yet))
- A key reason DOI's don't point data/papers is because not all are available online
  - Some obviously are pay walled
  - But some can't be digitalized at all. E.g. DOI's have been issued to laboratory samples, or to data centres.

DOI's are not the solution, but they are probably part of the solution

- The landing page is still just a website, it's domain name can expire
  - The data hosting is can expire (and hard-drives can fail).
- Did I mention under normal circumstances you can't go from DOI to data URL at all anyway?
- Still persistent metadata is also kind of nice to have
  - At least if the data goes down you know what was lost, so we can seek it else-where.
- Really, though we need to be applying periodic automatic link checking to all URLs we need not to break.
  - DataDeps.jl makes that pretty easy, you can just set it up as part of `runtests.jl` and set Travis or AppVeyor to run scheduled tests.