A Web-based Architecture for Interoperability of Lexical Resources

Riccardo Del Gratta, Luca D'Onofrio, Roberto Bartolini, Tommaso Caselli
Alessandro Enea, Monica Monachini, Valeria Quochi
Claudia Soria, Antonio Toral and Nicoletta Calzolari

Istituto di Linguistica Computazionale
Consiglio Nazionale delle Ricerche
Pisa, via Moruzzi 1, Italy
{firstname.lastname}@ilc.cnr.it

Abstract

In this paper we present a Web Service Architecture for managing high level interoperability of Language Resources (LRs) by means of a Service Oriented Architecture (SOA) and the use of ISO standards, such as ISO LMF.

We propose a layered architecture which separates the management of legacy resources (data collection) from data aggregation (workflow) and data access (user requests).

We provide a case study to demonstrate how the proposed architecture is capable of managing data exchange among different lexical services in a coherent way and show how the use of a lexical standard becomes of primary importance when a protocol of interoperability is defined.

1 Introduction

If, as computational linguists, we were asked to express requests, surely, one request would be the possibility to access the "*mare magnum*" of the existing Language Resources (LRs) and, why not?, the whole web, as a huge repository from which to extract all information needed, avoiding differences in formats, extraction issues, content misalignment, semantics and so on. Clearly, many of the above requests are still far to be fulfilled, although the increasing use of standards and shared protocols represents a step forward for such unification.

The large diffusion of machine-based LRs and computer-aided language processing, together with their different applications, has led to the need of incorporating such computing infrastructures into distributed paradigms according to specific integration frameworks and standards for accessing different data.

Such frameworks should be able to provide a standard way to access data which originated by different LRs, as well as to hide the heterogeneity among such different resources, for instance in terms of data formats, networking, or middleware technologies.

Nowadays, lexical standards, such as LMF (Francopulo et al., 2006; Francopoulo et al., 2008), the Data Category Registry catalogue[1] (Wright, 2004) and LAF (Ide, 2004) (among others) indicate the path to follow for designing architectures dedicated to face interoperability issues among Language Resources.

So far, only few architectures have been proposed to manage the techniques for integrating heterogeneous LRs. Among these architectures, the Service Oriented Architecture (SOA) enables cooperation and interoperability of different platforms providing a *de facto* standard for discovering, accessing and sharing services, data and resources.

In this article we focus on a web-service-oriented architecture to address the high level interoperability (cfr section 2.1) among LRs and provide a case-study to demonstrate how our architecture is capable of managing data exchange among different lexical services in a coherent way.

2 Background

In this section we report a list of projects where LRs interoperability is a core element. In addition, we report related approaches to LRs interoperability and a recent research work which describes one possible approach to low level interoperability.

2.1 Language Resources and Technologies Interoperability

By "resource interoperability" we mean that two (or more) resources can be combined in a work-

[1] http://www.isocat.org

flow fashion. Resource interoperability is usually defined at two distinct levels: a high level interoperability which addresses input/output issues, normally related to the *structure* of the exchanged information, and one low level interoperability which manages the *content*, i.e. the actual domain of the exchanged information. Pioneering works on resource interoperability started in the 90s with the EAGLES [2] and ISLE [3] projects, whose results have been consolidated into ISO standards.

As a brief recap of international projects where resource interoperability played an important role, we can cite: IMDI[4], LIRICS[5], INTERA[6], CLARIN[7] among European projects; LANGGRID[8], and KYOTO[9] among collaborations between Europe and Asian countries.

2.2 Related approaches to LRs Interoperability

LRs interoperability can obviously be managed from different perspectives. Among others, we can cite the UIMA (Ferrucci and Lally, 2004) and GATE (Cunningham, 2002) frameworks. UIMA defines an internal object, the CAS (Götz and Suhre, 2004), used to describe both the input physical data and associated annotation. The CAS provides a common representation of annotations, (the *features structure*), to ensure interoperability among different cooperating UIMA components.

GATE uses the annotation graph (AG) (Bird and Liberman, 2001) to manage the standoff annotation. The Graph Annotation Framework (GrAF), (Ide and Suderman, 2007) can be used as a *lingua franca* to manage interoperability in both UIMA and GATE frameworks, cfr. (Ide and Suderman, 2009).

2.3 Low Vs. High Level Interoperability

The German D-SPIN project[10] identifies the high level interoperability as the *syntactic* interoperability, while the low level interoperability is called the *semantic* one. The low level interoperability is called *semantic* since it concerns the semantics (content) of the interchanged information. The use of the Data Category Registry catalogue is fundamental to guarantee a *semantic* interoperability.

2.4 Remarks on High and Low level Interoperability

In this article we focus on high level interoperability, i.e. on the *structure* of the exchanged block of information among different resources. However, we, both as computer scientists and linguists, are aware that the low level interoperability, i.e. the *semantic* of the exchanged information, plays (and will play more) a big role in interoperability scenarios.

Despite of such important role, we took voluntarily apart *semantic* matters in order to define a basic architecture upon which it could be possible to build more specific software blocks capable to manage low level interoperability as further steps. See sections 3.4 and 5 for more details.

2.5 The web services dilemma

Web services for LRs or LRs as web services? (Calzolari, 2008).

Before the explosion of Internet based protocols, LRs were designed to be fully static. This means that one LR, for example a lexicon or a parser, was provided to end users by the resource owner in a physical device, eg. a CD. There was no clear distinction between the resource and the access to the resource. This distinction has became clearer because of the increasing of network based protocols and web services. In fact, resources are defined once and different services can be built upon resources to provide different data. According to this, resource owners can grant the access to the resource. Moreover, LRs accessible via web services increase the number of accessible resources by creating new resources on the basis of existing ones (web service orchestration). The LANGGRID project, for example, presents LRs in the form of distributed dynamic language services.

Conversely, LRs can be defined and built as web services. In such case resources can be viewed as parts of a vast infrastructure, and interoperability among resources is guaranteed by the infrastructure paradigms instead of the rules defined in the web services orchestration. LRs which belong to such infrastructure can be considered as shared distributed repositories and can be offered to the community as “language services”.

[2] http://www.ilc.cnr.it/EAGLES96/home.html
[3] http://www.ilc.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm
[4] http://www.mpi.nl/IMDI/
[5] http://lirics.loria.fr
[6] http://www.elda.org/intera
[7] http://www.clarin.eu
[8] http://langrid.nict.go.jp/en/index.html
[9] http://www.kyoto-project.eu/
[10] http://weblicht.sfs.uni-tuebingen.de/

Such an approach considers LRs as *fully* dynamic; resources can vary since the services it offers can vary. These two visions are not in contrast but, on the contrary, represent two alternatives for managing LRs: resources accessed via web services become fully dynamic resources when defined as lexical services. This landscape is, so far, in the future, but, as a prototype of such implementation, the *LexFlow* engine (Soria et al., 2006) already uses many of such fully dynamic techniques.

2.6 SOA and Web Services Orchestration

Web services are Internet-based applications that communicate with other programs to exchange data and/or functional services. These services are implemented by exposing a standard interface to wrap the back-end applications as well as other available web services as depicted in figure 1.

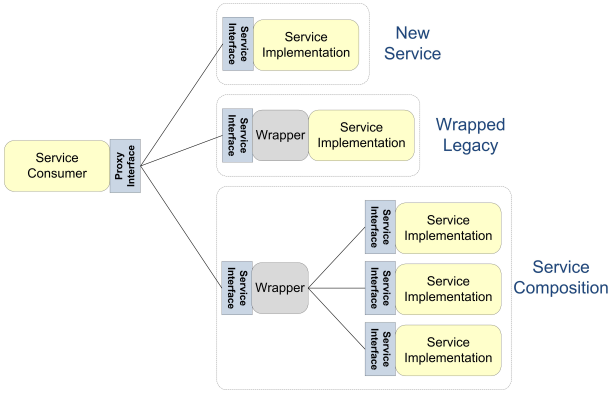

Figure 1: Web services architecture

The web service orchestration, i. e. the exchange of structured information between services as reported at the bottom right of figure 1 is crucial to facilitate interoperability, and different orchestration languages, such as WS-BPEL, are utilized to define rules for interoperability.

3 An architectural model for interoperable Language Resources

To design a distributed and open system to handle heterogeneous LRs, we propose an architectural model which is based on web services. We adopted a layered SOA paradigm that introduces the concept of service as the fundamental element for developing distributed applications in a network-based environment. Essentially, such models provide:

- functions to elaborate and present data to the end user;
- functions to discover, compose and elaborate services from other published services.

The development of web technologies and standards[11] to manage them, enables pervasive adoption and deployment of web services to reach high level interoperability.

In planning complex data-centric systems, the decomposition of the whole system in different layers is the most used approach. Each layer offers different services and is characterized by a deep functional specialization.

The proposed architecture is structured in horizontal specialized layers, and in a set of inter layer services which act as linkers between horizontal layers, as illustrated in figure 2. Grouping similar services in dedicated layers offers different advantages: horizontal layers deal with service composition and data managing, while, transversal ones manage interfaces and messages among horizontal layers.

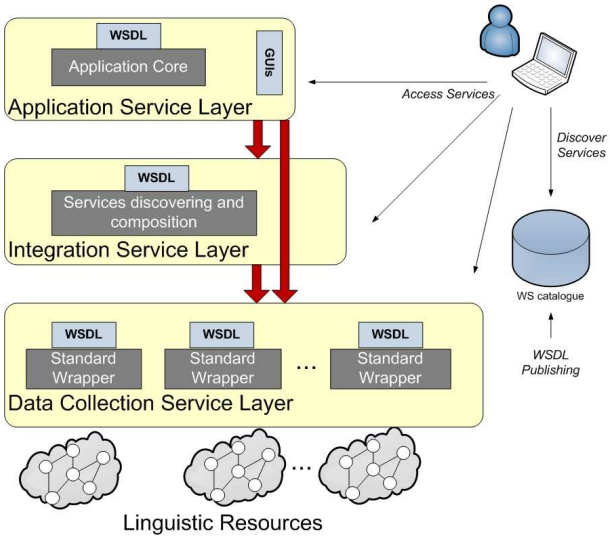

Figure 2: The layered architecture

This approach makes it possible, through the collaboration of various less complex components, to build a more robust, scalable and maintainable architecture with a clear logical structure.

In a bottom up order, horizontal layers, in figure 2, are the following:

Data Collection Service Layer. This is an interface for available LRs. The main goal of this

[11] Among others HTTP, XML, SOAP, WSDL ...

layer is to translate proprietary linguistic data format in a common data model which provides a standard view of the resources[12];

Integration Service Layer. This layer provides aggregated services. Single lexical services extracted from Data Collection Service Layer are combined accordingly to users' requirements;

Application Service Layer. This layer is the interface between the software/human agents and services provided.

In figure 2 horizontal layers represent macro-components for offering/using services. Transversal services (the arrows) represent the information flows (e.g. control and data) associated with component interaction.

3.1 Application service layer: a Service-based Architecture for LRs Interoperability

The application service layer is the top level of the proposed architecture (see figure 2) and it is detailed in figure 3.

The main purpose of this layer is to define a front-end for both human and software agents for creating/using services. According to agent requests, these services could be assembled into a more complex NLP pipeline through their orchestration. The application layer sends the possible request of orchestration to the integration service layer (the middle layer in figure 2) and forwards the orchestrated service to the agents.

At the bottom of the application layer, the persistence layer is an interface towards the Data collection (see again, figure 2) and it is responsible for wrapping LRs.

3.2 Services discovering and composition

The Integration Service (middle layer in figure 2) is the core layer which makes linguistic interoperability possible.

Web Services based systems are typically composed by orchestrating a number of atomic[13] services in a common workflow which uses a machine-to-machine interaction over a network.

[12]This target is achieved by associating an LMF data model to raw linguistic datasets for lexicons. Standardization for NLP tools is carried out at I/O level.

[13]Atomic services are services which perform one single operation, in our case linguistic services are atomic services.

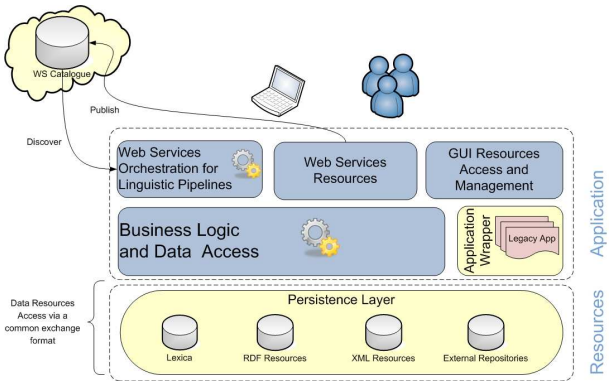

Figure 3: The application service layer

Web services composition is necessary for complex linguistic models where the composition logic is tightly driven by linguistic processes[14].

The Integration service layer is responsible for browsing a catalogue of available web services to obtain composed value-added services with desired capabilities. In fact, the discovering of web services aims at their composition to offer high level functionalities in more complex processes.

The Integration service layer has two main functions:

- discovering available services within repositories;
- mapping offered linguistic features onto a standard input/output format.

Each single atomic web service offers one or more linguistic features for data exchange. These features are used by the integration service layer as parameters for composing a web services based workflow. In such workflows, web services are composed (orchestrated) according to both the activities (of single web services) and the transitional states (of the workflow).

We assume a workflow described as

$$A = (s, a) \Leftrightarrow A = s_0\, a_1 \circ s_1\, a_2 \ldots s_{N-1}\, a_N \quad (1)$$

Where a_i is the single $i-th$ activity among all activities specified within the workflow and s_j is $j-th$ state (transition) among the set of transitions used to represent the logic of the workflow. Figure 4 shows the web services composition logic.
The web-service-composition logic as specified in equation 1, is made possible by the standardization

[14]For example, complex NLP pipelines which start from raw texts and provide a dependency parsed output.

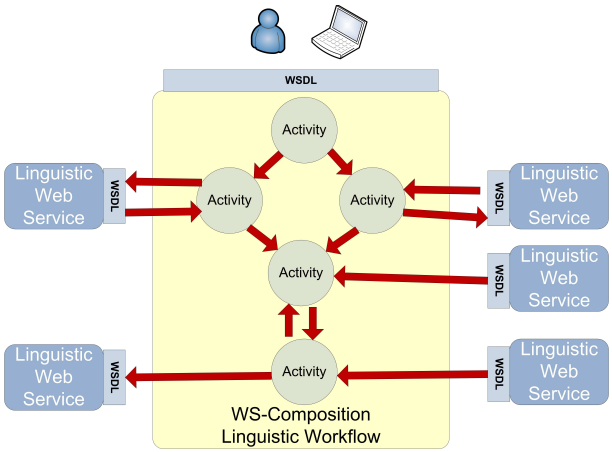

Figure 4: Web services composition logic

of the input/output structures exchanged by single activities. The integration service layer is built upon a BPEL Engine which provides the necessary functionalities for composing and deploying a new service.

3.3 Data Management Service Layer: a data model for Language Resources

The data collection layer is the bottom layer in figure 2 and it is responsible for managing heterogeneous resources. Data management for these resources is carried out at two distinct levels: resource description and provided information formats. Both levels are needed to improve high level interoperability.

Legacy resources, for example, can be described in relational databases as well as either XML or RDF documents and can provide information encoded in various formats. The proposed data collection layer uses, on one hand, descriptive metadata for defining XSD or DTD schema files to describe heterogeneous resources; on the other hand, it adopts standards, such as LMF for lexicons, to unify exchanged data.

3.4 Web Services and Semantics. What is going on?

The above described architecture as well as the web services are developed within the CLARIN project and the role of *WS catalogue* (see figures 2 and 3) is played by the *Component Metadata* for web services defined within the project. However, we are aware that web services (either WSDL or REST based) operate at syntactic level (with WSDL and WADL, respectively) but lack the semantic specificity needed to represent what web services can actually do. Semantics, in fact, can be used to improve web services discovery and to facilitate their orchestration.

As a further step, a registry of "semantic annotated" web services will play the role of the *WS catalogue*[15] in figures 2 and 3.

4 Case study

We present a case study to illustrate how the proposed architecture works in actual linguistic landscapes. In this scenario we map two lexical resources: an English domain specific lexicon, the BioLexicon (Quochi et al., 2008), and an Italian general domain lexicon, SIMPLE, (Ruimy et al., 2003; Ruimy, 2006).

Mapping these two lexicons allows users to obtain, given a domain specific English term, related general domain terms in Italian.

The above mentioned lexicons are mapped using two external resources: an Inter Lingual Index (ILI) to map WordNet[16] (Fellbaum, 1998) to the Italian WordNet, ItalWordNet, (Roventini et al., 2000) synsets and a mapper between ItalWordNet and the SIMPLE lexicon. Each resource (both the lexicons and the *mappers*) has an LMF conformant export which encodes the information exchanged among different services.

These LMF files need to be parsed during the process but, since they have the same (XML) structure, whatever resource they have been extracted, the XQUERY technique, for example, allows to extract the same nodes from different LMF files. This enables designers to develop one single detailed LMF parser capable of extracting the correct information from the LMF tree[17].

4.1 Case study technical aspects

This section shows how architectural layers, (see section 3) are used in our case-study.

We do not report any details about implementation, but rather focus on the general interoperability model of the SOA and its applicability in the specific scenario.

Architecture layers shown in figure 3 are detailed below:

[15] http://www.w3.org/Submission/WSDL-S and local references for detailed information.

[16] http://wordnet.princeton.edu/
version 3.0

[17] For example, the APIs to extract synsets can be used for both Italian and English WordNets.

Data collection layer: this layer is mapped on lexical resources. We have defined web services for accessing the above mentioned resources in their native formats[18] and providing an LMF export of a given word;

Integration service layer: we define a pipeline of web services to map the BioLexicon to SIMPLE. This layer is responsible for selecting the right service when needed. It contains a workflow engine which is in charge of monitoring the pipeline according to the composition rule (see equation 1). In section 4.2 we provide more details on the workflow engine;

Application service layer: we have implemented this layer according to the high level interoperability paradigms. Web services are orchestrated by using information on their input/output descriptive metadata. This means that in the "Business Logic" (see figure 3), the service composition is driven by input/output specifications.

4.2 Integration Layer workflow

The core of the SOA architecture is the Web Service Composition Engine, see figure 5: it executes the basic service invocation and data mapping. The integration service layer aggregates the web services exposed by the architecture. According to the composition rule in equation 1

$$A = (s, a) \Leftrightarrow A = s_0\, a_1 \circ s_1\, a_2 \ldots s_{N-1}\, a_N \quad (1)$$

we segment the case-study in simple activities (a) and states (s).

$\mathbf{s_0}$ A user agent invokes the service that exports an input lexical entry, extracted from the BioLexicon, in LMF;

$\mathbf{a_1}$ A web service, which belongs to the data collection layer, queries the BioLexicon and exports the input lexical entry in LMF. The activity $\mathbf{a_1}$ sets the global status of the workflow, $\mathbf{s_1}$, either to "Ok" or to "Failed" depending on the exit code of the query;

$\mathbf{s_1}$ If the status $\mathbf{s_1}$ is set to "Ok", the workflow engine executes the service for managing the first mapping resource. Otherwise the workflow engine exits;

$\mathbf{a_2}$ A web service parses the input LMF extracting the searched lexical entry. Then it queries the English WordNet LMF file extracting the synsets which contain the input lexical entry. The global status $\mathbf{s_2}$ is set to "Found" when at least one synset is extracted, otherwise the status $\mathbf{s_2}$ is set to "Exit";

$\mathbf{s_2}$ If the status $\mathbf{s_2}$ is set to "Found", the workflow engine executes the service for extracting Italian synsets from ItalWordNet. Otherwise the workflow engine exits;

$\mathbf{a_3}$ The Italian WordNet LMF file is parsed for retrieving Italian synsets which correspond to each of the WordNet synset(s) extracted in activity $\mathbf{a_2}$. The global status $\mathbf{s_3}$ is set to "Mapped" if an Italian synset is found, otherwise the status is set to "Continue". This service selects and provides to the application layer only Italian synsets linked to the English ones by either synonymy, EQ_SYN (*syn*), or near synonymy, EQ_NEAR_SYN, (*nsyn*) relation types and filters out the hyperonym, EQ_HAS_HYP (*hyp*) relations;

$\mathbf{s_3}$ If the status $\mathbf{s_3}$ of workflow is set to "Mapped", the workflow engine can execute the service for extracting the Italian word which corresponds to the input lexical entry. Otherwise the workflow engine exits;

$\mathbf{a_4}$ The LMF mapping file between ItalWordNet and SIMPLE is parsed for retrieving the Italian word(s) which corresponds to the input lexical entry. An LMF export of such words is sent back to user and the status is set to "Completed";

The web service composition engine works together with the integration service layer so that the resulting composition rule is:

$$A = (s, a) \Leftrightarrow A = s_0\, a_1 \circ s_1\, a_2 \circ s_2\, a_3 \circ s_3\, a_4$$

The integration service layer is responsible for pipelining web services selected in activities $a_{1,2,3,4}$ and to return the obtained web service to the application layer. The final result is prompted back to the agent.

4.3 Activities details

We detail activities described in section 4.2 by means of examples. We describe the activities

[18] Each lexicon is, in our case, a relational database.

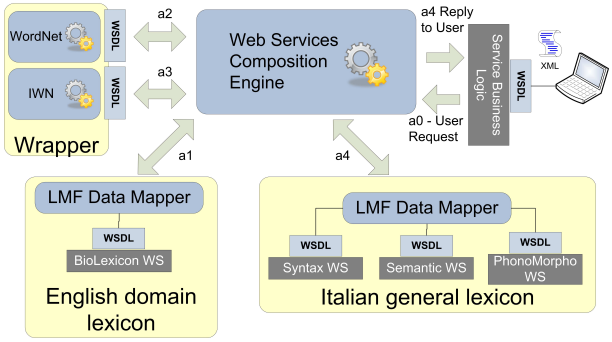

Figure 5: The proposed case study

needed to map the verb "activate" to automatically extracted Italian verbs.

For the sake of clarity, we provide information extracted from the LMF files by each activity. We start describing from activity $\mathbf{a_2}$, i.e. after the extraction of the lexical entry from the input LMF file.

Activity $\mathbf{a_2}$ The verb[19] is extracted from the input LMF file. The English WordNet is parsed and the corresponding synsets are extracted.

Synset Id	**Variants**
01643657-v	trip, actuate, trigger, **activate** ...
00190682-v	**activate** ("activate an old file")...
00191385-v	**activate** ("activate a metal")...
00190999-v	**activate**, aerate ...
00190886-v	**activate** (make (substances) radioactive)...

Activity $\mathbf{a_3}$ Mapping WordNet to ItalWordNet synsets. We have:

WordNet: 01643657-v

IWN Synset	**Variants**	**Relation**
41042-v	azionare	$nsyn$
36708-v	frenare	hyp
32604-v	attivare	$nsyn$
...	...	...

WordNet: 00190682-v

IWN Synset	**Variants**	**Relation**
41042-v	azionare	$nsyn$
39951-v	telecomandare	hyp
32604-v	attivare	$nsyn$
...	...	...

[19]Hereafter, by verb we mean the lexical entry which encode, according to the LMF specifications, the verb "activate".

This extraction and mapping procedures are repeated for each WordNet synset. We have reported here only two sets of mapping to show how the extracted information looks like.

We only consider the ItalWordNet synsets which are linked to English WordNet with a relation type either $nsyn$ or syn[20].

Activity $\mathbf{a_4}$ From verbs which belong to the Italian synsets, we extract the verbs that appear most frequently. Most frequent verbs are calculated counting the frequency of a given lexical entry in the set of extracted synsets.

These words are the *connector candidates* between the English and Italian terms.

Frequence	**Variant**
2	azionare
2	attivare

Lexical entries extracted are then encoded in LMF[21] and sent back to the user.

```
<LexicalEntry id="LE_attivare">
  <POSDC POSAtt="partOfSpeech"
   POSVal="verb"/>
  <Lemma id="LM_attivare"
   basename="attivare"/>
  .....
</LexicalEntry>
.....
<LexicalEntry id="LE_azionare">
  <POSDC POSAtt="partOfSpeech"
   POSVal="verb"/>
  <Lemma id="LM_azionare"
   basename="azionare"/>
  .....
</LexicalEntry>
```

We note that two Italian lexical entries, *azionare* and *attivare*, are found to correspond to the initial "activate" verb. In such case, there is no preferred connector candidate between English and Italian lexicons and the semantic information which can be extracted from SIMPLE (e.g. domain restrictions, predicates, relations) can be used by the user to select the most suitable lexical entry. In other word, the disambiguation between the two candidates (*azionare* and *attivare*) is performed in *a*

[20]We extract synsets related by the $nsyn$ relations only if no synset is related by the syn relation.

[21]The LMF excerpt provided is just a sample.

posteriori fashion using context related information.

In the case-study we just provided, there is no automatic (*a priori*) disambiguation, cfr. section 4.4 for more details.

4.4 Results

We iterated the procedure over all verbs contained into the BioLexicon and extracted two interesting figures: the coverage between English and Italian verbs and the automatically disambiguated verbs.

The BioLexicon contains both domain specific and general verbs. Table 1 shows the percentage of the coverage between English and Italian verbs divided in domain specific and general. The second figure is the automatic disambiguation, i. e. the percentage of cases in which only one most frequent Italian verb is the candidate for the English verb. Table 2 shows the automatic disambiguation between the previously 643 covered verbs divided in domain specific and general. These verbs can have one or more than one candidates. For example, the English `clean` has 12 Italian possible candidates but 3 of them appear 6 times: `nettare, pulire, ripulire`. These 3 Italian verbs are extracted as the most frequent verbs but they are not automatically disambiguated. On the contrary, the English verb `react` has only one most frequent Italian `reagire` verb and can be automatically disambiguated.

Domain	English	Covered	%
General	496	334	0.67
Specific	658	309	0.47

Table 1: Percentage of coverage between English and Italian verbs.

Domain	Total	Disamb.	%
General	334	192	0.57
Specific	309	170	0.55

Table 2: Percentage of automatically disambiguated verbs.

Results in tables 1 and 2 are not surprising: domain specific verbs are hardly to be mapped and/or automatically disambiguated. However the percentage of 55% for domain specific verbs (see table 2) is encouraging, since the described mapping has been performed without taking into account the semantic contributions that a resource such SIMPLE can provide.

4.5 Alternative approaches to our case-study

The proposed case-study can be approached and resolved with different techniques.

For example, the UIMA approach can resolve the lexicons mapping introducing a CAS to represent the LMF input file which contains the searched lexical entry, "activate" in our example. The CAS is managed and updated by specific Analysis Engines[22] responsible for accessing the resources and managing the workflow status. The updated CAS is sent back to the user in a standard standoff annotation (XMI) file. This file needs to be parsed to extract needed information before it could be encoded in LMF, for example.

This approach is, however, valid and has been already developed, at our institute, to define general TimeML classes(**?**). See for example (Del Gratta et al., 2008) and related references. However, we have preferred the web service techniques because they allow developers to access resources both locally and remotely by means of url-based descriptors upon which clients to access resources can be easily built. On the contrary, UIMA needs its framework to be distributed to allow users to develop their Analysis Engines.

Moreover the UIMA framework does not natively provide information encoded in LMF, while the web services we provided *are* totally designed to manage LMF as input/output files.

5 Conclusions and future work

We have presented a SOA-based architectural model based on service layers to face high level interoperability among LRs. The architecture consists of three different service layers which are responsible for managin data collection as well as data aggregation and access. This structure allows developers to design different kinds of applications (data access), using standard software interfaces to invoke and compose low level services (data aggregation and collection).

To show the advantages of the proposed architecture, we have presented a case-study in which we map two different lexical resources using the LMF standard as common data model. To add

[22] Analysis Engines are, according to UIMA, software programs which access/update the CAS.

the low level interoperability layer to our architecture, we are currently examining the Data Category Registry in order to map the content of the exchanged information upon standard concepts. This mapping can allow to manage the same structured files (for instance two LMF files) which use a different tagsets for tagging the part of speech, for example.

In our future works we intend to develop different data model according to other ISO standards, such as LAF, GraF to manage high level interoperability among NLP tools such as POSTaggers, parsers ...

Moreover, we intend to add semantics to our existing web services WSDL descriptor files so that an external service ontology can record and link web services according to their functionalities simplifying both web service browsing and composition. In fact, according to WSDL-S paradigms, for example, we can offer an evolutionary and compatible upgrade of existing web services standards that externalize the semantic domain models to ontology representation languages. In this perspective, some linguistic specific issues, such as used tagsets, information provided as well as used semantic concepts can be managed at web service semantic level description.

References

Riccardo Del Gratta , Tommaso Caselli, Nilda Ruimy, and Nicoletta Calzolari. 2008. Timeml: An ontological mapping onto uima type systems. In A.C. Fang (eds.) N. Ide, editor, *Proceedings of the First International Conference on Global Interoperability for Language Resources (ICGL'08)*, pages 89–96, Hong Kong, SAR, January.

Steven Bird and Mark Liberman. 2001. A formal framework for linguistic annotation. *Speech Communication*, 33(1-2):23–60.

Nicoletta Calzolari. 2008. Approaches towards a "lexical web": the role of interoperability. In *Proceedings of the First International Conference on Global Interoperability for Language Resources (ICGL'08)*, Hong Kong, SAR, January.

H. Cunningham. 2002. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254.

Christian Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database (ISBN: 0-262-06197-X)*. MIT Press.

David Ferrucci and Adam Lally. 2004. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348.

Gil Francopoulo, Nuria Bel, Monte George, Nicoletta Calzolari, Monica Monachini, Mandy Pet, and Claudia Soria. 2008. Multilingual resources for nlp in the lexical markup framework (lmf). *Language Resources and Evaluation*.

Gil Francopulo, Romary Laurent, Monachini Monica, Nicoletta Calzolari, et al. 2006. Lexical markup framework (lmf iso-24613). In European Language Resources Association (ELRA), editor, *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, Genova, IT.

Thilo Götz and Oliver Suhre. 2004. Design and implementation of the uima common analysis system. *IBM Systems Journal*, 43(3):476–489.

Nancy Ide and Keith Suderman. 2007. Graf: A graph-based format for linguistic annotations. In *Linguistic Annotation Workshop, ACL 2007*, Prague.

Nancy Ide and Keith Suderman. 2009. Bridging the gaps: Interoperability for graf, gate, and uima. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 27–34, Suntec, Singapore, August. Association for Computational Linguistics.

Nancy Ide. 2004. International standard for a linguistic annotation framework. *Journal of Natural Language Engineering*, 10.

Valeria Quochi, Monica Monachini, Riccardo Del Gratta, and Nicoletta Calzolari. 2008. A lexicon for biology and bioinformatics: the bootstrep experience. In Nicoletta Calzolari et al., editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

Adriana Roventini, Antonietta Alonge, Francesca Bertagna, Bernardo Magnini, and Nicoletta Calzolari. 2000. Italwordnet: a large semantic database for italian. In Nicoletta Calzolari et al., editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Athens, Greece, JUne. European Language Resources Association (ELRA).

Nilda Ruimy, Monica Monachini, Elisabetta Gola, Nicoletta Calzolari, Cristina Del Fiorentino, Marisa Ulivieri, and Sergio Rossi. 2003. A computational semantic lexicon of italian: Simple. *Computational Linguistics in Pisa, Istituto Editoriale e Poligrafico Internazionale*, pages 821–864.

Nilda Ruimy. 2006. Computational multi-layered italian lexicon for hlt applications. In *Proceedings XII EURALEX International Congress, Atti del Congresso Internazionale di Lessicografia*, volume 1, pages 221–227, Torino, 6-9 settembre.

Claudia Soria, Maurizio Tesconi, Francesca Bertagna, Nicoletta Calzolari, Andrea Marchetti, and Monica Monachini. 2006. Moving to dynamic computational lexicons with lexflow. In European Language Resources Association (ELRA), editor, *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'2006)*, pages 7–12, Genova, IT.

S. E. Wright. 2004. A global data category registry for interoperable language resources. In *Proceedings of the LREC 2004*, LREC 2004, Lisbon, Portugal.