A prototype infrastructure for DSpin

- DSpin/Clarin, LLS, prototype, toolwrappers, roadmap -

Volker Boehlke
University of Leipzig
Some facts about me

- iba Consulting Gesellschaft (programmer & consultant)
  - reporting software for energy trade
  - real estate asset management

- University of Leipzig
  - member of the scientific staff of the NLP group
  - DSpin/Clarin project (WS, Infrastructure)
Agenda

- DSpin/Clarin - a small introduction
- from LLS to a pragmatic DSpin prototype
- the toolwrapper proposal
- roadmap
Overview:

- DSpin is the german version/subproject to CLARIN
- CLARIN: http://www.clarin.eu/

The CLARIN project is a large-scale pan-European collaborative effort to create, coordinate and make language resources and technology available and readily useable.

- D-SPIN: http://www.sfs.uni-tuebingen.de/dspin/
- contribute german ressources (data+tools) to the federation
- Aim: define data formats and interfaces, clear up legal issues, cooperation with e-humanity projects, create training material: documentation, tutorials
Imagine...

- a vast amount of resources and tools being available through one European infrastructure
- no headache when configuring tools for your purpose
  - in many projects (bachelor/master students, e-humanity projects)
    much time is spent in finding/configuring resources/tools
  => less configuration work; more time for the actual problem at hand
- comparable results: once established workflows are available for a long period of time (even after years, the exact same version of a tool/resource maybe used in a also “stored” workflow)
- a federation of resource centers maintains the needed infrastructure
Non technical aspects

- legal issues:
  - WS allow (easy) automatic harvesting of huge data-collections
  - Who owns the content? Who is allowed to access it?
  - What are users allowed to do?
    - Build own services based on your services?
    - Sell products using these services?
  - Who pays for the infrastructure?
    - batch processing comes with high costs (CPU/Memory, Traffic)
    - who maintains the services (answer questions, support, ...)
resources and tools in DSpin/Clarin

- the language resources and tools survey:
  http://www.clarin.eu/summary-wp5-language-resources-and-technology-overview

- currently available: 768 resources and 132 tools

The Clarin community is open! Feel free to add your resources/tools to the pool!
resources and tools in Dspin/Clarin II

<table>
<thead>
<tr>
<th>Resources overview</th>
<th>Clarin</th>
<th>Tools overview</th>
<th>Clarin</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Wortschatz</th>
<th>Written Corpus</th>
<th></th>
<th>Collected from newspaper texts, webcrawling, etc.: words (+frequency), cooccurrences (+graph), left/right Germany neighbours, example sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Web Service</td>
<td></td>
<td>University of Leipzig • Department of Computer Science • NLP Group</td>
</tr>
<tr>
<td></td>
<td>Albanian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bulgarian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Catalan</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Chinese</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Croatian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Czech</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Danish</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dutch</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>English</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Esperanto</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Estonian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Finnish</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>French</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>German</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hungarian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Icelandic</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Indonesian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Italian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Japanese</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Korean</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Latin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Latvian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Lithuanian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Malay</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Norwegian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Occitan</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Romanian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Slovak</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Slovenian</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spanish</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sundanese</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Swedish</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Tagalog</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Turkish</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vietnamese</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
resou... Resources overview | Clarin | Tools overview | Clarin | Resources overview | Clarin | Tools overview | Clarin

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Country</th>
<th>Description</th>
<th>Organisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fine-Tracker</td>
<td>other</td>
<td>Netherlands (the)</td>
<td>Computational model of human word recognition; Fine-phonetic detail</td>
<td>Centre for Language and Speech Technology, Radboud University Nijmegen, The Netherlands</td>
</tr>
<tr>
<td>Functional Morphology</td>
<td>NLP development aid</td>
<td>Sweden</td>
<td>Functional Morphology is a development environment for computational morphologies.</td>
<td>Språkbanken</td>
</tr>
<tr>
<td>Glossa</td>
<td>evaluation tool</td>
<td>Norway</td>
<td>Search and post-processing tool</td>
<td>University of Oslo</td>
</tr>
<tr>
<td>Helsinki Finite State</td>
<td>NLP development aid</td>
<td>Finland</td>
<td>The Helsinki Finite-State Transducer software is intended for the implementation of morphological analysers and other tools which are based on weighted and unweighted finite-state transducer technology. The feasibility of the HFST toolkit has been demonstrated by full-fledged open source implementations of Finnish, Swedish, English, French and Northern Sami lexicons.</td>
<td>Department of General Linguistics, University of Helsinki</td>
</tr>
<tr>
<td>HTML annotation tool</td>
<td>annotation tool</td>
<td>Hungary</td>
<td>WYSIWYG HTML semantic annotation tool which reserves the DOM structure.</td>
<td>University of Szeged</td>
</tr>
<tr>
<td>Hunalign - sentence</td>
<td>single tool</td>
<td>Hungary</td>
<td>Hunalign is a powerful free sentence level aligner for building parallel corpora. Its input is tokenized and sentence-segmented text in two languages.</td>
<td>Budapest Technical University Media Research Centre</td>
</tr>
<tr>
<td>Level aligner</td>
<td>single tool</td>
<td>Hungary</td>
<td>Hunalign is a powerful free sentence level aligner for building parallel corpora. Its input is tokenized and sentence-segmented text in two languages.</td>
<td>Budapest Technical University Media Research Centre</td>
</tr>
<tr>
<td>Hungarian Lexical and</td>
<td>annotation tool</td>
<td>Hungary</td>
<td>NooJ is a linguistic development environment that includes large-coverage dictionaries and grammars, and parses corpora in real time. The large-coverage lexical resources (morphological and syntactic grammars) for Hungarian might be applied to texts in order to locate morphological, lexical and syntactic patterns and tag simple and compound words.</td>
<td>Dept. of Language Technology, Research Institute for Linguistics</td>
</tr>
<tr>
<td>Syntactic Resources for</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NooJ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hunmorph - morphological</td>
<td>single tool</td>
<td>Hungary</td>
<td>Hunmorph is an open source tool and programming library for spell-checking, stemming and morphological analysing of agglutinative, german and other languages.</td>
<td>Budapest Technical University Media Research Centre</td>
</tr>
<tr>
<td>Hunner - named entity</td>
<td>annotation tool</td>
<td>Hungary</td>
<td>Hungarian named entity recognition with a maximum entropy approach</td>
<td>Budapest Technical University Media Research Centre</td>
</tr>
<tr>
<td>recognizer for Hungarian</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LeipzigLinguisticServices - repository:

- relational database

- used fields are: ID, webservice-type, -name, algorithm, inputFields, description, status, authorizationLevel

<table>
<thead>
<tr>
<th>Id</th>
<th>WebServiceType</th>
<th>Algorithmus</th>
<th>InputFields</th>
<th>Name</th>
<th>Description</th>
<th>Status</th>
<th>Authorize</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLSelectType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>ServiceOverview</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>FREE</td>
</tr>
<tr>
<td>4</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLSelectType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>Cooccurrences</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>FREE</td>
</tr>
<tr>
<td>5</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLSelectType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>Grundform</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>FREE</td>
</tr>
<tr>
<td>8</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLModifyDBType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>AddWebServiceUser</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>FREE</td>
</tr>
<tr>
<td>9</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLModifyDBType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>AddCorpus</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>INTERN</td>
</tr>
<tr>
<td>12</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLModifyDBType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>DeleteWebServiceUser</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>INTERN</td>
</tr>
<tr>
<td>11</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.MySQLModifyDBType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>DeleteWebService</td>
<td>&lt;MEMO&gt;</td>
<td>ACTIVE</td>
<td>INTERN</td>
</tr>
<tr>
<td>0</td>
<td>de.uni_leipzig.wortschatz.webservice.webserviceTypes.FileUploadTestType</td>
<td>&lt;MEMO&gt;</td>
<td>&lt;MEMO&gt;</td>
<td>UploadTest</td>
<td>&lt;MEMO&gt;</td>
<td>INACTIVE</td>
<td>FREE</td>
</tr>
</tbody>
</table>

Ex:Edit Text

```
select v.wort_bin as Wort, wg.wortart AS Wortart from wortliste w, wortliste v, wortart_grundform wg where wg.wort_nr=w.wort_nr and grf_wort_nr=v.wort_nr and v.wort_bin=?
```
LeipzigLinguisticServices - services

- SOAP based
- Apache Axis framework was used
- data is transferred relying mostly on “string” (matrix) because many compatibility issues were observed over the years (we know of bindings to Java, .NET, Perl)
  - signed and unsigned datatypes in different languages
  - floating point numbers (0.00314 vs. 3.14E-3 vs. 3.14e-3)
  - date
- problems with services/tools: need too much ressources (memory, ...)
- problems with timeouts (planned to use asynchronous transfer in the future)
TMS schedule - more about LLS

25.03.2009; 13:00

LLS – A 4 Years Summary of Providing Linguistic Web Services
Marco Büchler (University of Leipzig, eAqua Project), Gerhard Heyer (University of Leipzig)
LeipzigLinguisticServices – what we can learn

simple pragmatic solution: transfer XML-Documents and encode the information
doesn't seem like a bad idea (REST vs SOAP discussion):
- simple services, very easy to describe (XML-Docs are transferred)
- we use XML to describe and code our datastructures; easier for tool maintainers
- we are able to use domain knowledge or existing standards to define interchange formats
- address QoS issues from the start and on all levels => scalability
- develop a box of tools: solutions for the different problems that need to be addressed
- a solution for simple applications maybe developed by tool maintainers
- more complex applications (especially e-humanity projects) are solved with the help of Clarin consultants (ressource center staff) => just like SAP does it
the technical challenge in short

- central issue: How to integrate completely different tools and ressources based on various standards into one infrastructure?
  
  => virtual collections, process chaining, authentification, quality of service, ....

- more precisely: What needs to be specified in order to integrate those different elements into one infrastructure:
  
  - standards for data encoding (“data pieces”, like utf8-text)
  - general data interchange formats (headers/envelopes/structures)
  - possibility to formally specify the used standards and formats
  - “informal” meta-data (additional information for the human user)
DSpin

DSpin

- bottom-up (start with real appl., limited time & resources)
- Leipzig, Stuttgart, Tübingen
- idea: pragmatic, prototype solution

4 layers:

- process chain tool
- “common” DSpin/Clarin webservice:
  - authentication, QoS, repository/metadata
- toolwrapper:
  - standardised input-/output formats, unified behaviour, used by a common clarin/dspin webservice and is a WS itself
- concrete ressource / tool
Overview

process chain builder/validator

DSpin webservice

toolwrapper A

resource A

DSpin webservice

toolwrapper B

tool format 1

tool b

service repository
layers explained

- on toolwrapper-layer (REST?) webservices with standardized functions and XML-formats (matching the “problem”) are used
- a common WS operates on top of a toolwrapper; typical problems like authentification, QoS etc. are solved; it only has to be configured for every use case (which toolwrapper to use; definition of QoS boundaries etc.) => maintained by the resource center staff

=> classic SOA architecture; it get's interesting (and challenging) when we talk about formats and (automatic) process chain building (“orchestration”)

D-Spin
central issue:
Not only provide access to different resources via webservices, but formalize those services in a way that allows orchestration without much effort. Note: Orchestration is done (configured) by “technically (WS etc.) unexperienced” users! => validation/invocation has to be done autom.

a small example; q/a (dialog style) to a tool-service:
- question: “Which data is needed?“
- answer: „tokens encoded in utf8 and information about language in iso-639-3.“
- question: “Which data is produced.”
- answer: „tokens encoded in utf8 annotated with POS-tags in STTS“.
the same information expressed “more formally” in XML:

```xml
<input>
  <type>token</type>
  <attributes>
    <lang>iso-639-3</lang>
    <encoding>utf8</encoding>
  </attributes>
</input>

<output>
  <type>token</type>
  <attributes>
    <pos>STTS</pos>
  </attributes>
</output>
```
It's totally clear that we need some kind of vocabulary repository / standards, which stores/specify the formal vocabulary (lang, encoding, POS) and possible values/standards (utf8, STTS).

An automatic process chain building tool may:
- decide if a chain is executable; formal: is all data needed on every level and step of the process chain available? (at least formally!)
- infer the steps that need to be done to use tool B on resource A ("sentence"-Resource - .? - POS-Tagger)
- detect conflicts and ask the user in order to resolve them
What happened up to now?

- we identified the problem and worked on extracting the essentially needed information

- plan: build a very small prototype at first
  - wire the first services of several different partners
  - aim: gather some real life experience (especially concerning the end-user)

- definition of a first, very abstract data exchange format for chaining

- criticism from Helmut Schmid (Stuttgart): too hard to handle from a tool-developers point of view; his idea: toolwrappers using a small number of formats enriched by formalised meta data (for process chaining!)
toolwrapper example (output of a tokenizer):

```xml
<TextCorpus lang="de" encoding="latin1" text="yes" tokens="yes">
  <text>Peter aß eine Pizza. Sie schmeckte ihm.</text>
  <tokens>
    <token ID="t1" start="1" end="5">Peter</token>
    <token ID="t2" start="7" end="8">aß</token>
    <token ID="t3" start="10" end="13">eine</token>
    <token ID="t4" start="15" end="19">Pizza</token>
    <token ID="t5" start="20" end="20">.</token>
    <token ID="t6" start="22" end="24">Sie</token>
    <token ID="t7" start="26" end="34">schmeckte</token>
    <token ID="t8" start="36" end="38">ihm</token>
    <token ID="t9" start="39" end="39">.</token>
  </tokens>
</TextCorpus>
```
toolwrapper 1:

- 3 parts:
  - informal meta-data (dates, sources, author etc.)
  - formal meta-data/header (standards, datastructures used)
  - concrete data

- maybe compared to the typical “structure” of source-code and concepts::
  - informal comments, formal declarations (datatypes, variables) and concretly “incarnated” data on runtime

- rough comparison to JAVA:
  - informal meta-data: JavaDoc, general documentation
  - formale meta-data: Java-Reflection (input params, return value, annotations, names of methods, ...)
  - concrete data on runtime: method.invoke() using concrete values
toolwrapper 2:

- the process chain builder works using the formalized data of the header / concrete input/output definitions (which maybe simple example XML-documents => easy to understand by a human reader)

- points that work in our favor (the gifts we collect):
  - very concrete datastructure; close to the problem at hand (tagging, ...)
  - BUT: there are still some common elements and a common agreement on the basic structure (some kind of envelope); this formal data is based on domain knowledge/vocabulary (language tools&ressources) and is needed in order to do automatic orchestration
  - metadata (formal and informal) is separated from the concrete data
  - reuse of already defined XML-formats
  - envelope existing standards
toolwrapper formats and service bubbles

converter/transformer repository (web 2.0; automatic inference, ...)

D-Spin
common webservice I:

- some functionality, like authentication and QoS, can be addressed generally
- aim: don't implement the same stuff again and again, but configure established software for a customer (a resource/tool maintainer/consultant)
- some example functions:
  - String authenticate(...) => some simple solution at first
  - String getTicket(String authId)
  - String invoke(String ticket, String inputData) => call the toolwrapper
  - String getInputHeader(), String getOutputHeader() => process chain/registry
- The caller authenticates himself and requests a ticket. This ticket enables him to use a certain service (ensure availability, reserve resources). In some simple cases we might not need a ticket or a ticket is valid for a certain amount of calls or time.
common webservice II:
- an example configuration:
  maxSimultaneousTickets=10
  maxSimultaneousTicketsPerUser=4
  ticketLeaseTime=30000
  toolwrapperUrl=http://localhost/MyToolWrapper

- limited amount of tickets per user in queue; prefer users that call once/seldom to the ones currently running batch jobs, ...
- in future: Further development of this simple implementation to a solution according to Clarin (for example: use Shibboleth/ “Single Sign-On”-solution )
- Advantage: developed once; hides complexity from the tool maintainer
- maybe configured by a consultant and hosted/maintained by a ress. center
Some decisions having impact on the infrastructure

- services need unique id's (PID)
  => if the service changed the id has to change too
  => a service should be available today, tommorrow and next year in the exact same way (produce the same result for the same query)

- services have to protected from beeing used in a “wrong” way:
  - unauthorized access
  - overload (affects all users and maybe other services running on the same machine!!!)

- obtaining metainformation about a service should be an infrastructure function
current roadmap:

M1 (February 2009)
- specify 2-3 simple “dialects” like “TextCorpus”
- implement some basic functionality of the common webservice
- first tests on a small amount of services

M2 (March 2009)
- implement a central repository for all existing services;
- add more services (as different as possible) and build more complex chains

M3 (June 2009)
- implement a GUI tool that allows end users to build and execute process chains
current roadmap:

M4 (august 2009)
- evaluate the prototype-infrastructure => performance, scalability, flexibility, ...
- especially: observe users and find out how they really want to work using this infrastructure
- report back to Clarin:
  - how complex to implement and maintain
  - estimate costs for:
    - further development of the infrastructure (security etc.)
    - maintainance
    - additional tools/ressources
- integrate more and more standards set by Clarin (meta-data harvesting etc.)
Thanks for your attention!