D2.4

Improved metadata harvesting workflow

Document information

Title	Improved metadata harvesting workflow
ID	CLARINPLUS-D2.4 (CE-2016-0839)
Author(s)	Menzo Windhouwer (CLARIN ERIC), Tomasz Naskręt (CLARIN PL)
Responsible WP leader	Dieter van Uytvanck (CLARIN ERIC)
Contractual Delivery Date	2016-08-31
Actual Delivery Date	2016-08-31
Distribution	Public
Document status in workplan	Deliverable

Project information

Project name	CLARIN-PLUS
Project number	676529
Call	H2020-INFRADEV-1-2015-1
Duration	2015-09-01 – 2017-08-31
Website	www.clarin.eu
Contact address	contact-clarinplus@clarin.eu

Table of contents

1 Executive Summary

This deliverable describes the work done on CLARIN's OAI harvester. The improvements mainly focussed on making it possible to handle a large number of endpoints with potentially a large number of records. Interaction with the OAI endpoints is improved by implementing the *ListRecord* scenario, which needs fewer requests to retrieve the records offered, and incremental harvesting, which allows to only downloading newly or updated records since the last harvest. For the system administrators of the CLARIN centres inspection of the harvesting results have become easier by endpoint specific log files and a new harvest viewer. This viewer also provides a jump off point to the metadata quality and curation dashboard developed in CLARINPLUS-D2.1.

2 Introduction

For its core catalogue service, the Virtual Language Observatory, the CLARIN infrastructure relies on an OAI (Open Archive Initiative; [1]) harvester that periodically harvests all CMD (Component MetaData; [2]) records from the CLARIN centres. The OAI harvester has been developed in house and has been improved over the years [3]. But still it lacks some functionality, which will help the harvesting process to keep up with the scale of growth of both the number of CLARIN centres and the number of CMD records. This deliverable describes the improvements made in CLARIN-PLUS, namely

- Endpoint specific logging;
- *ListRecords* scenario;
- Incremental harvesting;
- New OAI harvest viewer.

The next sections will describe these improvements in more detail.

3 Endpoint specific logging

During a harvest a log is created. The log tells which endpoints will be harvested and, when it is the endpoint's turn, how the interaction with the server went, *i.e.,* which records were retrieved and/or which errors occurred. When a centre's system administrator has questions about the harvest, the log used to be manually analysed by one of CLARIN's system administrators. Logging messages are now targeted at endpoint specific logs and made available via the OAI harvester viewer, which enable the centre's administrator to quickly determine if there have been any problems with the endpoint during the harvest.

3.1 Implementation

To complete this task the logging library from log4j has been upgraded to the newest version of log4j2 library [4], which offers additional capabilities for managing logging process in multi-threaded applications.
By changing configuration settings for logging, and adjusting initialization for *Workers* it became possible to route logging information's to separate files for each provider, i.e., the new implementation uses the log4j2 *RoutingAdapter* scenario to pipe logging information from the active thread to the correct log file.
As the result of these changes we get a log file, which contains only specified information about the current application runtime and separate log files for each harvested provider.

4 ListRecords scenario

There are various ways, also known as scenarios, to use the OAI-PMH protocol (OAI Protocol for Metadata Harvesting; [5]) to harvest the records provided by an endpoint. Initially the harvester supported only the *ListIdentifiers* scenario, which starts with creating a list of all record identifiers and subsequently these records are retrieved one by one from the endpoint. This does mean a lot of requests to the endpoint. Now a second scenario, the *ListRecords* scenario, is also supported. In this case the endpoint sends all records in batches to the harvester. This is means less requests from the harvester to the endpoint, but comes with its own challenges:

1. The endpoints determine the size of the batches, which can lead to higher memory needs for the harvester;
2. The endpoints take longer to collect the batches.

Work has been done on the harvester to properly support the *ListRecords* scenario and to deal as good as possible with these challenges.

4.1 Implementation

The harvester runs every request to a set of actions, which can be configured by the user. In the *ListIdentifiers* scenario these actions had to only process one record. In the *ListRecords* scenario batches of records have to be processed. All actions were changed to be able to do so, which makes it possible to change the scenario in a configuration file without touching the actions.
There are various options to process XML (eXtensible Markup Language; [6]) documents, as the OAI-PMH responses and CMD records are. The common option, also used by the harvester, is to parse the XML into an in-memory tree objects, a Document Object Model (DOM; [7]). This makes processing easy as one can traverse the tree in any direction and get the information needed. However, the DOM can need 3 to 8 times more memory than the original XML document [8]. Another approach is to act directly on the events emitted by a XML parser, *e.g.*, StAX [9]. This approach is now used by the harvester as long as possible, *e.g.*, all processing of the OAI envelope of a response can be handled via streaming meaning that a record that doesn't need any transformation will never have to be loaded into a DOM.

Still some endpoints can cause problems because they can take a long time to create their batches or they create very big batches due to record sizes. It is now possible to add specific configuration options to those endpoints that override the general configuration:

- Scenario: use *ListIdentifiers* for this endpoint instead of *ListRecords*;
- Timeout: use a bigger timeout for this endpoint;
- Exclusive: harvest this endpoint when no other endpoints are being harvested.

The harvest of specific endpoints can thus be tweaked, while the general settings remain optimal for the bulk of endpoints.

4.2 Performance

Several runs with a limited number of endpoints were done to gather some statistics on the differences between the *ListIdentifiers* and *ListRecords* scenarios:

Indicator	ListIdentifiers	ListRecords	Difference
Number of requests	60,303	22,870	**-62.08%**
Memory load	37.69MiB	48.16MiB	**+27.76%**
Running time	2:12:58	1:05:53	**-49.52%**

5 Incremental harvesting

The OAI harvester used to request all the CMD records from an OAI endpoint. However, the OAI-PMH standard, which describes the protocol used by the harvester, also has provisions for incremental harvesting. This has the potential to speed up the harvesting process considerably, but several requirements need to be met:

1. The endpoint needs to support this part of the protocol, and as some do not it should always be possible to fall back to a full harvest of such an endpoint;
2. The harvester needs to keep information about previous runs, *e.g.*, timestamps;
3. To be able to offer all CMD records to the VLO, the previously harvested records still need to be available, or the VLO needs to be able to process the delta information (*i.e.,* which records were deleted).

Before CLARIN-PLUS, some preliminary work on incremental harvesting was done, *e.g.,* an overview file is kept containing information on the last run to be used by a subsequent incremental run (see requirement 2). However, the full workflow to use incremental harvesting in production was never realized. Now it is.

5.1 Implementation

The existing foundations have been extended into a working feature. Every scenario can now be run in incremental mode by setting the incremental option to true in the settings section in of the overview file. If the global support for incremental harvest is turned on and a provider supports it an incremental harvest can be done.
In a nutshell the basic algorithm for incremental harvest is:
A. When the OAI Provider works in *TRANSIENT* or *PERSISTENT* deletion mode:

1. Check the existing overview file for last running mode for the endpoint. Continue if incremental mode is available.
2. In the current harvest download only records with modification date after last harvest date.
3. Every save action creates a new record in a history file. For monitoring purposes we register: harvest date, file name and operation type.
4. During the harvesting process we are able to extract information if the file was marked as deleted. Information is saved to a helper file, which will contain the list of file names to remove.

5. As the last operation in the harvest process we are going to execute the file synchronization process. This process removes all marked files that were listed in the helper file. The history file is updated with records for removed files.

B. When OAI Provider works in *NO* deletion mode:

In this case the record header doesn't contain information about deletion. The only way to synchronize state of repository is by comparing with current state on server.

1. Check existing overview file for last running mode for processed host. Continue if incremental mode is available.
2. In the current harvest download only items with modification date after the last harvest date.
3. Create entries in the history file for newly downloaded records.
4. Execute the following synchronization process:
   a. Use *IdentifierHarvest* to extract list of currently available records on the server and save it to a temporary file.
   b. Move files listed in temporary file to a new directory.
   c. Remove all files that are left in the current harvest directory and create deletion records in the history file.
   d. Remove the old directory.
   e. Rename the temporary directory to the correct name.

Of the 34 CLARIN centre endpoints [10] inspected[1] the distribution of the deletion mode was:

- *TRANSIENT*: 15
- *PERSISTENT*: 4
- *NO*: 15

So a slight majority of CLARIN endpoints will support incremental harvest natively. Fortunately the fallback incremental harvesting strategy for the endpoints with *NO* deletion information also decreases the number of requests and processing time needed compared to a full harvest.

In summary using incremental harvesting offers:

- A shorter processing time;
- In case of failure we don't have to download all repository metadata again, but we can start from a specific date by editing the overview file.

5.2 Performance

The following table shows the behaviour of the incremental harvests:

	22-08-2016[2]			24-08-2016			26-08-2016			29-08-2016		
Endpoint[3]	Requests	Records	Time(s)	Requests	Records	Time(s)	Requests	Records	Time(s)	Requests	Records	Time(s)
clarinoai.informatik.uni-leipzig.de	30	7,474	1,046	1	0	9	1	0	9	1	0	12
openscience.uni-leipzig.de	2	144	22	1	0	7	1	0	8	1	0	7
www.phonetik.uni-muenchen.de	49	24,332	4,361	1	0	9	5	2265	452	1	1	4
clarin.bbaw.de	12	3,034	66	13	0	8	13	0	9	13	0	9
clarin.oeaw.ac.at	4	7	56	1	0	10	1	0	11	1	0	10
metashare.ut.ee	1	60	100	1	0	19	1	0	15	1	0	16
clarin.dk	297	147,114	1,372	1	0	7	1	0	7	1	0	7
clarin-pl.eu	3	289	60	1	1	5	1	1	5	1	0	6
clarino.uib.no	10	196	54	20	196	54	20	196	54	20	196	54
repo.clarino.uib.no	1	36	9	1	0	3	1	0	3	1	0	3
www.clarin.si	1	44	30	1	1	15	1	1	11	1	0	15
alpha.talkbank.org (1)	211	40,969	429	1	0	6	1	0	6	1	0	6
alpha.talkbank.org (2)	89	16,527	238	1	0	13	1	0	12	1	0	13
cocoon.huma-num.fr	244	10,388	361	1	13	6	1	8	5	1	11	5
repository.sfb833.uni-tuebingen.de	1	0	1	1	0	1	1	0	1	1	0	1
weblicht.sfs.uni-tuebingen.de	58	58	64	59	0	60	59	0	60	59	0	60
metalb.csc.fi	4	351	13	1	0	13	1	0	13	1	0	13
corpora.uni-hamburg.de	26	2,496	163	27	0	60	27	0	60	27	0	60
repos.ids-mannheim.de	2,071	1,934	7,774	1	0	7	1	0	7	1	0	7
dspace-clarin-it.ilc.cnr.it	1	3	1	1	0	1	1	0	1	1	0	1
clarin04.ims.uni-stuttgart.de	1	69	28	1	0	17	1	0	15	1	0	15
repository.clarin.inl.nl	1	21	24	1	0	10	1	0	11	1	0	12
lindat.mff.cuni.cz	14	1,172	25	1	0	5	1	0	7	1	0	7

[1] On August 17, 2016.
[2] On August 22, 2016 a full harvest took place.
[3] Only the host name is given due to space considerations.

Endpoint[3]	22-08-2016[2]			24-08-2016			26-08-2016			29-08-2016		
	Requests	Records	Time(s)	Requests	Records	Time(s)	Requests	Records	Time(s)	Requests	Records	Time(s)
www.meertens.knaw.nl	250	249,659	4,673	1	0	8	1	0	8	1	0	8
www.nb.no	1	134	16	1	0	7	1	0	7	1	0	9
www.ota.ox.ac.uk	5	4,642	87	6	0	40	6	0	40	6	0	40
sldr.org	1	268	15	1	0	7	1	0	9	1	0	7
ws02.iula.upf.edu	1	254	20	1	0	3	1	0	4	1	0	4
fedora.clarin-d.uni-saarland.de	1	112	17	1	0	4	1	0	5	1	0	5
clarin.ids-mannheim.de	1	5	14	1	0	7	1	0	5	1	0	6
dspace.library.uu.nl	642	64,140	651	1	0	11	1	0	9	6	507	30
oai.beeldengeluid.nl	231	46,156	2,325	1	0	6	1	0	7	1	0	7
TOTAL	**4,264**	**622,088**	**24,115**	**152**	**211**	**438**	**156**	**2,471**	**876**	**157**	**715**	**459**

6 New OAI harvest viewer

The harvester used to contain a script to create a set of static HTML pages to enable browsing of a harvest result. Each endpoint was represented by one page containing links to all the harvested records. However, such a static page becomes huge when an endpoint provides thousands of records. A new harvest viewer with a more dynamic nature was created [11]. After the harvest a small tool crawls the results and the information on endpoints, requests and records is loaded into a database. Using an REST API [12] the database is queried by a dynamic web page, *e.g.*, to provide a paged list of all records harvested from a single endpoint and links to other CLARIN services.

6.1 Implementation

After a harvest the *oaiViewer* tool quickly crawls the results using a StAX parser. It creates a network of requests, records and the related files and stores it in a PostgreSQL database [13]. Using Dreamfactory [14] a REST API on this database is provided. A React-enabled web page [15] knows how to interact with this API and provides access to the endpoints, the requests, the records and the logs. This viewer can now also function as a jump off point to other CLARIN services. Especially interesting here is a link to the metadata quality and curation dashboard created in CLARINPLUS-D2.1. Using this a provider can see how the latest harvest went and also jump quickly to the dashboard to assess the quality score of the harvested records.

7 Conclusion

In CLARIN-PLUS the OAI harvest has been improved to a level that can handle a higher number of endpoints and records. Although some interactions are already in place further integration with other CLARIN services is still needed. For example, signalling when a harvest is finished so another process, *e.g.*, assessment of the quality of the CMD records, can pick up the results.

References

[1] Wikipedia. *Open Archives Initiative.* https://en.wikipedia.org/wiki/Open_Archives_Initiative

[2] CLARIN. *Component Metadata Infrastructure*. https://www.clarin.eu/cmdi

[3] CLARIN. *A simple Java application for managing an OAI-PMH harvesting workflow*. https://github.com/clarin-eric/oai-harvest-manager

[4] Apache. *Log4j 2*. http://logging.apache.org/log4j/2.x/

[5] Open Archives Initiative. *Protocol for Metadata Harvesting*. https://www.openarchives.org/pmh/

[6] W3C. *Extensible Markup Language*. https://www.w3.org/XML/

[7] W3C. *Document Object Model*. https://www.w3.org/DOM/

[8] Michael Kay. *Comparing DOM and other object models*. http://dev.saxonica.com/blog/mike/2012/09/comparing-dom-and-other-object-models.html

[9] Oracle. *Streaming API for XML.* https://docs.oracle.com/javase/tutorial/jaxp/stax/

[10] CLARIN. *Centre Registry*. https://centres.clarin.eu/oai_pmh

[11] CLARIN. *A web application to inspect an OAI harvest*. https://github.com/clarin-eric/oai-harvest-viewer

[12] Wikipedia. *Representational state transfer*. https://en.wikipedia.org/wiki/Representational_state_transfer

[13] The PostgreSQL Global Development Group. *PostgreSQL: The world's most advanced open source database*. https://www.postgresql.org/

[14] DreamFactory. *Turn any database into an API Platform.* https://www.dreamfactory.com/

[15] Facebook. *React: A JavaScript library for building user interfaces*. https://facebook.github.io/react/