Web Services for Language Resources and Language Technology Applications

Christian Biemann[*], Stefan Bordag[*], Uwe Quasthoff[*], Christian Wolff[†]

[*]Leipzig University
Computer Science Institute
Natural Language Processing Dept.
Augustusplatz 10/11
04109 Leipzig, Germany
{biem, sbordag, quasthoff}@informatik.uni-leipzig.de

[†]Regensburg University
Institute for Media, Information and Cultural Studies
Media Computing Dept.
Universitätsstr. 31
93040 Regensburg, Germany
christian.wolff@sprachlit.uni-regensburg.de

Abstract

In this paper we discuss the application of web service technology to the language technology (LT) and corpus processing domain. Motivated by a host of language technology tools which are widely available but which lack common technical standards for integrating them into language technology applications we discuss the implementation of a web service-based API for web-based processing and accessing of large linguistic data resources.

1. Introduction

Web services have been established as a new model for web-based distributed applications (see Vinoski 2002a, 2002b; Ferris & Farrell 2003). While the basic idea of making information services available via the world wide web has been discussed for some time (see Schranz 1998), the combination of a model of loosely coupled components with XML-based *standards* for accessing information and functionality on the WWW holds great promise for easier application access and integration.

Language resources are an obvious application area for web services, as there is a great need for standardized and web-based access to language resources. There are several benefits from making language resources available through web services:

- Language data becomes available via standardized interfaces and access mechanisms which is a great advantage in comparison with the variety of interfaces and APIs currently used for accessing language resources
- Other interfaces and APIs can be wrapped by a Web service, thus relieving the user of the need to program his own interfaces for each new source of information – thus, a programming language-independent level of functionality description may be reached.
- There is no need for an application to have large databases available locally.
- All technical details concerning data structures and implementation of corpus management or processing components can be encapsulated and hidden from the user or the application accessing a corpus via a web service.
- The same type of service (e. g. phrase lookup, tagging, or corpus analysis) may be offered by different language resource providers. Using web service registries, a client application in need of a specific language technology web service may ultimately select (or change) web services even during execution
- Easily available language resources might also be of great use for the *Semantic Web* initiative, providing means for to generating (local) ontologies (semi-) automatically.

On the other hand, the language resources and the according research may benefit as well from becoming easily accessible over the web:

- Most importantly, due to the standardized interfaces more compatibility amongst the various language resources will be achieved. In the end, all resources offering Web services might become fully compatible to each other without having to change their internal, years-grown structure.
- Another important aspect is the possibility of providing means for feedback, eventually generating a Wikipedia-like effect – errors and quality control in general of a given resource might be taken care of by the users to a great deal.
- Due to the feedback and standardization a synergy effect might take place, making the research in language resources more aware of the actual user needs.

In Quasthoff 1998 and Quasthoff & Wolff 2000 we have presented our approach to processing large text corpora and discussed the benefits of this kind of language resource and language technology application becoming available via the World Wide Web. In using a web service-based approach towards language data processing and access we believe that an additional step towards better LT availability and usage may be reached.

2. Applying the Web Service Paradigm

We discuss web services in the context of a large corpus processing project which has been available online for several years and which offers online access to monolingual text corpus information as well as text mining results. The online resources consists of general purpose as well as of domain-specific text corpora and at the same time addresses the needs of human users and are used for further processing in language technology (terminology extraction and verification, information retrieval, knowledge management).

While web-based front-ends for querying corpus information are quite straightforward and have been offered by many corpus research groups and language resource organizations (*ELRA*, *LDC*, the *Penn TreeBank* to name just a few), the problem of integrating language resource access in language technology applications calls for reli-

able and interoperable standards beyond parsing html code which might change any time.

From this point of view, web service standards like SOAP for encapsulating service request (*simple object access protocol*, see Gudgin et al. 2002), WSDL for service descriptions (*Web Service Description Language*, see Chinnici et al. 2002), and UDDI for maintaining service registries (*Universal Description, Discovery, and Integration*, see Bellwood 2002) promise to offer a solution comparable to what XML has done for information structuring: A unified and standardized method for accessing and integrating diverse and distributed language resources.

Figure 1 shows the typical web service scenario: A server publishes a web service in a service registry using WSDL / UDDI, while a prospective web service clients looks for the required service in the registry, selects an adequate web service host and finally uses the web service via a SOAP service call.

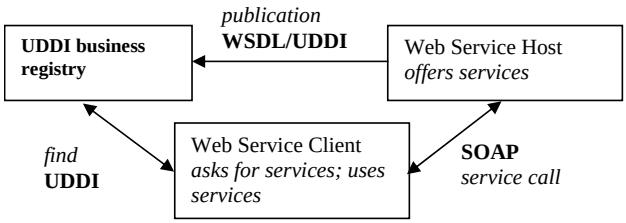

Figure 1: Typical Web Service Scenario

The following resources and tools are available in the context of our corpus processing infrastructure (see Quasthoff 1998, Quasthoff & Wolff 2000):

- Large text corpora from different domains (e. g. financial services, law, engineering) and temporal ranges as wells as in various languages like German, English, French, Dutch, Korean etc.
- A comprehensive dictionary of inflected word forms with a rich data structure for each entry (frequency information, semantic attributes, morphological and syntactical information).
- A set of algorithms, able to provide information about new or unusual words.
- Additional features extracted from text via text mining tools like collocations for each entry.
- A set of tools for corpus and dictionary setup, analysis, and maintenance.

All corpus analysis processes are based on standard database technology (MySQL). For each corpus (or, initially, each text collection to be processed) a database is created automatically. On top of these corpus processing technologies a number of different LT *applications* has been developed (see Böhm et al. 2002; Faulstich et al. 2002; Quasthoff et al. 2003):

- A web site as presentation platform for corpus querying, text mining results, and dictionary look-up
- services for *key word extraction* from arbitrary texts
- services for *automatic text summarization*
- tools for *daily media analysis*, key *concept extraction* and visualization
- services for knowledge management, especially extraction of *concept networks* from large text corpora.
- Text classification tools.

As these applications all make use of simple corpus lookup queries a bottom-up approach for modeling web services appears to be reasonable.

3. Modeling Web Services for LT Resources

Modeling web services for language resources requires technological as well as linguistic features to be taken into account. Our approach is currently restricted to the processing of large monolingual text corpora. Some of these modeling criteria are:

- Language resource *type* (e. g. text, multimodal)
- Language resource *domain* (e. g. financial services)
- *Processing phase* (data preparation, data analysis, data access)
- *Dialogue model* for web services (e. g. simple procedure calls vs. more complex and asynchronous corpus processing tasks)
- *Description vocabulary* used for a web service (i. e. an ontology for describing language resource-based services)

These modeling criteria serve as a starting point for identifying relevant types of services in the context of our text corpus infrastructure (for a more comprehensive overview of modeling criteria, see Schmidt & Wolff 2003; for a discussion of typical web service patterns see Haas & Orchard 2003).

4. Examples of Web Services

Starting with simple services which offer standard corpus queries like word and phrase lookup and queries for statistical information (e. g. absolute and relative word frequencies) we aim to define more complex types of queries in a bottom-up fashion (e. g. services for terminology extraction from corpora or generating collocation databases for a given corpus). The web services not only allow the access of (static) text corpus information as either primary information (the corpus data itself) or secondary/derived information on individual text corpora like text mining results (frequencies, collocations, additional categorical information, tagging data), but make text corpus processing tools available as well. The difference is actually not visible to the user – in both cases, an offered query is used and results are given. As the discussion of applications in the preceding chapter shows, there are manifold applications for web services in the LT domain. In the following, examples for four types of web services are given:

- Service requests which deliver information on single linguistic entities ("dictionary lookup queries"),
- requests for text mining results / frequency data,
- complex (application) tasks like tagging or keyword extraction and
- querying metadata for the underlying corpus data.

In the appendix, two small code examples for the service description as well as for a simple service call are given.

4.1. Simple Word-related Queries

getWordBaseForms	base form(s) of a word
getWordLanguage	language identification of a word or sentence

getExample	retrieves an example sentence for a given word, (sub-selection with additional criteria like sentence length, text type, date)

4.2. Text Mining and Frequency Data

getWordFrequency	frequency of a word
getWordFrequencyClass	frequency class of a word
getCollocates	list of collocations for a given word

4.3. Tagging and Information Extraction

getSentencePOSTags	part-of-speech-tagging for a given sentence
getSentenceNames	extraction of proper names from a given sentence
getTextKeywords	extract keywords from a given text

4.4. Meta Data on Language and Corpora

listCorpora	get a list of available corpora
getCorpusInfo	information on a specific text corpus (size. Date, text type(s), language)

4.5. Web Services for Corpus Processing and Corpus-based LT Applications

While the above lists are quite trivial as they exemplify web services which correspond to simple database queries, more complex tasks can be made accessible by web service standards as well. Basically we view two types of complex web services for our infrastructure:

- *Data processing* services which allow for the web service-based triggering of corpus analysis (text mining) processes and
- *Application Services* which encapsulate not simple data requests, but more complex language processing tasks as mentioned in ch. 2 above.

While the first type of service is addressed at researchers who would like to employ our tools for text and corpus analysis, the latter type is aimed at making language technology integration easier.

5. Implementation and Architectural Issues

A generic web service architecture based on SOAP as service wrapping standard, WSDL for service description, and the Apache Axis SOAP engine is used. Additionally a server side component provides *session management* and *user authentication* as different applications using web services may have access to different types of resources and services. All server side web service processing components are implemented as Java classes and interfaces. Figure 2 in the appendix gives a simplified overview of the architecture chosen for our approach.

6. Outlook

Among the open issues the *description problem* for web services: Without a standard ontology for describing language resource-related services, the discovery and integration of such services is possible only in a case by case fashion. We therefore propose the definition of a standard vocabulary for describing language resource related information and service functions. Another, more technical issue is the problem of composability of simple web services for complex application and the related question of session management for web services. Upcoming standards for composing web services like the *Business Process Execution Language for Web Services* may of help in this respect (see Thatte 2003).

7. References

Bellwood, T. et al. (2002): "UDDI Version 3.0"; Universal Description, Discovery and Integration (UDDI) Project, Published Specification, July 2002, http://uddi.org/pubs/uddiv3.htm.

Böhm, K.; Heyer, G.; Quasthoff, U.; Wolff, Ch. (2002). „Topic Map Generation Using Text Mining." In: J.UCS – Journal of Universal Computer Science 8(6) (2002), pp. 623-633.

Chinnici, R. et al. (2003). Web Services Description Language (WSDL) Version 1.2. World Wide Web Consortium Working Draft, June 2003, http://www.w3.org/TR/wsdl12.

Faulstich, L. C.; Quasthoff, U.; Schmidt, F.; Wolff, Ch. (2002). "Concept Extractor - Ein flexibler und domänenspezifischer Web Service zur Beschlagwortung von Texten." In: Hammwöhner, R.; Wolff, Ch.; Womser-Hacker, Ch. (edd.) (2002). Information und Mobilität, Proc. 8. International Symposium on Information Science, Regensburg, October 2002, pp. 165-180.

Ferris, Ch.; Farrell, J. (2003). "What are Web Services?" In: Communications of the ACM 46(6) (2003), p. 31.

Gudgin, M. et al. (2002). SOAP Version 1.2 Part 1: Messaging Framework. World Wide Web Consortium Working Draft, June 2002, http://www.w3.org/TR/soap12-part1.

Haas, H.; Orchard, D. (2003). Web Services Architecture Usage Scenarios. World Wide Web Consortium Working Draft, May 2003, http://www.w3.org/TR/ws-arch-scenarios/.

Heyer, G.; Quasthoff, U.; Wolff, Ch. (2002). "Knowledge Extraction from Text: Using Filters on Collocation Sets." In: Proc. LREC-2002. Third International Conference on Language Resources and Evaluation. Las Palmas, May 2002, Vol. III, pp. 241-246.

Quasthoff, U. 1998. "Tools for Automatic Lexicon Maintenance: Acquisition, Error Correction, and the Generation of Missing Values." In: Proc. First International Conference on Language Resources & Evaluation [LREC], Granada, May 1998, Vol. II, pp. 853-856.

Quasthoff, U.; Wolff, Ch. (2000). "An Infrastructure for Corpus-Based Monolingual Dictionaries." In: Proc. LREC-2000. Second International Conference on Language Resources and Evaluation. Athens, May/June 2000, Vol. I, pp. 241-246.

Quasthoff, U.; Richter, M.; Wolff, Ch. (2003). „Medienanalyse und Visualisierung: Auswertung von Online-Pressetexten durch Text Mining." In: LDV-Forum 18(1,2) (2003), pp. 452-459.

Schmidt, F.; Wolff, Ch. (2003). "Linguistic Knowledge Services – Developing Web Services in Language Technology". In: Böhme, Th.; Heyer, G.; Unger, H. (eds.) (2003). Innovative Internet Community Systems.

Proc. Third International Workshop IICS 2003, Leipzig, June 2003. Berlin et al: Springer, 229-238 [= Lecture Notes in Computer Science, Bd. 2877].
Schranz, M. W. (1998). World Wide Web Service Engineering. Object-Oriented Hypermedia Publishing. Ph.D. Thesis, Vienna University of Technology.
Thatte, S. et al. (2003). Business Process Execution Language for Web Services Version 1.1. Specification, second public draft release, May 2003, *http://www-106.ibm.com/developerworks/library/ws-bpel/*.
Vinoski, St. (2002a). "Web Services Interaction Models, Part 1: Current Practice." In: IEEE Internet Computing 6(3) (2002), pp. 89-91.
Vinoski, St. (2002b). "Putting the 'Web' into Web Services. Web Services Interaction Models, Part 2." In: IEEE Internet Computing 6(4) (2002), pp. 90-92.

8. Appendices

8.1. Sample WSDL Service Description – *Stemming Service*

```
<definitions name="urn:LdbApi" targetNamespace="urn:LdbApi">
<types>
  <xsd:complexType name="BaseFormResult">
    <xsd:all>
      <xsd:element name="baseforms" type="tns:StringArray"/>
    </xsd:all>
  </xsd:complexType>
</types>
<message name="GetBaseFormResponse">
  <part name="result" type="tns:BaseFormResult"/>
</message>
<portType name="LdbApiPort">
  <operation name="getWordBaseforms">
    <input message="tns:GetBaseFormRequest"/>
    <output message="tns:GetBaseFormResponse"/>
  </operation>
</portType>
<binding name="LdbApiBinding" type="tns:LdbApiPort">
  <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getWordBaseforms"/>
</binding>
</definitions>
```

8.2. Sample SOAP Web Service Code – *Stemming Request*

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope  soapenv:encodingStyle=
         http://schemas.xmlsoap.org/soap/encoding/ xmlns: (...)>
  <soapenv:Body>
    <ns1:getWordFrequencyClass xmlns:ns1="urn:LdbApi">
      <request href="#id0"/>
    </ns1:getWordFrequencyClass>
    <multiRef id="id0" soapenc:root="0" xmlns:ns2="urn:LdbApi"
              xsi:type="ns2:WordWithOptionalLanguage">
      <word xsi:type="xsd:string">Häuser</word>
      <language xsi:type="xsd:language" xsi:nil="true"/>
    </multiRef>
  </soapenv:Body>
</soapenv:Envelope>
```

8.3. Architectural Overview

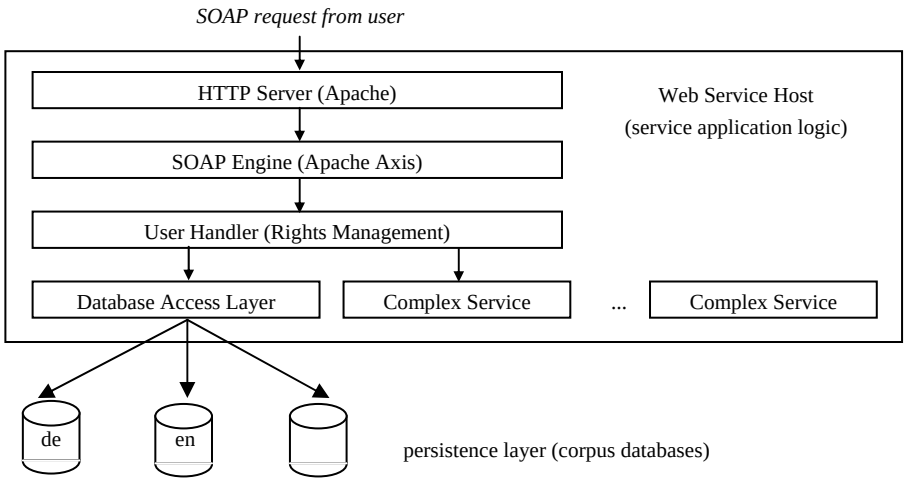

Figure 2: Simplified Overview of Language Technology Service Architecture