Online Speech and Language Resources

Saturday, 31st May 2014

Presenters:

Dieter van Uytvanck, Max-Planck Institute of Psycholinguistics Nijmegen

Christoph Draxler, Ludwig-Maximilian University Munich

Thomas Eckart, University of Leipzig

Daniel Jettka, University of Hamburg
Tutorial Programme

08:45-09:00 Registration

09:00-10:30 Session Resource and Tool Creation

  Metadata (Dieter van Uytvanck)
  Web Services (Christoph Draxler)

10:30-11:00 Coffee break

11:00-12:30 Session Online access to Resources

  Persistent Identifiers (Thomas Eckart)
  Repositories (Daniel Jettka)
Tutorial Outline

Research and development in speech and language technology are facing a fundamental paradigm shift. More and more speech and language resources, tools and even entire workflows are becoming accessible online in the context of the emerging research infrastructures.

In this tutorial, leading experts in the field of speech and language resources will discuss both opportunities and challenges of online resources. They will present state of the art technology for metadata, web services, persistent identifiers, and human and machine readable online repositories. Showcases and real world applications will illustrate how these technologies can be put into use.

The target audience of the tutorial are professionals from speech and language technology development, research and higher education.

The tutorial consists of four presentations organised in two sessions. Each session consists of presentations and live demos of online resources and services.

Resource and Tool Creation

Metadata describes speech and language corpora, tools and other linguistic resources so that they can be indexed. A metadata schema must fulfil contradicting requirements: it must be sufficiently precise to adequately describe a resource, but it must be general enough to cover the wide variety of existing resource types; it must be stable for long-term access, but it must be easily adaptable to new types of resources or technical developments; it must be fine-grained to capture the relevant details of a resource, but at the same time efficient and easy to use by humans and automated processes. Finally, a metadata schema should be as theory-neutral as possible with respect to the primary resources to allow for broad application across disciplines.

Web services are increasingly becoming popular in speech and language processing. Many tools have been freely available for years, but the efforts and technical expertise needed to install or run them locally prevented their wide adoption. Webservices offer an elegant solution: a tool runs on a server, and remote clients, e.g. web browsers, standalone annotation tools or application programs, access and exploit these services via the net.

In the tutorial, we present a component-based and self-describing metadata schema built on the foundation of agreed standards and terminology in the field and show how tools can be used to generate metadata descriptions with minimum effort. Furthermore, the design and implementation of webservices, and their description with metadata, is presented in some detail; as a case study, the automatic speech segmentation system of the BAS will be used.

Persistent Identifiers and Repositories

Speech and language resources, as well as tools and processing workflows, evolve over time. Persistent identifiers provide a way of assigning a unique and immutable identifier to a specific version of a resource, and they may be used to refer to this resource independently of its physical storage location or means of access.

Repositories provide controlled access to language and speech resources and services both to humans, e.g. via a browser, as well as to automated processes, e.g. search engines or harvesters. Repositories require a minimum set of metadata, a flexible and powerful storage management, and access authorization, amongst other features. Although there exist software packages with repository functionality, they require considerable technical expertise to maintain.
In the tutorial, we present the alternative schemes for obtaining and maintaining persistent identifiers. Furthermore, we discuss the far-reaching consequences - including the benefits - of providing persistent identifiers for one's own resources, with a special focus on versioning and long-term storage. With respect to repositories we give an overview of existing software solutions including the integration of repositories and content management systems, and discuss in some detail the technical aspects of querying and harvesting language and speech repositories. As a case study, we will present the repository and services of the CLARIN-D centre Leipzig.

**Tutorial Presenters**

*Christoph Draxler*, Bavarian Archive for Speech Signals, LMU Munich, head of the corpus and tools group. He has developed a number of speech tools, e.g. SpeechRecorder, WebTranscribe, and percy, and he was responsible for the collection of several large-scale speech databases, e.g. SpeechDat II and SpeechDat-Car (German), Ph@ttSessionz, VOYS

*Thomas Eckart*, Natural Language Processing Group, University of Leipzig, research associate. After graduating in Computer Science at the University of Leipzig he worked in projects on the creation and usage of large written language resources in computer linguistics, Digital Humanities and in infrastructure projects. His research interests are methods for quality assurance of textual resources and the interpretation of component-based metadata. He is co-developer of the Virtual Language Observatory (VLO).

*Daniel Jettka*, Hamburger Zentrum für Sprachkorpora, Hamburg, research associate. After studying General and Computational Linguistics, Text Technology, and Social Sciences at Universität Bielefeld/Germany and Trinity College Dublin/Ireland he joined the HZSK in early 2012. He worked on the implementation of the HZSK Repository for Spoken Language Corpora, and created webservice for the conversion and visualization of transcription formats. His main research interests include Text Technology, Corpus Linguistics, Research Infrastructures, XML Technologies, and Data Visualization.

*Dieter van Uytvanck*, Max-Planck-Institute of Psycholinguistics, Nijmegen, is a research infrastructure specialist at The Language Archive. He graduated in Informatics (Ghent University) and Language and Speech technology (Radboud University Nijmegen) and has been involved in technical infrastructure building for LRT purposes since 2008.

*Susanne Haaf*, Berlin-Brandenburgische Akademie der Wissenschaften, and *Thomas Kisler*, Bavarian Archive for Speech Signals, contributed substantially to the material presented in this tutorial.

**Acknowledgements**

CLARIN-D is funded by the German Federal Ministry of Education and Research (BMBF) under grant no. 01UG1120I.
Links

BAS Web Services: http://clarin.phonetik.uni-muenchen.de/BASWebServices

CLARIN Centre Registry: http://centerregistry.clarin.esc.rzg.mpg.de/

CLARIN CMDI: http://clarin.eu/cmdi

CLARIN-D Federated Content Search: http://weblicht.sfs.uni-tuebingen.de/Aggregator/

CLARIN Persistent Identifiers: https://www.clarin.eu/content/goals-and-requirements-pid-systems

CLARIN on Repositories: https://www.clarin.eu/content/repositories

CLARIN XSLT stylesheets for converting CMDI: http://www.clarin.eu/faq-page/274


Fedora Commons: http://www.fedora-commons.org

Handle System: http://www.handle.net/hs_manual/server_manual_1.html - SEC1

Islandora Project: http://islandora.ca/

ISOcat Registry: http://www.isocat.org

OAI-PMH: http://www.openarchives.org/pmh/

Shibboleth: https://shibboleth.net/

Virtual Language Observatory: http://catalog.clarin.eu/vlo/
Online Speech and Language Resources

Metadata: specification, creation and use

Susanne Haaf (BBAW)
Dieter Van Uytvanck (CLARIN ERIC)

Overview

- Introduction and definition
- Traditional metadata
- Component metadata
- Data categories
- The big picture
- In practice:
  - Building components
  - Using components

Metadata?

- Data about data
- More exactly: **structured** data about data
  - Not just prose (although that can be a part)
  - But keyword/value type of data:
    - Name = “Nordic Syntax Database”
    - Languages = “Danish, Faeroese, Icelandic, Norwegian, Swedish”
- Used for:
  - Resource discovery / accessing
  - Management

Metadata?

- In this context: description of language resources and tools
  - for human consumption
  - for machine processing
- Different levels of description (granularity):
  - complete corpora, e.g. Brown Corpus.
  - subcorpora or corpus components, e.g. all Flemish recordings in the Spoken Corpus Dutch
  - (recording) sessions, e.g. the recording of a dialogue (sound file + transcript)
  - individual resources, e.g. a text file
Traditional Metadata

- Lack of flexibility
  - Too many fields...
  - ... but the one you are looking for is missing
- Lack of interoperability
  - My metadata does not work with your infrastructure
  - Vocabularies (and their semantics) often problematic:
    - Nederland? Netherlands? The Netherlands? Holland? NL?
    - community-specific terminology

Component Metadata

- Metadata infrastructure based on a “Component Metadata Model”
- Aims
  - Flexibility
    - Researcher can specify her/his needs
    - Offer ready-made metadata components
    - Allow creation of new metadata components needed
  - Interoperability built-in
  - Complete Infrastructure: software for editing, harvesting, exploitation
  - Compatibility with existing frameworks
Some terminology (1)

- **Element** = atomic unit (a “field”) – e.g. recording date
- **Instance** = one metadata description – e.g. myresource.cmdi
- **Schema** = technical (formal) grammar describing a profile – e.g. olac.xsd

Some terminology (2)

Metadata **Component**: An aggregation of metadata elements and other components aimed at describing a specific aspect of a resource.

**Reusable building block**

Metadata **Profile**: An aggregation of metadata components and elements that can be used to create metadata descriptions. The profile is used to describe all relevant aspects of a resource or collection.

**Blueprint for metadata description of a resource**

Example Profile

- Goal: Create metadata for a comic book resource

Comic book: Resource

First edition of "The Green Hornet" in which The Green Hornet and Kato combat a ruthless gang whose victims are forced to obey criminal commands!

Written by [Fran Striker (U.S., 1903)] published by Gold Key in 1966.

Has 36 pages in color.
Comic book: Profile

A close look at a CMDI file

- A toy example: http://hdl.handle.net/1839/00-DOCS.CLARIN.EU-102

- A corpus description: http://hdl.handle.net/1839/00-DOCS.CLARIN.EU-103

Data Categories
Data categories in practice

- A data category provides a definition for a CMDI element (or component)
  - to avoid ambiguity
  - to enable semantic mapping
- Data categories are stored in the ISOcat registry: http://www.isocat.org
- The Component Registry is connected to ISOcat
- Metadata browsing applications (like the VLO) are using these definitions

The big picture

Process overview

- Check the Component registry:
  - Any profile that fits your needs?
  - If not:
    - Any component that fits your needs?
    - If not:
      - Create your own component!
      - Looking for a data category that is not there?
        - Create a new data category!
      - Combine components together in a profile

Metadata creation

- Manually:
  - Arbil or your XML-editor
  - Select the profile/XSD that suits your needs
  - Create metadata instances
- Automatically: via e.g. a web service or script
- Conversion:
  - plenty of XSLT stylesheets available: http://www.clarin.eu/faq-page/274
  - DC/OLAC, TEI header, MetaShare, IMDI, MODS, Paradisec
What is out there?

- More than 20 Language Resource repositories are using CMDI: http://centres.clarin.eu/
- About 560,000 metadata records: http://clarin.eu/vlo
- About 150 profiles and 860 components: http://catalog.clarin.eu/ds/ComponentRegistry
- About 1100 metadata data categories: http://www.isocat.org/

Compatible software

- repositories (Fedora, Dspace, LAT) & processing and creation tools – see http://clarin.eu/cmdi
- exploration & searching:
  - facet-supported full-text search: VLO – http://clarin.eu/vlo
  - hierarchical browsing + fine-grained search: YAMS – http://clarin.eu/yams

Conclusions

- Component metadata ensures flexibility while maintaining technical and semantic interoperability
- It comes with out-of-the-box conversion methods for existing schemas
- There is a whole software stack available for the production and usage of CMDI
- More information: http://clarin.eu/cmdi
Persistent Identifiers
Using Handles for Identification and Retrieval of Linguistic Resources

Thomas Eckart
Natural Language Processing Group
Institut of Computer Science, University Leipzig
teckart@informatik.uni-leipzig.de

Motivation 1

- Long-term preservation and availability of resources
- Long-term availability vs. short-lived URLs (“Link rot“)
- Reference to a resource independent of its physical storage location or means of access
- *Persistent Identifiers (PIDs) can be treated as the incarnation of the resources and not as one of their many copies that may exist.*

Motivation 2

General approach:
- Additional layer on top of resource locators

Agenda

1) Motivation
2) Potential Criteria
3) Existing Approaches & Evaluation
4) Persistent Identifiers and Granularity in the Handle System
5) Demo Handle System/EPIC API
6) Usage in CLARIN
   - 6.1) referenced objects
   - 6.2) content negotiation
   - 6.3) versioning
7) Examples
   - 7.1) CLARIN Centre Leipzig
   - 7.2) Demo Resource Access & Retrieval in CLARIN
Persistence and Uniqueness

Contexts of References
- Resolving in different contexts (web-sites, papers etc.)
- Allows rewriting into URL

Resources Granularity
- Supporting reference to collections & fragments
- Versioning

Criteria 1

Copies
- Loadbalancing
- Long-term archiving

Compatibility and Standards
- Compatibility to URI standard of IETF

Additional Information
- Allows descriptive metadata

Criteria 2

(No) Semantics

Fragment Addressing
- Resolution supports Fragment Identifier
- "Pass-through" mechanism

Performance/Robustness/Availability
- Resolution as potential bottleneck
- Redundancy/Caching mechanisms
- Long term support

Criteria 3

Security
- Authorization for write access

Independence/Openness
- Influence on policies
- Open and free software

Costs
- No correlation to number of issued PIDs

Criteria 4
Approaches

- Uniform Resource Name **URN**
- **Handle** System
- Digital Object Identifier **DOI**
  - Uses Handle System
- Archival Resource Key **ARK**
  - ...

### Handle System – Introduction

- General-purpose identifier resolution system
  - Foundation for DOI
- Developed by the *Corporation for National Research Initiatives* (CNRI)
- Syntax: **PREFIX/SUFFIX**
  - **PREFIX**: Naming authority (→ Resolver)
  - **SUFFIX**: Local name

<table>
<thead>
<tr>
<th>Criteria</th>
<th>URN</th>
<th>Handle</th>
<th>DOI</th>
<th>ARK</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Copies</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Standards</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Additional Data</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Semantics</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Fragments</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Performance/Robustness</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Security</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Independence</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
</tr>
<tr>
<td>Spreading</td>
<td>-</td>
<td>0</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>Costs</td>
<td>+</td>
<td>0</td>
<td>-</td>
<td>+</td>
</tr>
</tbody>
</table>

* Specific resolver

Persistency and unique identifiers (Daan Broeder, Malte Dreyer, Marc Kemps-Snijders, Andreas Witt, Marc Kupietz, Peter Wittenburg, 2009, [http://hdl.handle.net/1839/00-DOCS.CLARIN.EU-30](http://hdl.handle.net/1839/00-DOCS.CLARIN.EU-30))
**Example:**
- European Persistent Identifier Consortium (EPIC)
- Handle: 11022/0000-0000-2099-F
  - hdl:11022/0000-0000-2099-F
- Resolver:
  - http://hdl.handle.net
  - http://hdl.handle.net/11022/0000-0000-2099-F

**Handle System – Part/Fragment Identifier**

- Syntax: Prefix/Suffix@PART_IDENTIFIER
- Part Identifier without predefined structure
- Example:

<table>
<thead>
<tr>
<th>Solution 1</th>
<th>Solution 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>11022/0000-1111-F</td>
<td>11022/0000-1111-F</td>
</tr>
<tr>
<td>11022/0000-2222-F</td>
<td>11022/0000-2222-F</td>
</tr>
<tr>
<td>11022/0000-3333-F</td>
<td>11022/0000-2222-F@ActorLaura</td>
</tr>
</tbody>
</table>

10 PIDs 4 PIDs

**Persistent Identifier and Granularity**

- Most resources are structured
- PIDs may reflect structure
- For many resources this could mean a very large amount of PIDs

**Full responsibility of the PID's owner**

**More examples**

- Offset in audio/video files
  11022/0000-1111-F@offset=3:00
- Display hints
  11022/0000-1111-F@version=html
  ...

...
Example:

PID 11858/00-229C-0000-0001-B06F-3@type=dataprovier&id=2
will be rewritten to
1858/00-229C-0000-0001-B06F-3?type=dataprovier&id=2

Handle System – Part/Fragment Identifier

Creating new PIDs vs. Part Identifiers

Rules of thumb:

• New PID, when
  − Associated with "complete" content
  − Resource is "autonomous"
  − Should be citable on its own

• Part Identifier, especially when
  − Only used in larger context

EPIC API

Demo API EPIC

http://handle.gwdg.de:8080/pidservice/ (v1)

GWDG PID Handle Service

View Handle

PID / Handle:

- jump to proxy view of data
- jump via proxy to resource
- jump directly to metadata
- view XML instead of HTML (no effect for 'jump')

Enter PID (Handle) above and
Submit
EPIC API

Demo API EPIC – Creation of new Handle

Create Basic Handle


Suffix: [user defined, optional]

Confirm in XML: [default: HTML]

Enter URL above and [Create Basic Handle]

Create Verbose Handle

EPIC API

Demo API EPIC

http://hdl.handle.net/(Handle Resolver)

http://hdl.handle.net/11858/00-229C-0000-0023-682A-B?noredirect

http://hdl.handle.net/11858/00-229C-0000-0023-682A-B

Handle System

Resolve a Handle and View the Values

EPIC API

New PID Handle created:

11858/00-229C-0000-0023-682A-B

User details

User: 229C
Institution: 229C
Contact: email: clarin@informatik.uni-leipzig.de
Can create PID: yes
Can update PID: yes

[Back to input form]

CLARIN Recommendations (Excerpt)

- It is recommended to all (potential) CLARIN centers to get acquainted with the requirements and solutions for creating and maintaining PIDs.
- We recommend taking care of the PID requirements in all CLARIN related software developments.
- We recommend establishing a CLARIN PID service that is independent of any commercial business model.
Centres need to associate PIDs with their metadata records

Non-metadata files should receive a PID or a PID in combination with a part identifier, if these files:
  * are accessible via internet
  * are considered to be stable by the data provider
  * are considered to be worth to be accessed directly (not via metadata records) by the data provider

PIDs should be suitable for both human and machine interpretation

Webservices make use of HTTP-accept header

Resources evolve over time

Problem: long-term accessibility

Changes in published resource require new PID

Compromise: identifier for general resource and each version of this resource

Example: CLARIN-D Centre Leipzig

Corpus structure & PIDs
  * Data Provider
    - Data Source (=Document)
      * Sentence
Example: CLARIN-D Centre Leipzig

CLARIN Walkthrough

Demo Persistent Identifier in the CLARIN infrastructure

CLARIN Walkthrough

Metadata Search Virtual Language Observatory (http://catalog.clarin.eu)
Federated Content Search
(http://weblicht.sfs.uni-tuebingen.de/Aggregator)

Language resource management -- Persistent identification and sustainable access (ISO 24619:2011)
- Persistent Identifiers in CLARIN
  - https://www.clarin.eu/content/goals-and-requirements-pid-systems
- EPIC
  - http://www.pidconsortium.eu
Summary

- Persistent Identifiers are long-lasting references (in contrast to URLs) and can form a basis for a stable resource infrastructure.
- Several systems exist (Handle, DOI, ARK, URN, PURL …), all with different features.
- Usage of PIDs: clear policy and maintenance efforts.
- Your institution may already have a policy about using PIDs.
Web Services
Architecture and Examples

Christoph Draxler
Thomas Kisler
{draxler|kisler}@phonetik.uni-muenchen.de

W3C Definition

• A Web Service is a software system
  • designed to support *interoperable* machine-to-machine
    interaction
  • over a network.

• Its interface is *described* in a machine-processable
  format.

• Other systems interact with the Web service in a
  manner prescribed by its description using *messages*
  • typically conveyed using *http*
  • with a *serialization* in conjunction with other Web-related
    standards.

Agenda

• Web Services Definition
• Approaches to Web Services
  • Technology
  • Discussion
• Requirements and Commitment
• Web Service Examples
• Chaining

Approaches

• REST-compliant Web services
  • manipulate XML representations of Web
    resources using a uniform set of stateless
    operations
  • e.g. crowdsourcing, online shopping,…

• Other Web services
  • perform arbitrary operations
  • e.g. cloud computing, server-based
    processing,…
Common concepts

- URIs identify resources
  - Uniform Resource Identifier
    - scheme:hier-part[?query][#fragment]
- W3C protocols define operations
  - http
    - hyper text transfer protocol
  - SOAP
    - network protocol for remote procedure calls
    - using XML-formatted messages

Technology

- REST – Representational State Transfer
  - resource-oriented *architectural style*
  - based on stateless http methods
    - get, post, put, delete
    - and serialized resource representations

REST style

- 4 http methods mapped to 4 database operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Create</th>
<th>Read/Retrieve</th>
<th>Update</th>
<th>Delete/Destroy</th>
</tr>
</thead>
<tbody>
<tr>
<td>http</td>
<td>POST</td>
<td>GET</td>
<td>PUT</td>
<td>DELETE</td>
</tr>
<tr>
<td>SQL</td>
<td>CREATE</td>
<td>SELECT</td>
<td>UPDATE</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

- XML representation of resources
  - increasingly, JSON is used

Web Services: Pros

- User view
  - immediate and easy access
  - no need for local software installation
  - always newest version
  - take advantage of provider’s processing power
- Provider view
  - full control of runtime environment (software, hardware)
  - easy monitoring, logging
  - immediate availability of updates
  - set the service up once, use it everywhere
    - → with [rd]ecent browser
Discussion: Cons

- User view
  - only one version available
  - network access necessary
  - processing speed dependent on server
  - data has to be given away

- Provider view
  - immediate availability of updates
  - single point of failure
  - difficult to estimate usage (amount, regularity)

SOA Advantages

- Loosely coupled functionalities
- Service providers are only responsible for the service and software
  - they know and
  - they need anyway
- Modular setup, easy to be reused
- Well-specified interfaces
  - WADL, CMDI, etc.

Summary

- In practice
  - most users want a system that simply works
  - humanities researchers should not be burdened with the installation of software
  - increased use of laptops → servers more suited for number/text crunching
- New opportunities
  - web services promise access to rich set of software the researcher might not have otherwise

Requirements & Commitment

- A web service provider must have
  - a service suitable for online usage
  - plus
    - developer(s)
      - to create and maintain the web service(s)
    - administrator
      - to keep it up and running
    - servers that are accessible
      - 24/7 with (very) low downtime
      - over a longer time-frame (years)
Web Service Architecture

- User client
- Other service
- Workflow engine
- Web service interface
- Local software
- Local data
- Remote software
- Remote data
- WSDL/WADL

Web Service Description

- WSDL & WADL
  - Web Service Description Language
  - Web Application Description Language
  - Technical description

- CMDI – Component Meta Data Infrastructure
  - Semantic description of web services
  - Well-defined reusable building blocks
  - Flexible framework

WADL example

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2008/02" xmlns:jersey="http://jersey.dev.java.net/" jersey:generated="Jersey 1.1.5.1 83/28/2010 02:13 GMT">
  <resource paths="">
    <resource path="/WADLExample">.../resource</resource>
    <resource path="/WADLExample/*">.../resource</resource>
  </resource>
  <method name="GET" id="wadlWadlBasic">
    <representation mediaType="text/html; charset=UTF-8"/>
    
  </method>
</application>
```

CMDI Example

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://wadl.dev.java.net/2008/02" xmlns:jersey="http://jersey.dev.java.net/" jersey:generated="Jersey 1.1.5.1 83/28/2010 02:13 GMT">
  <resource paths="">
    <resource path="/CMDIExample">.../resource</resource>
    <resource path="/CMDIExample/*">.../resource</resource>
  </resource>
  <method name="GET" id="cmdiWadlBasic">
    <representation mediaType="text/html; charset=UTF-8"/>
    
  </method>
</application>
```
Web Services in CLARIN

- Distributed centres provide resources and services
  - currently, existing tools are being converted to web services
  - mostly RPC using RESTful style
- Single sign-on and authentication
- Workflow engines to chain web services
  - Tool Chain Format ensures compatibility

CLARIN web services

- Virtual Language Observatory lists services
  - www.clarin.eu/content/virtual-language-observatory
- Linguistic tools
  - stemmer, parser
  - tree-bank explorer
- Speech technology
  - text-to-Speech
  - automatic segmentation and labeling
- Visualization, format converters...

Examples

- BAS Web Services help
  - online help function
    - curl -X GET clarin.phonetik.uni-muenchen.de/BASWebServices/services/help
- BAS WebMAUS
  - automatic phonemic segmentation and labelling
- HZSK transcription format converter
  - convert annotation files

Example BAS help function
WebMAUS

- Substantial speedup of workflow
- Less qualified work needed
- Easy to use
  - GUI in browser
  - comfortable multi-file options

clarin.phonetik.uni-muenchen.de/BASWebServices
WebMAUS in Terminal

- `curl -v -X POST -H 'content-type: multipart/form-data' -F SIGNAL=@German.wav -F STARTWORD=0 -F TEXT=@German.txt 'http://clarin.phonetik.uni-muenchen.de/BASWebServices/services/runMAUSBasic'

- `<WebServiceResponseLink>
  <success>true</success>
  <downloadLink>http://clarin.phonetik.uni-muenchen.de:80/…/German.TextGrid</downloadLink>
  <output>
    /usr/local/bin/maus OUTFORMAT=TextGrid
    OUT=/usr/share/…/German.TextGrid
  </output>
  </WebServiceResponseLink>`

WebMAUS sequence diagram

WebMAUS in ELAN

Transcription Format Converter

Webservice for conversion of transcription formats

Sample call:

```
wget --post-file=input.exb --header='Content-Type: text/eb+xml' 'http://virt-fedora.multilingua.uni-hamburg.de:8080/converter/resources/convertExb?to=eaf' -0 output.eaf
```
Tool Chaining

- Enhance workflow by chaining services
  - output of one service is input to the next
- Web service chains
  - not restricted to local software
  - alternative services may be explored
  - automated processing becomes feasible

WebLicht

- Web Licht
  - automatic orchestration and execution environment
  - incremental annotation of corpora
- Service Oriented Architecture
  - web interface for interactive access
  - based on Text Corpus Format (TCF)

Summary

- Web services will change the way researchers work
  - access to services without software installation
  - more and bigger data
- Service providers need to prepare
  - convert tools to services
  - long-term commitment is mandatory
Repositories

Online Speech and Language Resources, Tutorial

Daniel Jettka, daniel.jettka@uni-hamburg.de
Hamburger Zentrum für Sprachkorpora, Universität Hamburg

1. Introduction

What is a digital repository?
Existing repository solutions

2. The Fedora repository system

Data storage, long-term accessibility, version management, access control, interfaces, Islandora software framework

3. Repositories in CLARIN

Federated login, federated content search, metadata harvesting, assessment
Repository demo: ASV Uni Leipzig, HZSK Uni Hamburg

Motivation: long-term storage and availability of digital resources

Bob Kahn: “repository is a network accessible storage to store objects for later access”

JISC: A digital repository is a managed, persistent way of making research, learning and teaching content with continuing value discoverable and accessible. Repositories can be subject or institutional in their focus. Putting content into an institutional repository enables staff and institutions to manage and preserve it, and therefore derive maximum value from it. A repository can support research, learning, and administrative processes.

(Wittenburg, 2011)

Motivation: long-term storage and availability of digital resources

Forrester Research: Knowledge workers spend 40% of their time trying to find information and 70% of that time is spent recreating information that cannot be found. A digital repository offering refined categorisation and search tools that help locate information quickly provides quantifiable savings in terms of time and resources.

(Wittenburg, 2011)
**ESFRI: key objectives**

- Availability
- Permanency
- Quality
- Right of Use
- Interoperability

(cf. ESFRI Position Paper, 2009)

---

**Existing repository solutions**

- Ready-made solutions (often no application logic):
  - D-SPACE, ePrints, eSciDoc, LAMUS
- Toolkits:
  - Fedora
- Grid and database solutions:
  - IRODS, MySQL, Postgres, Xbase, eXist, etc
- Commercial solutions:
  - ORACLE etc, CMS, ArchivalWare, DigiTool, VITAL

(cf. Wittenburg, 2011)

---

**Comparisons**

- Marill/Luczak (2009): DAITSS, DSpace, EPrints, Fedora, Greenstone, Keystone DLS, ArchivalWare, CONTENTdm, DigiTool, VITAL

---

**The Fedora repository system**
Flexible Extensible Digital Object Repository Architecture
Free, open-source, community project
Use of open standards and protocols:
  - DC, RDF, XACML, XML
  - OAI-PMH, LDAP
  - SOAP & REST web services
Foundation for building variety of information management schemes for different use cases – not full solution for specific use case

Fedora Commons

Prerequisites:
- Java SE Development Kit (JDK)
- Database (MySQL, Oracle, PostgreSQL, or Microsoft SQL Server)
- Application Server (any that implements Servlet 2.5/JSP 2.1 or higher; included: Tomcat)
- (Maven 2: for building from source)

Fedora Commons

Storage of any sort of digital content in any format (e.g. documents, videos, images, metadata)
Storage of relationships between content items
Also possible:
- Storage of metadata and relationships for content which is held by another organization or system

Fedora Commons

Fedora Digital Object Model

Four types of Digital Objects:
- Data Object
- Service Definition Object
- Service Deployment Object
- Content Model Object
Fedora Commons

Datastreams

- Persistent ID (PID)
- Object Properties
- Relations (RELS
-ent)
- Public Core
- Audit Trail
- Datastream 1
- Datastream N

Digital Object Identifier
System Properties (manage and track the object)
Reserved Datastreams (key object metadata)
Datastreams (aggregates content items)

https://wiki.duraspace.org/display/FEDORA37/Fedora+Digital+Object+Model

Fedora's archival and preservation capabilities include:
- XML for Fedora objects (preserved at ingest, during storage, and at export)
- Object to Object Relationships: can be stored via metadata included in objects (RDF) → hierarchical relationships for related objects
- Content Versioning & event history: audit trail of objects (optional)
- Support of date-time stamped requests

Access control

Authentication
- Basic: list of users in fedora-users.xml
- LDAP
- Java Authentication and Authorization Service (JAAS)

Authorization
- Simple servlet container authentication – can do everything
- Basic access roles authorizations – mapping onto preconfigured roles
- XACML policies

https://wiki.duraspace.org/pages/viewpage.action?pageId=28181276

OAI-PMH

Open Archives Initiative Protocol for Metadata Harvesting:
- Low-barrier mechanism for repository interoperability
- Data Providers: repositories that expose structured metadata via OAI-PMH, e.g. CLARIN repositories - CMDI
- Service Providers: make OAI-PMH service requests to harvest that metadata, e.g. Virtual Language Observatory
- OAI-PMH: set of six verbs or services invoked within HTTP (GetRecord, Identify, ListIdentifiers, ListMetadataFormats, ListRecords, ListSets)

http://www.openarchives.org/pmh/
Fedora OAI Provider

Fedora OAI Provider Service:

- Based on Proai
- Supports any metadata format available through the Fedora repository via a datastream or dissemination
- Supports sets that are expressed as RDF relationships in digital objects' RELS-EXT datastreams
- Runs as webapp in any servlet container, acting as web service client to Fedora
- Caches content of the Fedora disseminations and datastreams intended to be exposed as OAI records

http://fedora-commons.org/download/2.2/services/oaiprovider/doc/
User web interface

Fedora Commons: Summary

+ Stable and approved in many projects
+ Flexible way of storing data
+ Everything on board or can be added (PID handling, OAI-PMH)
+ Extensive programming API in RESTstyle

- Setup, data modelling, training of repository managers can become time-consuming
- No user friendly interface (→ eSciDoc, DSpace)
- Can only be used for storing file-based data: no access to databases


Islandora

- Open-source software framework
- Focus on collaborative management, and discovery of digital assets using a best-practices framework
- Built on the basis of Drupal, Fedora, and Solr

Islandora

- Support for any file type (via Fedora repository system)
- Multi-language and functionality support via Drupal
- Modular Solution Pack framework for defining specific data models and associated behaviors (e.g. for audio, PDF, images, books)
- Formbuilder module for the creation of a data-entry/editing form for any XML schema

http://islandora.ca/
**Support for semantic ontologies and the creation of relationships between objects**

**Flexible faceted search driven by Apache Solr**

**Micro service-based workflows for automating the transformation of assets**

**Editorial workflows for approving submissions to the repository**

---

**Intermediate conclusion**

„Ultimately, the institution must evaluate its collections, technical expertise, and research distribution strategy in order to choose the platform that will best support its research goals“ (UNESCO, 2014)

**Features of CLARIN repositories**

- In many cases it might make sense to integrate data into existing repositories
- CLARIN centres have competences in several areas & have well-defined policies to host data
- Overview of centres and basic technology: [http://centerregistry-clarin.esc.rzg.mpg.de/](http://centerregistry-clarin.esc.rzg.mpg.de/)

---

**Metadata harvesting**

Harvesting of CMDI metadata from repositories
Goal: easier access to password-protected resources for academic users

- academic users should be able to login with their existing institutional credentials
- user stores from universities and academic institutions ("Identity Providers") are connected to password-protected web applications ("Service Providers") - connection based on mutual trust

http://www.clarin.eu/node/3788

Single sign-on solution, mainly used for web-application security

- based on SAML; session used to manage authentication state
- software components are implementation of the SAML protocols and bindings

http://www.switch.ch/aai/demo/2/resources/simple_complete.png

standard XML-based protocol for search queries, utilizing CQL - Contextual Query Language (a standard syntax for representing queries)

- three main operations: Explain, Scan, SearchRetrieve
- extension of SRU/CQL-protocol as common harmonized interface (lingua franca) that individual repositories willing to join the federated search have to implement
- individual repositories implement FCS-interface as "endpoints"

http://www.loc.gov/standards/sru/

http://www.clarin.eu/content/federated-content-search
Data seal of approval

http://www.datasealofapproval.org

Towards sustainable and trusted data repositories

There are 9 guidelines that together determine whether your data repository qualifies for the Data Seal of Approval.

Part of CLARIN centre assessment procedure

Samples of CLARIN repositories

Repository CLARIN-D Centre Leipzig

Introduction to the Repository

The CLARIN-D repository at the University of Leipzig offers long-term preservation of digital resources, along with their descriptive metadata. The mission of the repository is to ensure the availability and long-term preservation of resources to preserve knowledge gained in research, to aid the transfer of knowledge into new contexts, and to integrate new methods and resources into university curricula.

CLARIN-D is developing a digital infrastructure for language-centred research in the social sciences and humanities. The main function of the CLARIN-D service centre is to provide relevant, useful data and tools in an integrated, interoperable and scalable way. CLARIN-D will sell the infrastructure out in close collaboration with expert scholars in the humanities and social sciences, to ensure that it meets the needs of users in a systematic and easily accessible way. Integration of the repository into the national and international CLARIN-D infrastructure gives it wide exposure, increasing the likelihood that the resources will be used and further developed beyond the lifetime of the projects in which they were developed.

Among the resources currently available in the Leipzig repository are a set of corpora of the Leipzig Corpora Collection (LCC), based on newspaper, Wikipedia and Web text. Furthermore several NLP-based workflows are provided for a variety of different NLP-relevant tasks.

http://clarin.informatik.uni-leipzig.de/repo/

https://www.corpora.uni-hamburg.de/repository
Recapitulation

(1) Digital repositories
   Purpose, key objectives, existing solutions

(2) The Fedora repository system
   Features, interfaces, pros & cons, Islandora as front-end

(3) Repositories in CLARIN
   Metadata harvesting, federated login, federated search, assessment
   Repository demo: ASV/University Leipzig, HZSK/University Hamburg

References

Elbers, W. (2011). Shibboleth @ the MPI. Presentation at CLARIN-D Tutorial, AAI and PIDs, MPI for Psycholinguistics, Nijmegen, 08.09.2011.


