DataDeps.jl

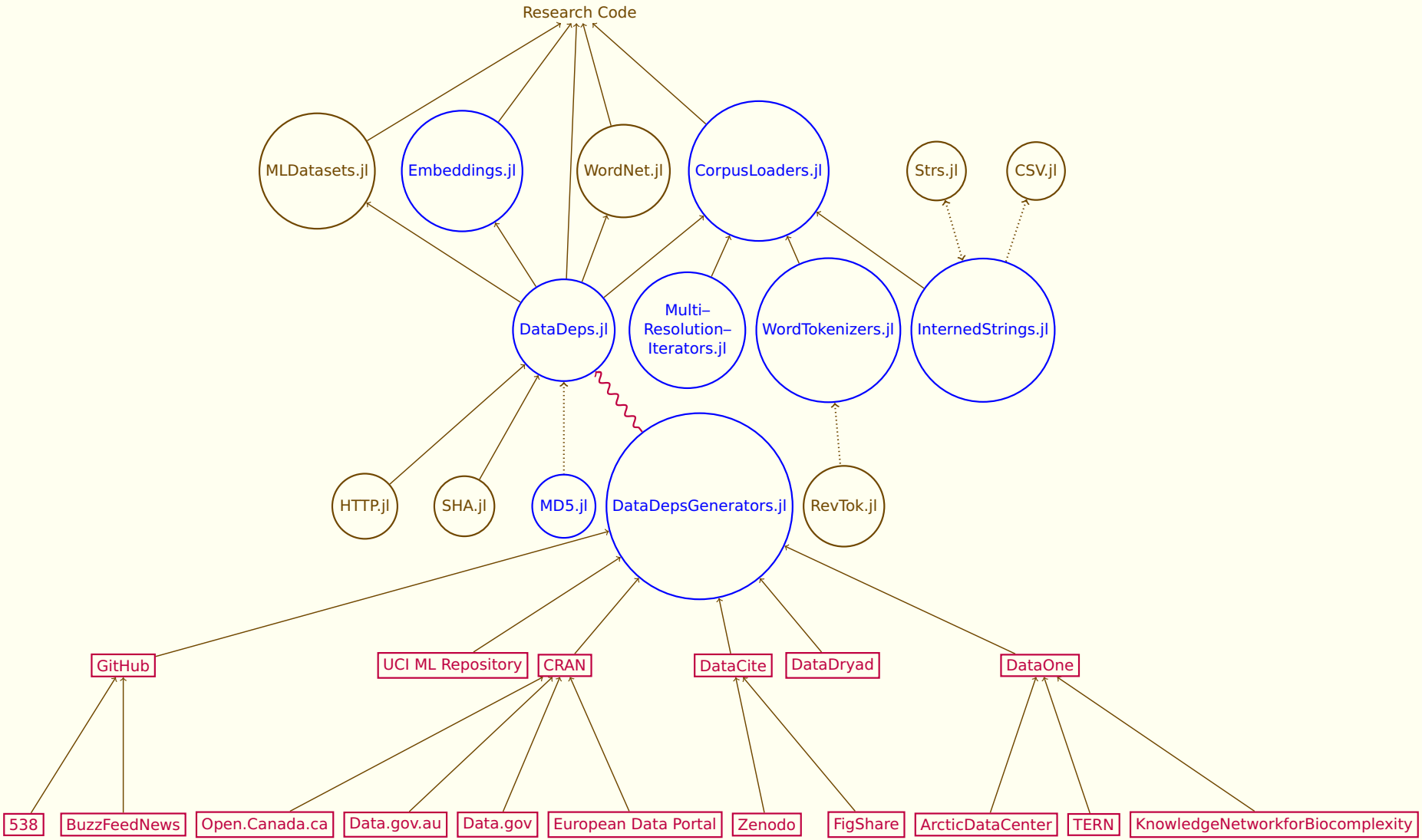
and other foundational tools for data driven research (Especially NLP)

Lyndon White

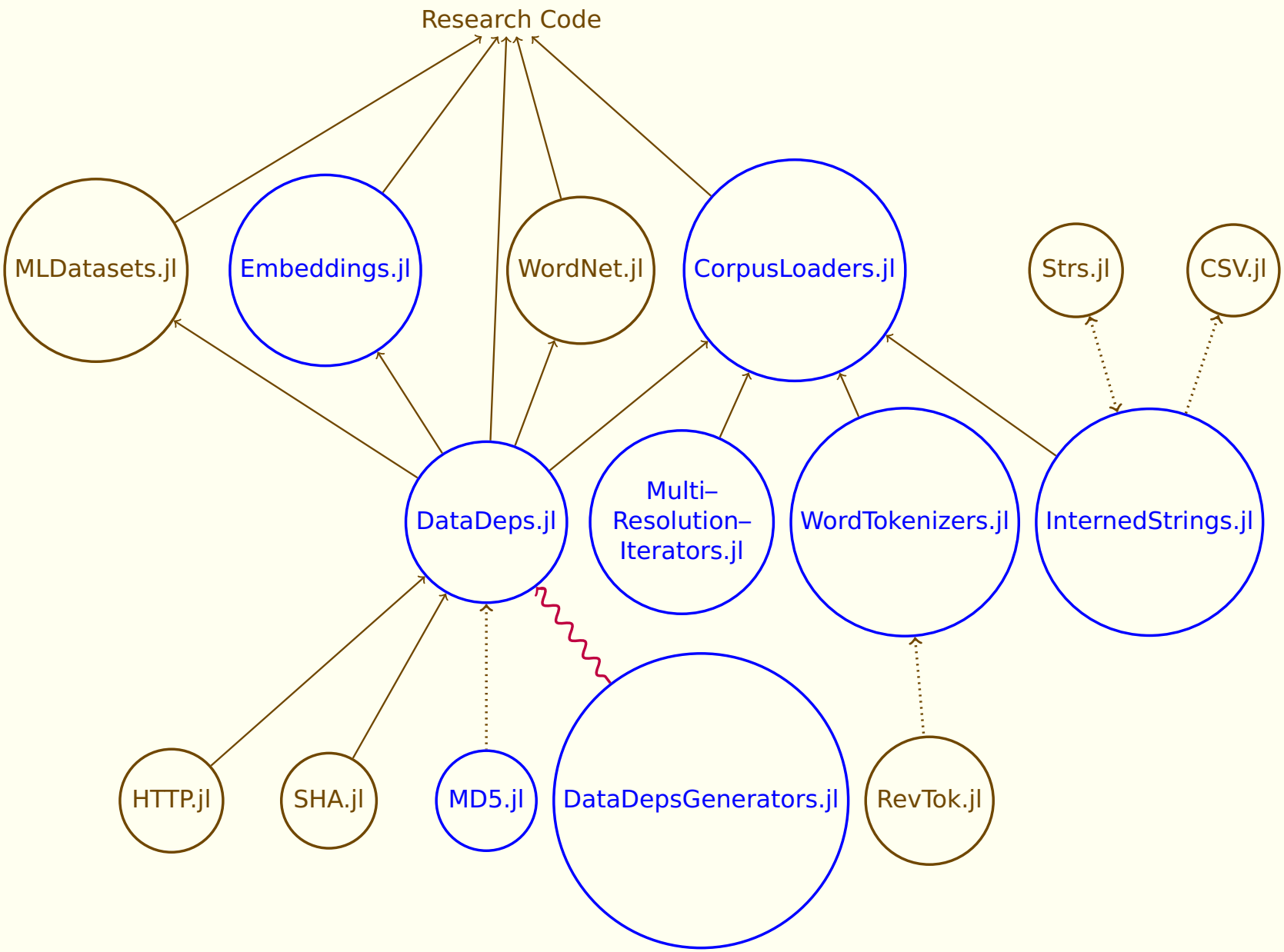
School of Electical, Electronic and Computer Engineering
The University of Western Australia

THE UNIVERSITY OF WESTERN AUSTRALIA

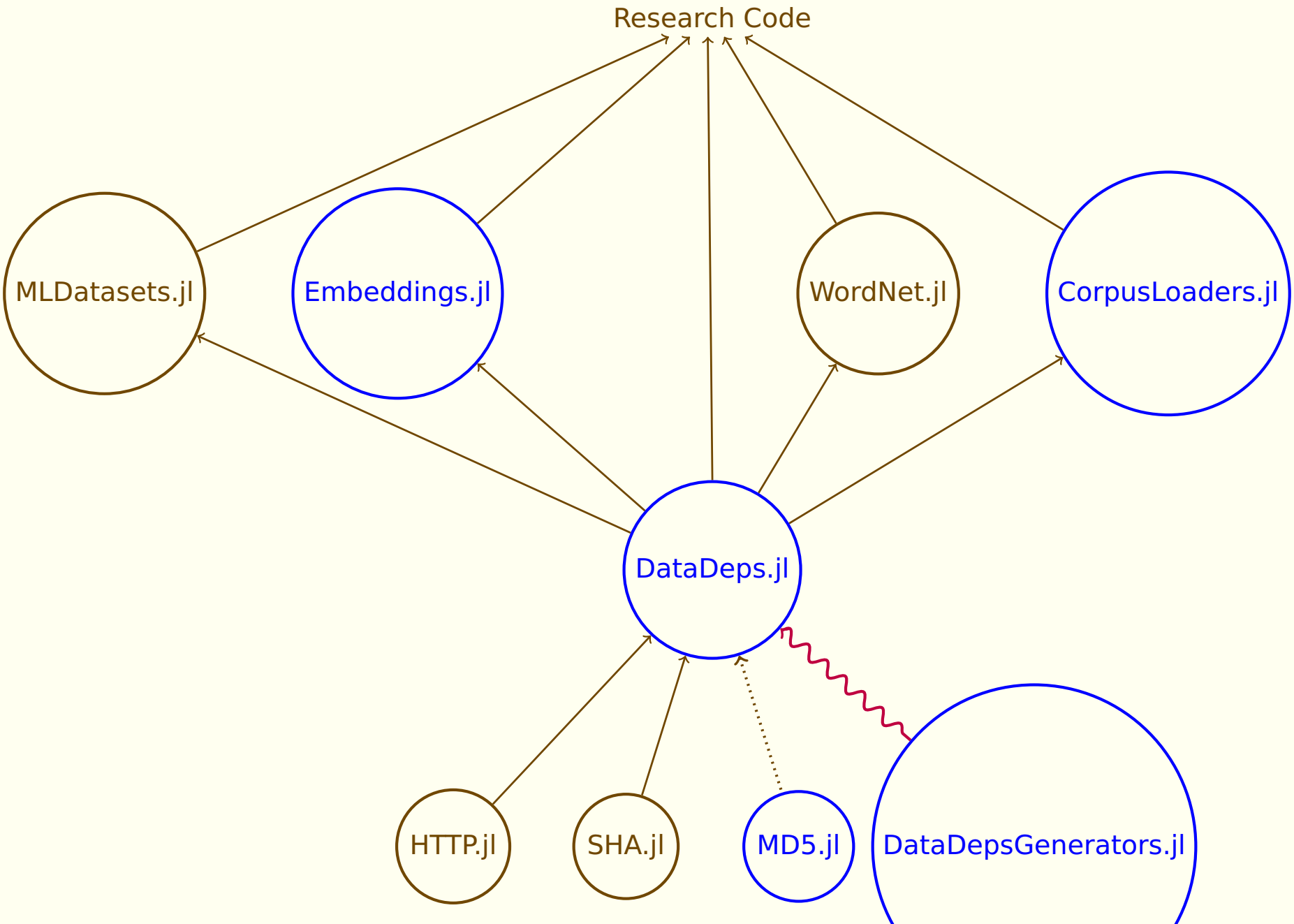
Big Picture


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

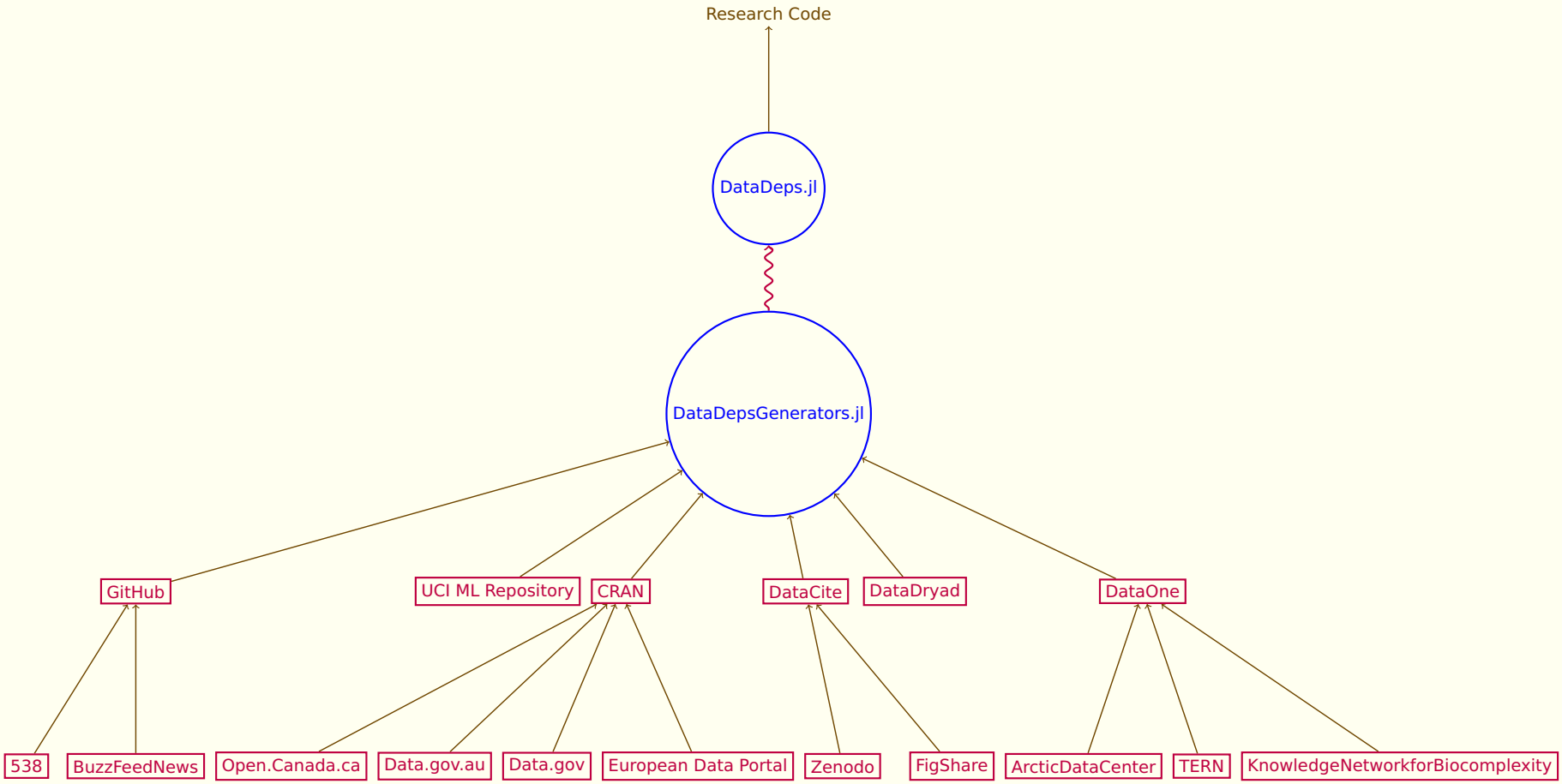
Julia Packages



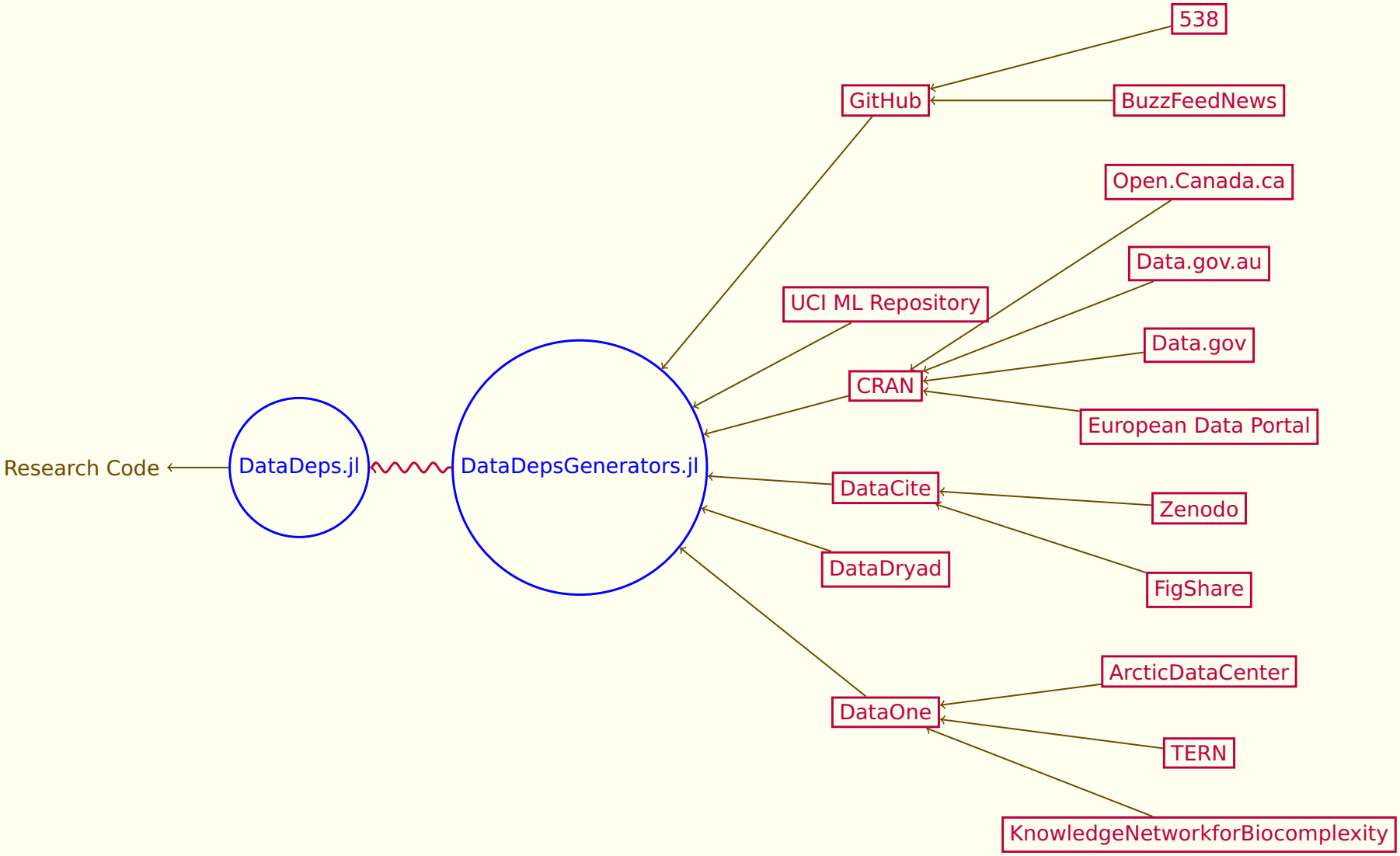
DataDeps.jl


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

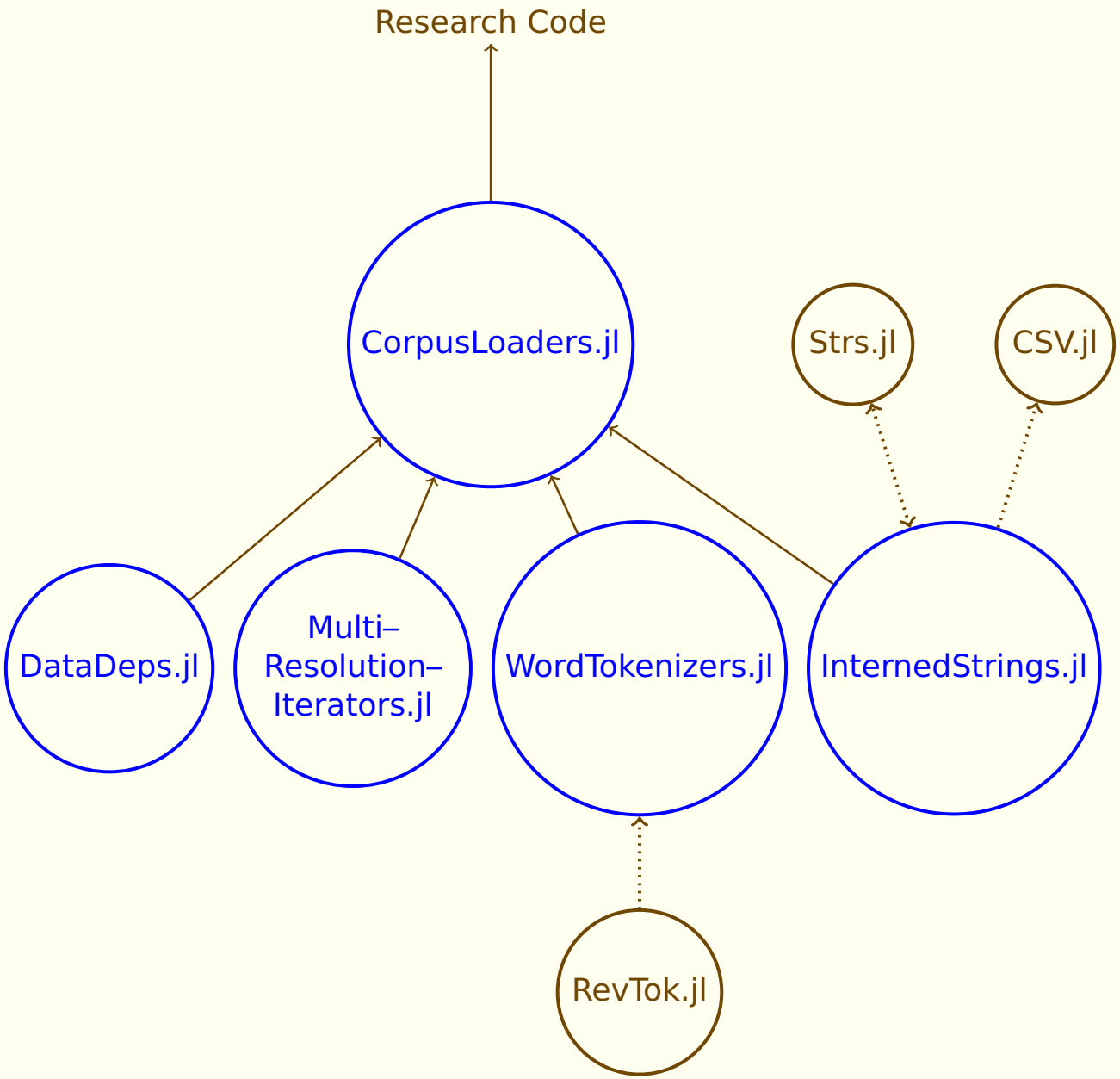
DataDepsGenerators.jl



DataDepsGenerators.jl



CorpusLoaders.jl



Code depends on data
Especially research code

- To manage your julia dependencies you use
  - Pkg
- To manage your binary dependencies you use
  - BinDeps,
  - Conda etc
  - or preferably BinaryProvider

How are you managing your data dependencies?

- Instructions in the README.md
  - So how does your CI work out?

How are you managing your data dependencies?

- Instructions in the README.md
  - So how does your CI work out?
- Sticking a `download` command, in `build.jl`
  - So it gets downloaded even if not used?

How are you managing your data dependencies?

- Instructions in the README.md
  - So how does your CI work out?
- Sticking a `download` command, in `build.jl`
  - So it gets downloaded even if not used?
- Just putting it in the git repo
  - I am glad it isn’t too large for git.

How are you managing your data dependencies?

- Instructions in the README.md
  - So how does your CI work out?
- Sticking a `download` command, in `build.jl`
  - So it gets downloaded even if not used?
- Just putting it in the git repo
  - I am glad it isn't too large for git.

These are all options.
But can we do better?

DataDeps doesn’t protect you from link breakages.
But at least you know when it is broken

- Nothing can truly protect against link-rot
  - Not even DOIs
  - The data must physically exist somewhere on a hard-disk. And someone must be maintainingthat

DataDeps doesn’t protect you from link breakages.
But at least you know when it is broken

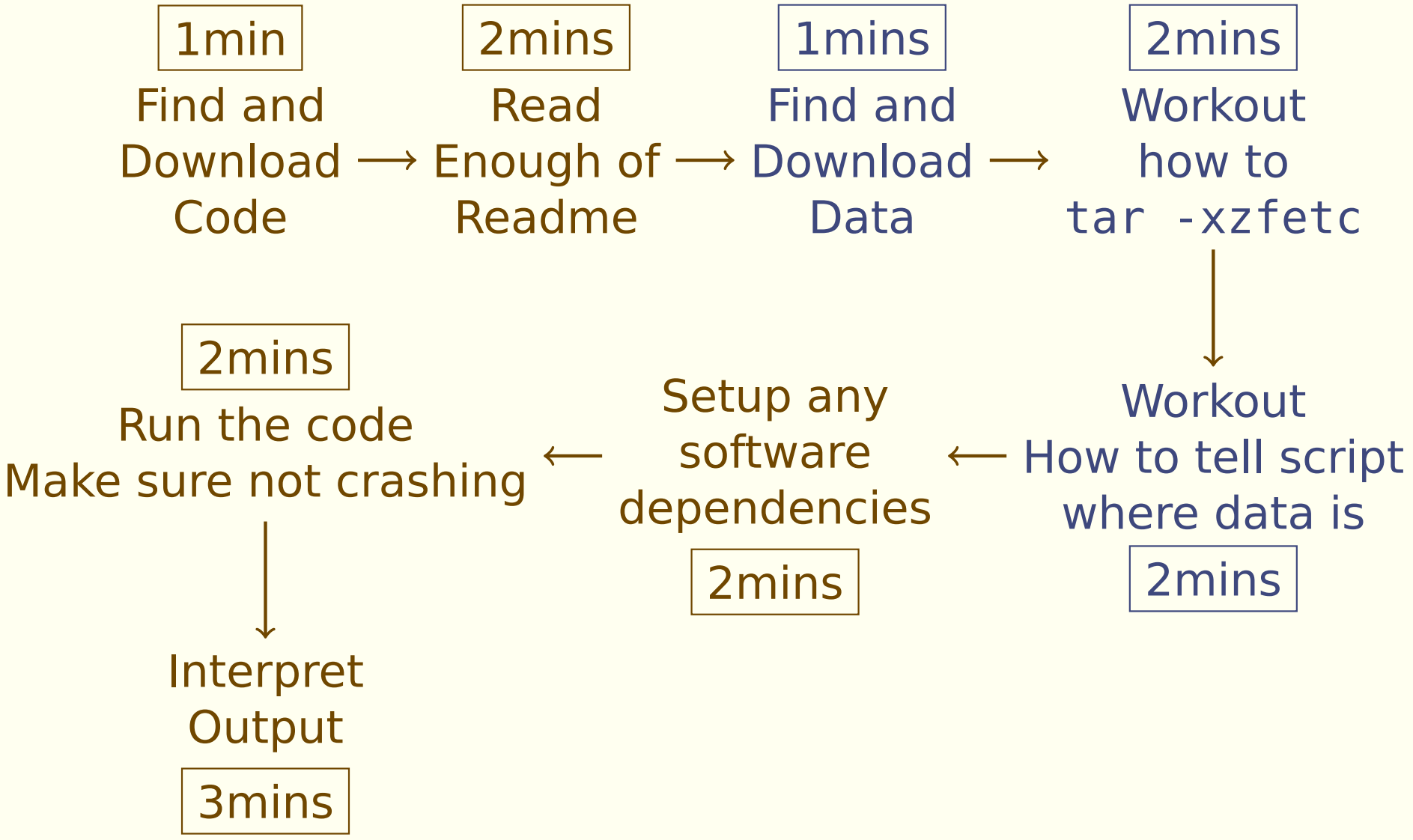
- Nothing can truly protect against link-rot
  - Not even DOIs
  - The data must physically exist somewhere on a hard-disk. And someone must be maintainingthat
- If you are downloading your data automatically you can check it is still there via automated testing
  - TravisCI and AppVeyor offer scheduled testing options.

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 run someone's research code?



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 user doesn’t have to input them.
- But they break if anything is moved.

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-codingit 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.

Automatic fetching is really convenient

Some things that have happened to me lately:

- Supervisor tells me shared workstation hard-drive is dangerously full
  - Delete all of them
  - knowing that the ones I am still using will be fetched as required.

Automatic fetching is really convenient

Some things that have happened to me lately:

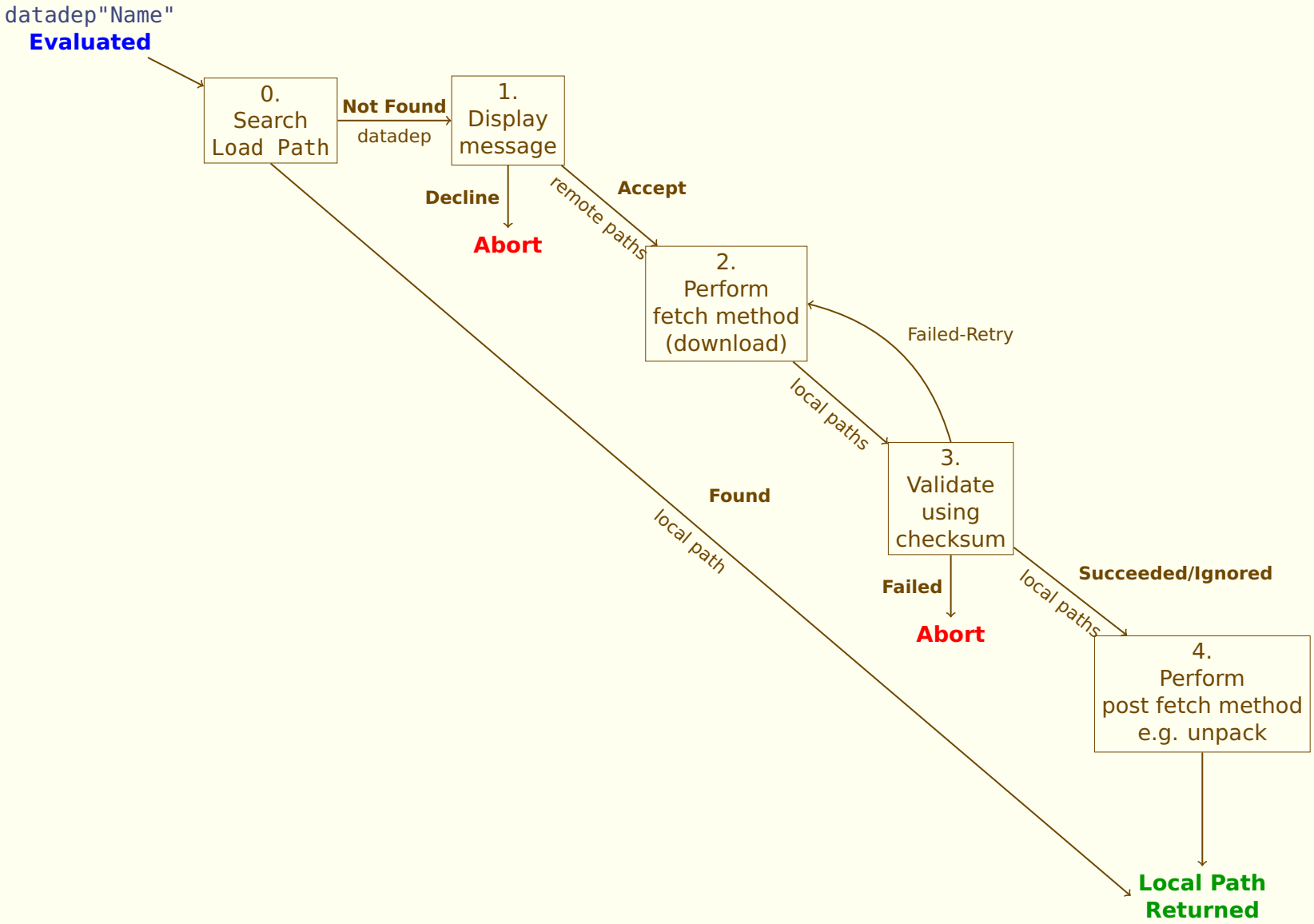
- Supervisor tells me shared workstation hard-drive is dangerously full
  - knowing that the ones I am still using will be fetched as required.
- Want to start up copy of work on another server to run an extra set of experiments in parallel.
  - Literally, just clone the project repo,
  - the data will make its own way there

Automatic fetching is really convenient

Some things that have happened to me lately:

- Supervisor tells me shared workstation hard-drive is dangerously full
  - knowing that the ones I am still using will be fetched as required.
- Want to start up copy of work on another server to run an extra set of experiments in parallel.
  - Literally, just clone the project repo,
  - the data will make its own way there
- Realize that I've modified one of the data files by mistake
  - `rm(datadep"Foo", recursive=true)`
  - Next time it is needed a new copy will be downloaded.

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]
))
```

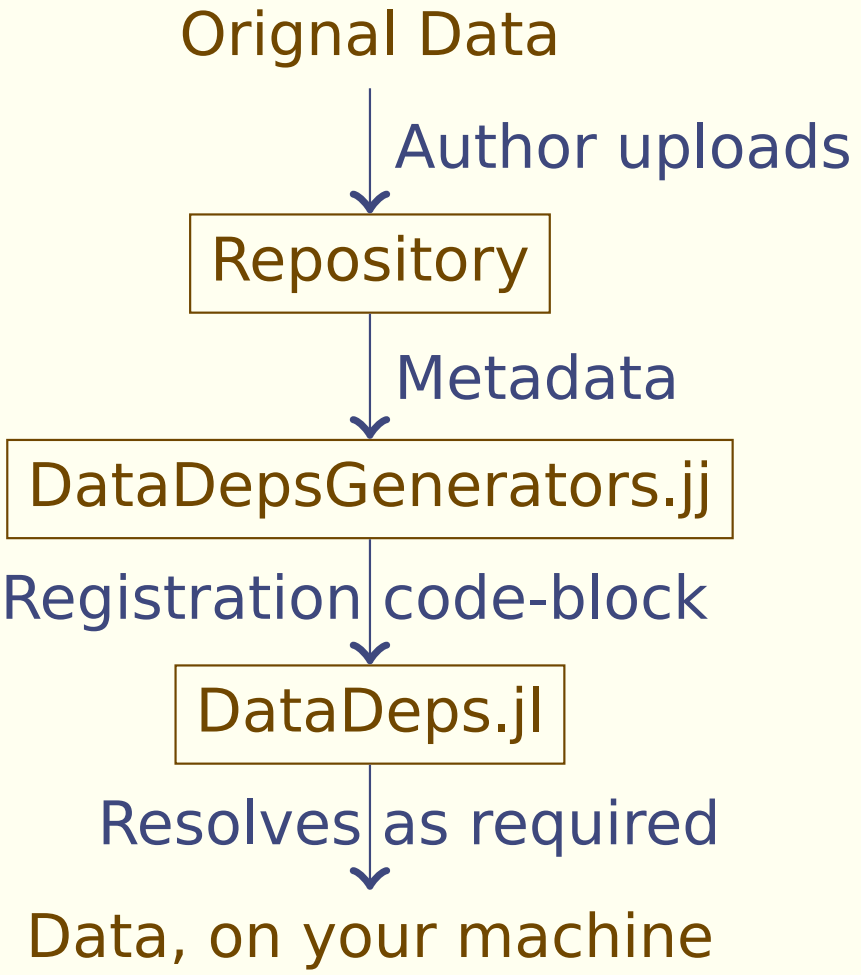
DataDepGenerators.jl

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

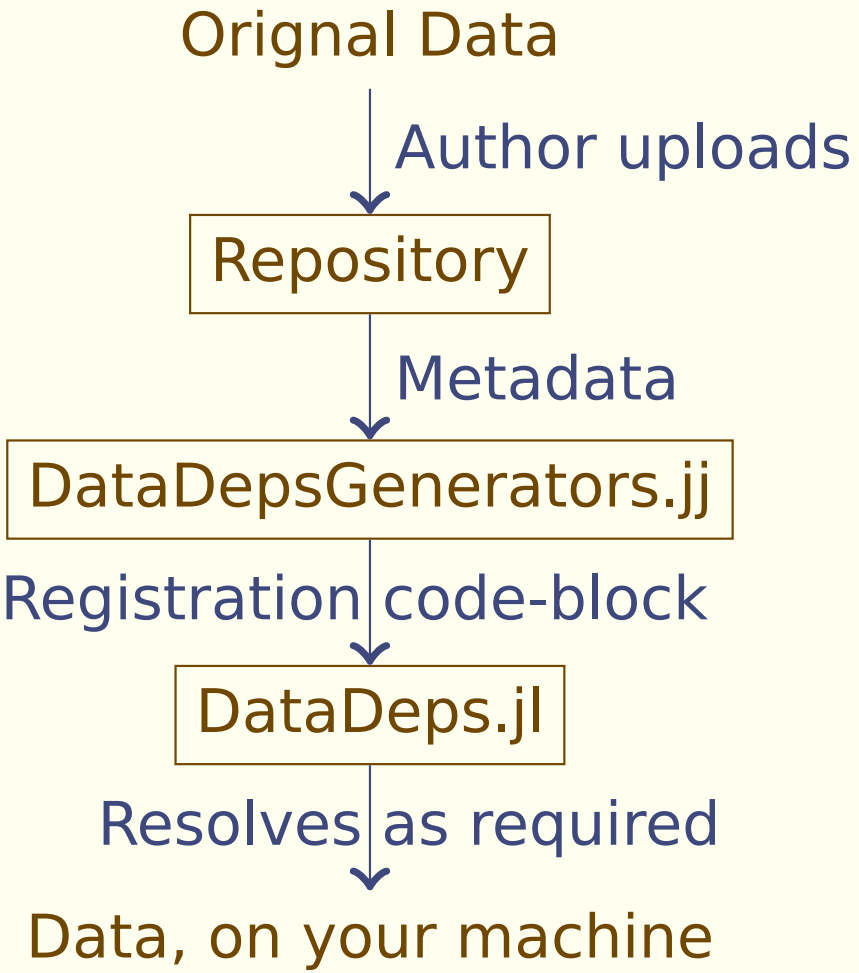
For published data this information is available from some API

- 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



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.

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.

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.

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 `WeakRefs` 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 `WeakRefs` strings can still be garbage collected.

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