A case study on interoperability for language resources and applications

Marta Villegas, Núria Bel, Santiago Bel, Víctor Rodríguez
Universitat Pompeu Fabra
IULA, Roc Boronat 138, 08018 Barcelona, Spain
E-mail: marta.villegas@upf.edu, nuria.bel@upf.edu, santiago.bel@upf.edu, victor.rodriguezferrera@upf.edu

Abstract

This paper reports our experience when integrating differ resources and services into a grid environment. The use case we address implies the deployment of several NLP applications as web services. The ultimate objective of this task was to create a scenario where researchers have access to a variety of services they can operate. These services should be easy to invoke and able to interoperate between one another. We essentially describe the interoperability problems we faced, which involve metadata interoperability, data interoperability and service interoperability. We devote special attention to service interoperability and explore the possibility to define common interfaces and semantic description of services. While the web services paradigm suits the integration of different services very well, this requires mutual understanding and the accommodation to common interfaces that not only provide technical solution but also ease the user's work. Defining common interfaces benefits interoperability but requires the agreement about operations and the set of inputs/outputs. Semantic annotation allows defining some sort of taxonomy that organizes and collects the set of admissible operations and types input/output parameters.

1. Introduction

The research reported in this paper is part of the activities carried out within the CLARIN (Common Language Resources and Technology Infrastructure) project. CLARIN is a large-scale European project to create, coordinate and make language resources and technology available and readily useable by the European Humanities and Social Sciences (HSS) research community. HSS researchers will be able to efficiently access distributed resources and apply analysis and exploitation tools relevant for their research questions.

The CLARIN infrastructure strongly relies on the SOA (Service Oriented Architecture) approach to bring together the range of resources and tools available in the research community and make them accessible to others. All these resources constitute a distributed computing network or Grid.

In a Grid environment, researchers have access to Web Services which enable them to execute services on a remote system. Such architecture poses important challenges that need to be addressed: interoperability and integration between different tools and resources.

This paper reports our experience when integrating different resources and services into a grid environment.

From now on, this paper is organised as follows: Section 2 describes the use case we implemented. In Section 3, we describe the interoperability problems encountered especially focusing on the service interoperability problems. Section 4 is devoted to semantic annotations of services as a way of easing service interoperability. We first review some of the proposals from other e-science projects and, afterwords, we describe and exemplify our approach. Finally, in Section 5 we list our conclusions and lessons learnt.

2. Use case

The use case we describe here originates from a real case in our institution, where a researcher in historical linguistics is doing research on the evolution of the present perfect in different romance languages. She is performing a diachronic study and therefore she needs samples of these verbal constructions from different periods of time. Currently, the task of getting these primary data is not easy. Finding potential data providers is a very time consuming task as (at least for old Catalan and Spanish corpora) sources are rather small and scattered and, besides, they have little visibility.

Usually, corpus providers offer their data as web applications. Although, in essence, most corpus applications provide the same kind of service, the fact is that each application is different. Our researcher, therefore, needs to fill in different web forms following different instructions in order to get the desired data. Moreover, the data she gets comes in different formats and annotations which implies further efforts in data harmonisation tasks.

To ease the situation described above, we decided to build a data aggregator service that enables content search (and analysis) of scattered annotated corpora. At the moment, the aggregator accesses three Catalan annotated corpora[1], but we see it as an open application where potential providers can be eventually plugged-in. In this context, interoperability between scattered and heterogeneous data and services is crucial.

The corpus aggregator needs to address the following aspects:

Data collection: We need to supply a unique access interface to scattered data in order to reduce the efforts currently devoted to data collection tasks. It is important to notice here that, although from the user perspective the

1 These are: the Corpus textual informatitzat de la llengua catalana (CTILC at http://ctilc.iec.cat/), the Corpus Informatitzat del Català Antics (CICA at http://webs2002.uab.es/sfi/cica/) and the El Diccionari del Català Antic (DCA at http://www.ub.edu/diccionari-dtca/).

complexity of the whole system may be hidden, the system can not obviate this complexity and needs to guarantee certain aspects. Thus, even though the user may think that (s)he interacts with a unique repository (thus being unaware of the fact that behind the prototype there are different applications), the system needs to guarantee that all collected data are correctly described. This is especially important for journaling and tracking tasks, to make future replications possible and to guarantee correct citations.

Data integration: We need to guarantee interoperability of data coming from different sources.

Data analysis: We need to easy the integration of different NLP tools so that collected data can be analysed as a unique occasional corpus. The NLP tools we integrated include some basic statistical tools and a concordancer. In addition, a PoS tagger was also needed as some of the data were not annotated.

The overall architecture of the system includes three main modules: (i) the Data Collection Module, which is responsible for finding and getting the data the user needs and constitutes the first step in the sub-corpus building process, (ii) the Corpus Indexing Module, which receives collected data and indexes them into a unified temporal corpus and (iii) the Corpus Querying Module, which enables querying and analysing the corpus.

As we will see in much more detail in the next section, the problems we faced derive from metadata interoperability, data interoperability and services interoperability.

3. Interoperability

In this section, we first describe the metadata and data interoperability problems we have faced and, finally, we focus on the service interoperability problems.

3.1 Metadata and Data interoperability

Dealing with metadata coming from different sources is often difficult. In our case, for example, the three Catalan corpora include some 'date' metadata for document description, but neither the label nor the used values are common. In these cases, the use of standards is obviously a must. As far as metadata are concerned, we assume that the use of standards is the natural way to overcome heterogeneity and we suggest for wrappers that mediate between standards and 'in-house' metadata.

Unfortunately, semantic typing is not always enough. For example: the three corpora classify documents in terms of 'genre'. In this case, we not only lack an agreed genre classification tag set (or semantic typing) but also an agreed classification framework. Thus, whereas in the CICA corpus genre is encoded by means of a closed list of admissible values, in the case of the IEC corpus, genre is hierarchically organised.
Data interoperability is also complex as each corpus has a different annotation format. This is not an exceptional situation but rather a common one.

A survey we carried out on a set of 281 written corpora taken from ELRA database and CLARIN registry showed that only 30% of them reported some information about the used standard.

Despite the figures show that the use of annotation standards is far from being generalised, our approach adheres to the principle: "use standards whenever possible". Thus we require that collected data are annotated using MAF (Morpho-syntactic Annotation Framework) and we developed wrappers that served to accommodate provider's idiosyncrasies to the aggregator.

As far as interoperability of (meta)data is concerned, the corpus aggregator strictly adheres to standards. For those cases where the usage of a given standard is far from being a common practice, the standard is used as a sort of pivot language.

3.2 Service interoperability

The use case described in 2 implies the integration of different services into a large process or workflow. Broadly speaking, the overall system includes: (1) distributed metadata search, (2) corpus indexing process and (3) corpus querying and analysis process.

For the distributed metadata search, we benefit from the existence of the SRU[2] protocol developed by the Library of Congress. In 3.2.1 we briefly describe the protocol and the way we integrate it into the overall system.

For the rest of the services to be integrated in the use case, the situation is different as they are not web services ready to be plugged in. In this case, some of the services are well known applications such as the CWB (the IMS Open Corpus Workbench[3]), FreeLing[4], Apertium[5] or Weka[6]. Others are 'in-house' tools developed in perl or other script languages. In any case, they are all command line tools that generally need to be locally installed and executed. Prior to anything else, we had to deploy these NLP tools as web services.

In 3.2.2 we introduce some of the decisions taken when deploying these command line tools as web services. In section 4 we give a much more detailed description.

3.2.1 Distributed metadata content search

SRU (Search/retrieve via URL) is a search and retrieval protocol developed by the Library of Congress that uses the Internet to carry the messages between user and target. SRU enables searching remote systems having different specific query syntax, database designs and indexing conventions. The user's query is turned into a standard format. Remote servers receive that standard search message and translate it into the syntax that their databases understand.

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

3 http://cwb.sourceforge.net/

4 http://www.lsi.upc.edu/~nlp/freeling/

5 http://www.apertium.org/

6 http://www.cs.waikato.ac.nz/ml/weka/

Essentially, the SRU request parameters include the recordSchema and the query. The recordSchema specifies the schema in which the records must be returned. The query parameter contains a query expressed in CQL (Contextual Query Language) which is a formal language for representing queries to information retrieval systems. CQL uses Context Sets in order to ensure interoperability. Context sets enable CQL users to create their own indexes, relations, relation modifiers and boolean modifiers. Each context set has a unique identifier. Registered context sets include Dublin Core (DC) and MARC, among others.

Note that SRU protocol is a way to carry messages between user and provider. What is crucial here is that messages can be understood by both agents. Providers need to translate the search message into something that their databases can manage. In our case, providers prefer to export their data as XML files and reload them in Zebra[7]. Zebra is a digital library system which includes interfaces for SRW/U and CQL.

The kind of metadata we use in our corpora aggregator is rather simple and the DC context set is enough for our purposes. Obviously, CLARIN will need more sophisticated metadata descriptions but at the time we worked with our prototype these were not fully developed yet. We assume that further context sets will be defined in order to include CLARIN metadata descriptions

3.2.2 Web Services paradigm (common interfaces)

The ultimate objective when deploying NLP applications as Web services was to create a scenario where our user has access to a variety of services she can operate at her will. In addition, such services should be easy to invoke and be able to interoperate between one another if necessary. These requirements led us to carefully examine the approach to be followed and explore the possibility to define common interfaces.

In computer science, interoperability is achieved by separating interfaces from implementations.

SOAP web services are described using the W3C recommendation WSDL (Web Services Description Language). WSDL distinguishes between messages and ports: messages are abstract and describe the syntax and semantics of the service whereas, ports are concrete and give the address where services are invoked. A WSDL file can only include the abstract interface part, without giving details about the concrete implementation part.

Having a standard (SOAP) interface means having an agreed set of operations and their corresponding inputs and outputs. This is the situation we find in industrial business services: there is one common interface and different implementations.

From the WS perspective three aspects are crucial when dealing with common interfaces: (i) Equivalence of functionality categorisations between WS, necessary to ease searching of services and their interoperability. (ii) Compatibility of elements in I/O messages. (iii) Compatibility of schema structures in message elements.

Interoperability and (re)usability require that we define WS interfaces in a modular fashion separating between:

a) Type definition via XML Schemas (enabling type sharing and reusing)
b) Message definitions
c) Binding (enabling multiple service bindings to the same message)

WSDL not only enables modular publishing of services but also adding semantics to the descriptions. A common approach to web service discovery is the semantic annotation of services usually based on some sort of ontology. These annotations are then used for semantics based search and discovery. In Section 4 we briefly describe some of the approaches to semantic annotation of web services in e-science.

4. Semantic annotation of services

4.1 Brief overview

There are different approaches to semantic annotation of services (OWL-S[8], SAWSDL[9], WSMO[10]). They all agree on using some sort of semantic model that is used to annotate their services descriptions. These annotations are eventually used for classifying, discovering, matching, composing and invoking Web services.

OWL-S is an ontology meant to describe the properties and capabilities of Web services. OWL-S covers everything from service description to service grounding. OWL-S relates ontological concepts to real implementations.

WSMO also suggests an ontology based framework for service description. The ontology provides the terminology used by other elements of the framework. In WSMO, every resource description is based on ontologies and every data element interchanged is an instance of the ontology. Additionally, WSMO includes a description language WSML and an execution environment WSMX. The WSML conceptual syntax is used to model Ontologies, Web Services, Goals and Mediators which are the main components of the framework. WSMX is a software framework for runtime binding of service requesters and service providers. WSMX reads service requester's goal to discover matching services and make the service invocation. Optionally, WSMX provides data mediation for interoperability purposes.

SAWSDL is a W3C standard for semantic annotation of WSDL. SAWSDL provides standard means to relate WSDL documents to semantic descriptions. SAWSDL does not specify a particular semantic framework; rather it defines a small set of extension attributes used to refer

7 http://www.indexdata.com/zebra

8 OWL-S, Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/.

9 Semantic Annotations for WSDL and XML Schema. http://www.w3.org/TR/sawsdl/.

10 Web Service Modelling Ontology. http://www.wsmo.org/

to constructs. These constructs may belong to any external semantic framework.

Semantic descriptions of services need to be linked to the corresponding syntactic descriptions (the WSDL file). This linking is commonly known as grounding and can be performed in different ways: either the grounding is performed in the semantic model or in the syntactic one. OWL-S and WSMO describe grounding at the semantic layer but also allow expressing grounding at the syntactic layer by using the SAWSDL capability to extend WSDL. In OWL-S and WSMO, therefore, grounding is specified with links from the semantic descriptions and additionally allow semantic annotations in WSDL

Within the Bioinformatics field, we find more lightweight approaches. The MyGrid semantic model (Wolstencroft et al., 2007) distinguishes between Grid Service Ontology and Grid Domain Ontology which acts as controlled vocabulary for the model. Contrary to OWL-S and WSMO approaches, in the MyGrid Service Ontology invocation details are not included. Semantic services descriptions are encoded in XML documents conforming to the data model. This approach avoids much of the complexity of OWL-S based descriptions.

In the MyGrid Service Ontology, operations play a crucial role. Operations are essentially described in terms of the task they perform, the method and resource they use and the inpus/outpus involved. Both, inputs and outputs are instances of the Parameter type. Parameters distinguish 'real' parameters from configuration parameters and essentially can be defined in terms of their semantic type (which describes the domain specific data type) and the format (which describes the representation of the data).

Semantic descriptions of services are published in the registry. The Feta engine imports these descriptions together with the RDF version of the Domain Ontology and allows querying the system (Lord, et al, 2005).

Another example is that of Soaplab (Senger, et al 2003). Soaplab is a Web Services software framework used also in bioinformatics. Soaplab services are command line applications, wrapped as SOAP services, and served from a Soaplab server. All Soaplab services have the same generic set of SOAP operations as they all share a standardized interface. This is somehow an extreme example where all the services have the same WSDL interface. A Soaplab service is invoked from a Soaplab client, the Soaplab server calls the corresponding command line application.

Quite different is the approach followed by MOBY-S (REF) services. In MOBY, every Web Service passes messages validated against a global type system that uses an ontology-based messaging standard. MOBY-S defines all valid data types in an ontology. This simplifies the problem of interoperability by limiting the possible range of admissible interfaces.
Another proposal comes from the Open Geospatial Consortium, Inc. (OGC). The OGC is an international organization that is involved in the development of standards for geospatial and location based services. OGC works to create open and extensible interface and encoding standards for geographic information systems. The OGC Web Services Common Specification defines the aspects that are, or should be, common to interface Implementations. Essentially, these aspects have to do with the parameters and data structures used in operation requests and responses.

The OGC also defines a Geography Markup Language (GML) which describes an encoding specification for geodata in XML that enables the storage, transport, processing, and transformation of geographic information.

Currently, there are a good number of interoperable tools for geodata access and related geoprocessing services, thanks to the fact that software vendors implement their products in compliance with open geospatial web service interface and data encoding specifications.

Within the NLP domain we can also find interesting proposals. The NICT Language Grid Project[11] aims at building a multi-language service base, or Language Grid. They started to wrap various language resources existing on the Internet (machine translation engines, dictionaries, etc.) and enable Web services. To support these Web services, the Language Grid project defined a NICT service interface and a series of wrapping libraries for the purpose of developing Web services using the Java language.

These wrappers make language resources accessible through Web services, and adjust input and output of language resources to defined input/output using the NICT Language Service Interface.

Wrappers are deployed on the language grid service node, and accept requests from the grid core node. Results are returned to the core node, formatting the necessary data in the NICT Language Service Interface output format.

Currently, there are about 70 language resources published in the Language Grid. All these resources adhere to some of the available interfaces for machine translator, parallel text, morphological analyzer, bilingual dictionaries and adjacency pair services.

Language Grid advocates for a shared language service ontology that covers all possible elements in the domain (Hasashi, et al, 2007). This ontology includes three sub-ontologies for data resources, processing resources and abstract linguistic objects such as linguistic expressions and linguistic meaning.

Hasashi et al (2008) further explore this ontology and provide detailed descriptions for linguistic annotations and lexicons.

Klein and Potter (2004) also explore the development of an ontology for NLP services and suggest for an OWL-S description.

Unfortunately, none of the above proposals provide a

11 http://langrid.nict.go.jp/en/index.html

detailed ontology for NLP services.

4.2 Proposal

Our proposal necessarily deals with WSDL schema typing and semantic annotation of services. We regard schema typing as a first step towards interoperability and eventual semantic annotation.

4.2.1 Schema typing

Following other approaches we assume that messages are modelled according to some XML schema. Note, however, we do not force all objects moving around to adhere to some global type system as, unfortunately, we still lack an agreed type system. In addition, we assume that in some cases data type elements are not necessarily in XML format. Even though XML is widely used as data modelling and most annotation formats are expressed in XML, the fact is that we cannot ignore that a good number of NLP applications and tools consume and/or deliver data that are not XML data.

In order to have an idea of the kind of input elements we have in the NLP domain we have carried out a survey of the tools we deployed as web services in our institution. Broadly speaking we can classify these services into four groups: (a) annotation services, (b) concordancers (c) statistical services and (d) collocation services.

Annotation services are those that add some sort of annotation to an input text. Typically, these include tokenizers, PoS taggers. Among the services we explored, these are the ones that most likely consume and deliver XML data type objects.

Concordancers allow extracting sample contexts using (complex) content query. In this case we have enabled the CWB as web service. Statistical services perform some basic calculations about word distribution and provide some relevant lexicometric measures. Finally, our collocation services deploy the well known Ted Pedersen's Ngram Statistics Package[12].

Note that in these cases, services do not produce XML data type objects. Following Lord et al (2005), we think that service providers may be reluctant to spend time and resources describing inputs and outputs conforming to some global data model. A global model that, in many cases, does not exist.

Bearing all this in mind, our assumptions can be summarized as follows:

1) In the best case, types assigned to input / output elements come from a common type system defined in the data model. These types may simply describe the existence of a particular data type or can further describe the internal structure of the data type. (*pos* annotated corpus using CWB input format vs. *pos* annotated corpus using MAF format).
2) In the worst case, types are local and remain 'underspecified' as far as their content structure is concerned.

12Ngram Statistics Package http://sourceforge.net/projects/ngram/.

Allowing local and underspecified types contradicts the principle of interoperability but it is a pragmatic approach that

- Avoids being too restrictive. We understand that only when there is a large enough critical mass of services and users the network effect will naturally bring about the use of standards and common practices.
- Reflects current practices in the NLP domain and allows the natural creation of communities of interest.
- Even in the case of 'local' types, message typing allows sharing and reusability and benefits from XML machinery.

4.2.2 Semantic annotation:

Despite the nice efforts briefly reported and the ongoing work done in CLARIN, the fact is that we lack a general semantic description for NLP services.

In addition, services can be deployed for many different purposes, and not all necessarily foreseen by their initial developers. This means that often the initial classification does not foresee other perspectives. This is particularly relevant in the case of the CLARIN project since the potential user community is very large and heterogeneous.

Bearing these things in mind our position is based on the following premises:

1) We are far from suggesting any modelling for NLP services. We simply explore some possibilities that pave the way towards a service modelling and assume that different models can co-exist
2) In any case, at this early stage, the model is only meant to identify and (hopefully) describe the operations and objects that move around in an NLP service environment. In other words, the model does not pretend to model the domain of NLP but, rather, the services.
3) As far as operations are concerned, we assume that these can be mapped against some taxonomy (See below.)
4) Regarding the I/O, we need to distinguish between the objects moving around (that is, basic concepts in the NLP domain) from the formats these objects may have.
5) We strongly believe that different semantic models can co-exist.

We have seen that some sort of semantic grounding at the level of messages can be performed by making sensitive use of type declarations in WSDL (remember that WSDL allow type declarations and type import). For the operations, however, we cannot use this procedure. So we need additional means to enable the annotation of operations. As we saw in 4.1, one possibility is to use the annotation operation machinery of SAWSDL and place the grounding in the WSDL file. Another possibility is to assume that our services have a semantic description besides the syntactic one (ie the WSDL file) and place the grounding there. Note that this option means that we have (i) a syntactic description of web services (a WSDL file possibly enriched with typing), (ii) a semantic model that describes the kinds of operations and elements in the domain and (iii) a semantic description of services.

In our case, given the complexity of frameworks like OWL-S and WSMO, we will explore an approach similar to that of MyGrid.

Somehow MyGrid approach can be regarded as a faceted classification. We find a similar proposal in Dale et al (2006). They suggest a two-pronged approach to classification of services which includes a taxonomic classification and a faceted classification. The taxonomic classification places a given service in the taxonomy. The faceted classification characterises the service in terms of a set of attributes. The interesting aspect is that the system accepts having multiple taxonomies and that the possible values for each of the facets classification schema may derive from a taxonomy appropriate to that facet.

The CLARIN metadata infrastructure is based on the notions of elements, components and profiles. A metadata element is an atomic part of a metadata description and it is characterized by a name and a value domain. A metadata component is an aggregation of metadata elements and components aimed at describing a specific aspect of a resource. Profiles are similar to components except that they are used to describe all relevant aspects of a resource or collection.

Metadata elements are expected to be standardized by being defined in accepted registries such as ISO TC37/SC4 and Dublin Core. Components and profiles, however, are not standardized and users can create their own ones.

As far as services are concerned, the Service Component describes a service and its capabilities. The Service component includes the technical metadata required to support the process of profile matching (a process that checks whether a resource can be processed by a service). Service components essentially describe operations and the corresponding input/output parameters. Parameters are basically defined by means of a technical metadata component.

According to the CLARIN preliminary works on taxonomy, tools can be classified with respect to at least four facets: (i) Task/Problem, (ii) Approach/Technology, (iii) Implementation details and (iv) Format of processed data.

These top facets are naturally mapped against the MyGrid Service Ontology. Thus, we include in the CLARIN Service component the task element which serves to place the service into the taxonomy, the The approach/technology aspect corresponds to the operationMethod (which corresponds to the approach/technology aspect) and the service type. The operation component can be sketched as follows:

```
<xs:element name="operation">
<xs:complexType>
  <xs:sequence>
<xs:element name="Pid" …/>
<xs:element name="Name" …/>
<xs:element ref="operationTask" />
<xs:element name="operationDescription" …/>
<xs:element name="operationMethod" …/>
<xs:element name="operationApplication"…/>
<xs:element name="operationResource" …/>
<xs:element ref="input"  …/>
<xs:element ref="output"  …/>
          …
```

Figure 1: Operation component

Finally, I/O specifications are encoded in the Parameter component defined as follows:

```
<xs:element name="parameter">
    <xs:complexType>
       <xs:sequence>
         <xs:element name="Name" …/>
         <xs:element ref="parameterType" …
         <xs:element name="URL" … />
         <xs:element name="messageName" …/>
         <xs:element name="parameterDescription" …/>
         <xs:element name="XMLSchemaURI" …/>
         <xs:element ref="formats" …/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Figure 2: Parameter component

Essentially, parameter elements bear information about their semantic type, the schema (if any), the format and the involved message in the WSDL file (grounding).

4.2.3 Example

We have argued that there are NLP services for which it is reasonable to define a common interface. We can even expect that most I/O format conflicts can be solved by means of extensive use of standards and wrappers. We know, however, that things are not so easy: when thinking of a POS tagger we understand that it takes as input a text and returns a tagged text. Most POS taggers, however, require additional input parameters. Thus, for instance, FreeLing has up to 26 different configuration parameters and TreeTagger, 12.

In WSDL, operations can only have one input message and one output message. The input message defines the information the service receives when the operation is invoked. When defining an interface for a service that is already implemented, the data types of the implemented operations need to be assembled into messages. In other words, we must ensure that each parameter used by the method implementing the operation is represented in the message. The quickest solution, often, is to map all implementation parameters into the corresponding message parts.

This approach not only contradicts the wrapped document style where each message has a single part but also poses many problems to our assumption that part messages need to be typed and to the task of suggesting for common interfaces. Examples FreeLing and TreeTagger show that the attempt to define a common set of typed input parameters is not feasible when dealing with implementation parameters. Rather, we suggest an approach where complexity derived from ‘implementation idiosyncrasies’ has no significant consequences on interfaces. We achieve this by avoiding the proliferation of ‘idiosyncratic’ parameters in WSDL messages.

We start defining a POStagger operation with the corresponding input/output messages. Assuming we take the wrapped document style, the POStaggerRequest message has only one part. All implementation parameters will be represented in that part by means of the associated type POStaggerParams. POStaggerParams type is defined as belonging to the ParamsType. ParamsType is a complex type consisting of two kinds of elements: the mainParams and optParams. The mainParams refer to objects that typically move around in NLP services either as inputs or as outputs. The optParams refer to what we call ‘application parameters’, which are typically used as configuration parameters. mainParameters are expected to be common enough so as to be typed using some general type system. optParams are optional (they are assigned some default value) and may lack a general type.

In our PoS tagger example, mainParams include *text* and *language*. Both types are expected to be general and therefore collected in the general model. *Language* will obviously be declared in terms of ISOcat. *Text* type is a bit more complex as here we can further specify mimetypes, encoding formats etc. Functional parameters are collapsed into the optParams.

The corresponding XML payload for the POStaggerParams type goes as follows:

```
< POStaggerParams>
        < mainParams>
                <language>some language</language>
                <text>some input text to be tagged</text>
        </mainParams>
        <optParams>optional params</optParams>
< /POStaggerParams>
```

Complex type derivation (extension, restriction and abstract types) allows further refinements of declared inputs. Note that the POStaggerParams element above could be further constrained as a subtype of a global ParamsType. Whereas the ParamsType merely says it is a collection of main and optional parameters, the POStaggerParamsType would define the configuration of prototypical input parameters for PoS taggers and would determine that these include at least *text* and *language* and, optionally, a collection of optional parameters:

Note that our approach differs from that of Language Grid in that the suggested interface does not imply that we reduce complexity by ‘skipping’ what we call ‘application parameters’. In our case complexity derived from the differences in parameters is not removed from interfaces. Although for some users it may not be interesting to face such things, the fact is that there are users that will not trust in services unless they can tune the experiment and control it.

As far as the semantic description of services is concerned, the description for our FreeLing PoS tagger follows the Operation component schema in figure 1. Essentially, the description assigns a task from the taxonomy (PoStagger) and describes inputs/outputs in terms of some semantic type from the semantic model (ie. Language, Text and PoSAnnotatedText).

5. Conclusions

Scenarios where interoperability plays a crucial role will lead to the extensive usage of standards. Though, the present situation is rather disappointing, the need of standards will be evident when interoperability becomes a requirement. There are many examples from the industry and e-business.

Sometimes, providers need to adjust their data so that these are compliant with some standard. It is interesting to note that corpus providers saw this data migration as a positive fact despite the cost.

Though we strongly advocate for standards as a way to overcome interoperability problems, it is necessary to note that standards are not a panacea. We list some of the problems we have found when working with linguistic standards: (i) often standards are not well documented and lack examples, demos and tools. (ii) often linguistic standards are too ‘weak’ because they try to accommodate to ‘everything’ and eventually do not serve their purpose. Thus, for example, the fact that MAF allows for different annotation styles poses additional problems to the aggregator.

Some mismatches cannot be attributed to standards failure, but simply reflect differences in methods, approaches, theories etc… In our scenario, the way tokenisation is addressed by the different corpus providers poses additional problems. Again some agreement would be of much help when integrating heterogeneous resources.

In conclusion: we claim that standards enhance data ‘compliance’ but not data interoperability.

While the WS paradigm suits the integration of different services very well, this requires mutual understanding and the accommodation to common interfaces that not only provide technical solution but also ease the user’s work.

SRU/CQL is a protocol for message sending that also defines the syntax of those messages. The semantics of the messages (expressed in Context Sets) is crucial for mutual understanding. CQL enables different models that not only can co-exist but also could be mapped.

Defining common interfaces benefits interoperability but requires the agreement about operations and the set of inputs/outputs. Semantic annotation allows defining some sort of taxonomy that (i) organizes and collects the set of admissible operations and (ii) types input/output parameters. The ‘application driven’ differences observed, as far as input parameters are concerned, lead us to a pragmatic approach. Broadly speaking, functional parameters are optional (with assigned default values) and do not need to be typed according to some general schema.

Our conclusions can be summarized as follows: (i) in the best case, types assigned to input / output elements come from a common type system defined in the data model; (ii) these types may simply describe the existence of a particular data type or may further describe the internal structure of the data type; (iii) in the worst case, types are local and remain ‘underspecified’ as far as their content is concerned.

Allowing local and underspecified types contradicts the principle of interoperability but it is a pragmatic approach that: (i) voids being too restrictive. We understand that only when there is a large enough critical mass of services and users the network effect will naturally lead us to the use of standards and common practices. (ii) Reflects current practices in the NLP domain and allows the natural creation of communities of interest.(iii) Even in the case of ‘local’ types, message typing allows sharing and reusability and benefits from XML machinery.

As far as semantic annotation of WS is concerned, the model we suggest is only meant to identify and (hopefully) describe the operations and objects that move around in a NLP service environment. The model needs not to model the domain of NLP but, rather, the services. We assume that operations can be mapped against some taxonomy and that we need to distinguish between the I/O objects moving around from the formats these objects may have. Finally, we believe that different semantic models can co-exist.

6. Acknowledgements

This work, developed in the framework of the CLARIN project, has been supported by the Spanish Ministerio de Ciencia y Tecnología (ACI2009-0995), the Departament d'Innovació, Universitats i Empresa of the Generalitat de Catalunya, and the European Union (FP7-INFRASTRUCTURES-2007-1-212230). We wish to thank Eva Revilla for her helpful comments and support.

References

Akram, A, Meredith, D and Allan, R. (2006).Best Practices in Web Service Style, Data Binding and Validation for use in Data-Centric Scientific Applications. In *Proceedings of the UK e-Science All Hands Meeting 2006*

Atserias, J., B. Casas, E. Comelles, M. González, Ll. Padró and M Padró. FreeLing 1.3: Syntactic and semantic services in an open-source NLP library. *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006),* ELRA. Genoa, Italy. May, 2006.

Bramantoro, A, Tanaka, M, Murakami, Y, Schäfer, U, Ishida, U. (2008). A Hybrid Integrated Architecture for Language Service Composition. In *Proceedings of the 2008 IEEE International Conference on Web Services. (ICWS 2008 Research Track*).

Dale E. Lichtblau, Andrew W. Trice, Steven P. Wartik. (2006). Taxonomic and Faceted Classification for Intelligent Tagging and Discovery in Net-Centric Command and Control. In *2006 CCRTS The State of the Art and the State of Practice*.

Hayashi, Y. (2007). A linguistic service ontology for language infrastructures. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*

Hayashi, Y, Declerck, T, Buitelaar, P Monachini. M. (2008). Ontologies for a Global Language Infrastructure, In *Proceedings of the 1st International Conference on Global Interoperability for language Resources*, pp.105-11.

Klein, E and Potter, S. (2004). An Ontology for NLP Services. In Proc. of LREC 2004 Workshop on Registry of Linguistic Data Categories.

Kopecky, J, Roman, D, Moran, M and Fensel, D. (2006). Semantic Web Services Grounding. In *Proceedings of the Advanced Int’l Conference on Telecommunications and Int’l Conference on Internet and Web Applications and Services*

Pantschenko,K, Noppens, O and Liebig, T (2005). Grounding Web Services Semantically: Why and How?. *W3C Workshop on Frameworks for Semantics in Web Services 2005*.

Sakai, S et al. (2008). Language Grid Association: Action Research on Supporting the Multicultural Society. In Proceedings *of International Conference on Informatics Education and Research for Knowledge-Circulating Society (ICKS-08).*

Senger, M ; Rice, P and Oinn, T. (2003). Soaplab - a unified Sesame door to analysis tools. In Proceedings, UK e-Science, All Hands Meeting 2003, Editors - Simon J Cox, p.509-513, ISBN - 1-904425-11-9.

Schmid, H. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees.In *Proceedings of the International Conference on New Methods in Language Processing* (1994), pp. 44-49.

Lord, P., Alper, P, Wroe, C., and Goble, C. (2005). Feta: A light-weight architecture for user oriented semantic service discovery. In *Proceedings of the European Semantic Web Conference 2005*, Vol. 3532. pp. 17-31.

Wilkinson MD, Gessler DD, Farmer A, Stein L: The BioMOBY project explores open-source, simple, extensible protocols for enabling biological database interoperability. In *Proceedings of the Virtual Conference on Genomics and Bioinformatics 2003*.

Wolstencroft K, Alper P, Hull D, Wroe C, Lord PW, Stevens RD, Goble CA. (2007). The myGrid ontology: bioinformatics service discovery. *International Journal of Bioinformatics Research and Applications*. Volume 3 , Issue , Pages: 303-325.