1 Download OAI repository data and convert to RDF

1.1 Requirements:

- SWI-Prolog 5.8.0
- Triple20 (GIT version, see http://www.swi-prolog.org/web/Triple20.html

1.2 Installation

- On Unix, edit #!... line in `run.pl` to point to Prolog
- In `load.pl`, edit `file_search_path/2` to reflect location of Triple20 sources.
- In `oai.pl`, add identifiers and URLs of servers you want to access.

You can (re-)built the documentation on a Unix environment by editing the #!... line on `pltotex.pl` and running `make`. This generates a LaTeX file from `README` and the PlDoc comments in the sources and creates a PDF file from this using `pdflatex`

1.3 Running (example uses fontys):

- Start `run.pl`
- To get server identifier, run

```
?- sp(fontys).
```

- To get records

```
?- gr(fontys).
```

- To start Triple20 in environment, so you can examine the loaded data:

```
?- triple20.
```

- To generate an RDF file:

```
?- rdf_save('fontys.rdf', fontys).
```

1.4 Big repositories

Big repositories typically offer *ResumptionToken* support that splits the download into smaller chunks and can restart if an error occurs. This is supported using `oai_crawl/3` in `oairdf.pl`. Typical usage is below to crawl all data from a server and store it in the file `fontys.ttl` as Turtle data.

```
?- oai_crawl(fontys, 'fontys.ttl', []).
```

The last argument is a list of options. The supported options are given with the source of `oai_crawl/3`.

If the repository is split into many datasets, one may wish to use

```
?- oai_crawl_by_set(fontys, 'fontys.dir', []).
```

This creates a directory `fontys.dir` and a file for each dataset. See `oai_craw_by_set/3` for details and options.

1.5 Predicate reference

1.5.1 Download repositories to a a file

The predicates below are the main predicates for downloading (large) repositories.

oai_crawl(+*Server, +File, +Options*)
Collects all records from *Server* into *File*. Some useful options:

retry(+*Times*)
Retry the fetch max Times. Default is 200.

retry_delay(+*Seconds*)
Time to wait for the first retry. Default is 10 seconds. For subsequent retries, the time is doubled each try. until it reaches `retry_maxdelay`

retry_maxdelay(+*Seconds*)
Maximum delay between retries. Default is 3600.

resumption_count(+*Count*)
Do at most Count resumptions. Default is infinite.

metadataPrefix(+*Prefix*)
Meta-data format to collect. Default is `oai_dc`.

resumptionToken(+*Token*)
Start from the given resumption token

set(+*Set*)
Specify a dataset

from(+*From*)

until(+*Until*)
FIXME: OAI attributes; what do they do?

This predicate use the RDF graph `oai_crawler` for storing intermediate data. This graph is filled and wiped for each resumption-token.

oai_crawl_by_set(+*ServerId, +Dir, +Options*)
Crawl a server set-by-set, where each set creates a file <set>.ttl in the directory *Dir*. *Options* is passed to `oai_crawl/3`. In addition, we process the following:

if(*not_exists*)
If provided and the download file already exists, the set is skipped.

1.5.2 Explore (small) repositories from memory

These predicates downloads (partial) repositories into memory and allow to inspect the contents using the Triple20 ontology browser.

sp(+*ServerID*) *[det,private]*
Fetch server properties and store them into an RDF graph named *ServerID*.

gr(+*ServerID*) *[det,private]*
Fetch all records for *ServerID* and store them in the named graph *ServerID*. This works for relatively small repositories. For **big** repositories, see `oai_crawl/3`.

gr1(+*ServerID*) *[det,private]*
As `gr/1`, but does not resume.

triple20
Start the Triple20 RDF browser. This predicate is intended for embedding Triple20 into applications. The application can specify options by adding clauses for `triple20:option/1`.

oai_reset_warnings
Reset warnings about implicitely mapped properties that are already given.

1.5.3 Customization predicates

oai_server(+*ServerID, -URL*) *[multi]*
Defined short-names for OAI servers. Please add rules to this predicate to define your own servers.

1.6 Roadmap

`run.pl` Toplevel driver

`load.pl` load the components

`servers.pl` Defines aliases for OAI servers. see `oai_server/2`.

`oai.pl` : AOI HTTP client, calling user hook to process OAI content.

`oairdf.pl` Uses `oai.pl`, providing a hook to translate into RDF

`t20_extensions.pl` Define some extensions to Triple20

`pltotex.pl` Call PlDoc to create PDF documentation from `README`

Makefile Drive translation of PDF documentation

oai.ttl RDFS description of AOI records (partial) used to guide translation.