UFRA: A UIMA-based Approach to Federated Language Resource Architecture

Riccardo Del Gratta, Roberto Bartolini, Tommaso Caselli, Monica Monachini

Claudia Soria and Nicoletta Calzolari

Istituto Di Linguistica Computazionale
Consiglio Nazionale delle Ricerche
Via Moruzzi 1, Pisa, Italy
{riccardo.delgratta, roberto.bartolini, tommaso.caselli, monica.monachini, claudia.soria, nicoletta.calzolari}@ilc.cnr.it

Abstract

In this paper we address the issue of developing an interoperable infrastructure for language resources and technologies. In our approach, called UFRA, we extend the Federate Database Architecture System adding typical functionalities coming from UIMA. In this way, we capitalize the advantages of a federated architecture, such as autonomy, heterogeneity and distribution of components, monitored by a central authority responsible for checking both the integration of components and user rights on performing different tasks. We use the UIMA approach to manage and define one common front-end, enabling users and clients to query, retrieve and use language resources and technologies.

The purpose of this paper is to show how UIMA leads from a FEDERATED DATABASE ARCHITECTURE to a FEDERATED RESOURCE ARCHITECTURE, adding to a registry of available components both static resources such as *lexicons* and *corpora* and dynamic ones such as tools and general purpose language technologies.

At the end of the paper, we present a case-study that adopts this framework to integrate the SIMPLE lexicon and TIMEML annotation guidelines to tag natural language texts.

1. Introduction and background work

The huge amount and diversity of language resources and tools, together with the availability of mature standards for content interoperability, suggests that the time is ripe for trying to weave the various resources scattered over different sites into a single organism of language services and repositories.

The integration and exploitation of language resources and tools into an architecture where users can combine elements of static language resources and dynamic processing resources is an active research topic being pursued at our site, both independently and in the framework of European and international projects. Our research group is involved in an ESFRI project, CLARIN (see section 3.1.), for the development of a pan-European integrated and interoperable infrastructure of language resources and technologies for Humanities and Social Science. Our institute is also partner of the international project Language Grid (see section 3.3.), that aims at realizing an infrastructure for inter-cultural collaboration based on the composition of language services. Language Grid defines requirements to "ontologize" language resources, their communication as well as their interaction protocols. We have met these requirements in the complete case study about the integration of the SIMPLE lexicon (Ruimy, 2006) and TIMEML (Pustejovsky et al., 2006), as reported in (Del Grattaet al., 2008); moreover the need of input and/or output format standardization, required by Language Grid, is at the base of the resource registry definition (see section 3.1.) which is fundamental also for the CLARIN project.

Standards are the precondition for content interoperability and for providing tools able to process data with standardized descriptions and return standardized output. Expectations of the scientific and industrial community about standards are that, once made operational in an integrated resource platform, they will be beneficial to the definition of both standardized access functions and automated workflows. The challenge for them is to enable a modern service-oriented infrastructure with a set of stable language services. It is worth noting that, at least initially, different output formats can be mapped onto the ISO standard, though not being properly standardized (see section 5.2.).

In this paper, we describe an approach to the realization of an integrated and interoperable infrastructure of language resources and technologies based on the UIMA architecture. We use the term "*language resource*" to refer both to static resources (lexicons, corpora) and dynamic processing resources (NLP tools, general purpose technologies).

In the following we present a brief description of the UIMA framework (section 1.1.) and of FEDERATE DATABASE SYSTEM, (FDBS, section 1.2.).

1.1. The UIMA framework

This section briefly describes the UIMA platform and philosophy. The interested reader can find more details in the following articles: (Ferrucci and Lally, 2004; Götz and Suhre, 2004).

UIMA deals at language resources as software "hooks" that can be "handled" by a common framework; the UIMA platform provides facilities for embedding tools and resources into an Integrated Development Environment, such as Eclipse[1] and defines an object, the Common Analysis Structure, CAS, which contains both the input physical data (document(s)), meta data and any annotation level added by linguistic tools. It provides cooperating UIMA components

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

with a common representation and mechanism for shared access to the document(s) being analyzed.
In a standard workflow, the CAS object runs from the input to the output steps and is accessed, manipulated and updated by each resource. This behavior of the CAS object allows developers to consider the UIMA workflow as an *assembly-line*. Developers, therefore, can choose the step as well as the conditions for a tool to be executed within the *assembly-line*. The UIMA framework manages resource integration by defining specific "descriptors", i.e. XML files. One of these files contains annotations performed on the content of the documents, while another contains the framework-provided infrastructure (primitive analysis engines) that allows them to be easily combined in a workflow (aggregate analysis engines).

1.2. A Federate approach to integrated resources

One approach to standardized and integrated resources management is made by the Federate Database Architecture System, FDBS (Heimbigner and McLeod, 1985; Sheth and Larson, 1990). This approach is preferred to a standard resource-sharing architecture, since the FDBS approach manages resource-sharing issues as well as users and roles definition. The FDBS clearly defines a central authority responsible for all the interoperability outcomes among components and for standardization of input/output formats as well as resource structure. This central authority oversees to the federation policies. Internal rules, groups and user rights, components cooperation are pillars for the FDBS architecture.
We point out three main characteristics an architecture should have to be considered "federable":

Heterogeneity: heterogeneity in resources is due to several factors. Among them, differences in structures and semantics of data;

Autonomy: in a few words, autonomy stands for the quality, for a resource, to function independently from others;

Distribution: resources exist before the federation is built. The federation aims at the interactions of each single resource with others, defining interoperability rules, access rights and a common access language.

2. UIMA approach to FDBS

The administration of a FDBS is a challenging task and the administration of a federation of resources is more than a challenging task. In this architecture, the central authority has to manage both user rights and resource interoperability. User rights join interoperability rules and define a complex scenario in which the 'motto' "*who can do what and how*" is the key question.
In our approach, called UFRA, we combine functionalities offered by the UIMA platform with FDBS features to define a clean *central authority* capable of addressing interoperability and user privileges issues.
In the following subsection, we list steps needed to set-up the architecture: the basic idea is the introduction of specialized repositories to manage language resources, Natural Language Processing (NLP) pipelines and user requirements defined in and for the federation.

2.1. Resources survey

The resource survey is the first step to be executed before the architecture is made up. We have defined six different repositories and pulled apart resources from workflow and user information. Resource types as well as their capabilities are stored in differentiated repositories. By *capability* we mean a specific functionality of one given resource and, by generalization, the piece of information added to the input document by that capability.

meta-repository: this repository contains information on each single resource. Each resource is assigned a unique persistent identifier. This repository keeps track of each *resource instance*: a *resource instance* is a physical copy of the resource identified by the unique persistent identifier;

resource type repository: this repository classifies the resources in different resource types in agreement with resource capability;[2]

resource-schema: this schema describes which kind of data is managed by a resource. Other elements of this schema manage input/output and standard compliance;

meta workflow schema: this schema represents possible workflows within the UFRA environment. One single workflow schema assembles a list of resource types ordered according to execution priorities. The meta workflow schema provides a skeleton for NLP pipeline. The advantage of this schema is that we can define workflows using resource type as building block. Sometimes, a workflow can be specified upon peculiar resources;

user rights repository: this repository contains privileges assigned to users and/or groups. User privileges are defined both at resource type and resource level. User rights (or privileges) have to be placed within an *Identity Management* scenario and addressed with specific technologies (Foundation, 2007; Erdos and Cantor, 2001);

federal dictionary: this is a specialized component, which regulates the UFRA topology. The federal dictionary is built upon other repositories described above and manages the resource taxonomy and interaction.

Figure 1 describes the interactions defined among the repositories.

3. Connection with other similar projects

This section lists two straightforward connections settled for the UFRA platform at our institute.

[2]Resource types can further be sub-typed: for example, if a Tagger-type-A resource has three annotation levels, Lemma,Pos and Morphological information and a Tagger-type-B has only two of the above levels, then the latter is a sub-type of the former.

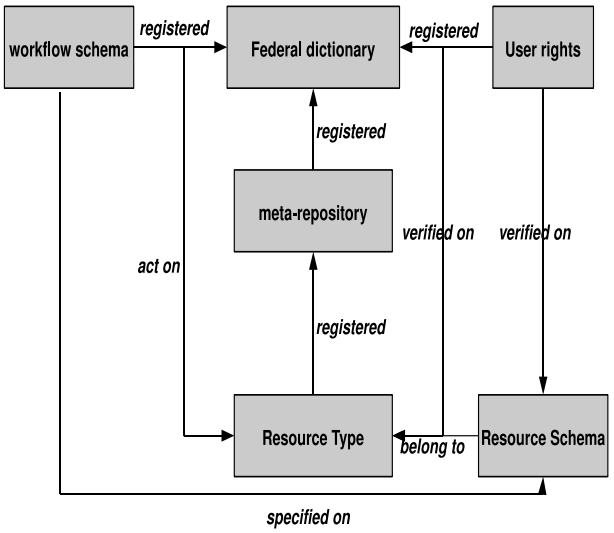

Figure 1: Interactions and relations at repository level.

3.1. Relevant CLARIN Experiences

As mentioned in section 1., our institute is member of an ESFRI project within the FP7, CLARIN (The Clarin Project, 2008), which aims at the definition of an architecture of interoperable language resources as well as the setting up of a network of repositories and service centers that keeps track of registered resources and user privileges. One of the pillars of CLARIN is, consequently, the setting-up of a *registry* of available resources. These resources are described using a set of metadata already defined and registered in specialized repositories. For details, see, among others, (Broeder and Wittenburg, 2006; Schäfer, 2007). A persistent unique identifier is assigned to each resource.
Users can query this registry to find out relevant resources for their researches. These resources are accessed by web services and/or organized in virtual collection to be reused.

3.2. UFRA: Resource Registry and Service Providing

In UFRA, we adopted the CLARIN strategy with respect to the *resource registry* setting up. Every repository described in section 2.1. is defined accordingly with metadata directives. As in CLARIN, this registry is the backbone for the UFRA architecture: it is used for both resource querying and services providing. Repositories defined above are internally accessed by the UIMA framework and, externally, by users who want to build their own resource collection relevant for their research.
It is straightforward to identify one single resource with a primitive analysis engine. These analysis engines can be deployed as web services, since this is one of the deployment options supported by the UIMA framework.

3.3. Connection with Language Grid

As mentioned in section 1., our institute cooperates with the National Institute of Information and Communications Technology (NICT) within the Language Grid project (Ishida, 2007). This project aims at the definition of a new infrastructure both to improve the use of existing language services and to encourage users to create new ones relevant for their needs. Web services, modelled on language resources, are the backbone of the Language Grid project: this technology makes it possible to combine already available resources over the Internet; moreover, information and data sharing is also managed and implemented by a strong use of web service technology.

3.4. Remarks

An infrastructure like UFRA is a challenging task from both a linguistic perspective and a technological point of view. Nevertheless, the strong connections settled with CLARIN and Language Grid encourage to persist in such an infrastructure. UFRA mixes identity management with access to distributed language resources via web services.

4. UIMA Role

This section describes the UIMA role within the UFRA platform.
UIMA is a framework designed to manage resource interoperability and integration in a corporate research environment. As explained in section 2., it is the obvious candidate to carry out the *role* of the *central authority* in a resource federation.
We can simply use the UIMA framework to access and manipulate documents. In this basic scenario, the CAS object is instantiated upon the document[3] and it is accessed, manipulated and updated by analysis engines.
In order to define the role of UIMA within the UFRA platform, we decided to add *operational annotations* to the whole document. In such a case, the CAS contains the document to be analyzed ***and*** the linguistic annotation a user wants, expressed as *operational annotations* defined at document level. Each *operational annotation* records linguistic annotations and/or language resources pertinent for these annotations.
UIMA is, then, a software interface between users and the *Fdbs*. It is responsible for accessing the federal dictionary, selecting the right resources to perform linguistic annotations.

5. UIMA Type System for document annotation

This section introduces the UIMA type system for document annotation. Briefly, a type system is a schema or a model for the CAS object. It defines the types of objects and their features (capabilities) that may be used by a CAS. Analysis engines conform to a type system.
The UIMA out-of-the-box framework provides a simple Document Type System (DTS). We enriched this DTS adding two features: *typeOfOperation* and *preferredResource*. We have introduced these two features to define NLP operations and available resources. These features are directly linked to two type systems, OPERATION and RESOURCE, that include range types which can restrict the value of possible operation and/or resources a user can select.

[3] If a document belongs to a collection, the UIMA framework iterates on the collection and instantiates a CAS for each single document.

Language: a string identifying the language of the input text;

Operation: a type system used to model CAS objects built upon a typical NLP operation;

Resource: a type system used to model CAS objects built upon language resources;

typeOfOperation: a feature array linked to the OPERATION type system. It is used to pick-up from a list of available NLP operations;

preferredResource: a feature array linked to the RESOURCE type system. It is used to choose a particular resource.

Figure 2 describes the DTS scheme implemented as UIMA Type System.

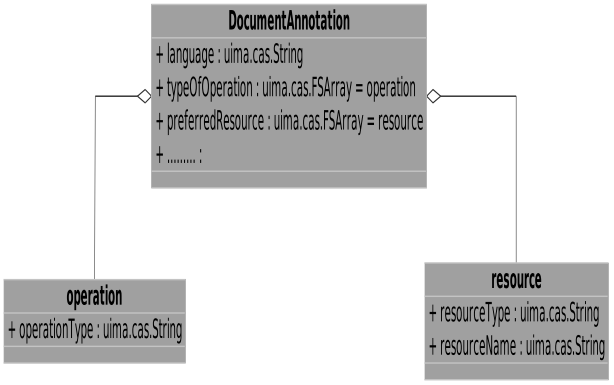

Figure 2: Document annotation scheme

5.1. From Data to Pipeline Driven Architecture

The document level annotation (DTS), defined in section 5., moves the architecture of interoperable language resources from (typically) data-driven architecture to a (NLP) pipeline-driven one. UIMA, by its very nature, is a data-driven architecture, but the DTS, contained in the *starting CAS* object [4], tells the UIMA framework which NLP operations perform rather than which annotation add to the input document. This reversal of perspective moves the annotation process from a standard inline/outline process to an NLP resource execution one. Users choose annotation levels and preferred language resources to carry out these annotations. These requirements set the values of *typeOfOperation* and *preferredResource* DTS features.
The workflow implemented in UFRA is, however, standard. The UIMA platform accesses the *starting CAS* and compares its information with the federal dictionary. At this point UIMA behaves just like a *central authority*: it checks whether selected NLP operations and preferred language resources are consistent with each other and tries to build an NLP pipeline according to user requests. The pipeline is concretely assembled by aggregating primitive analysis engine into an aggregate analysis engine. The aggregate analysis engine is made on meta workflow schema.

[4]By *starting CAS* we mean the CAS built on input document.

In this scenario we can reasonably talk of an NLP pipeline-driven architecture. The following section, 5.3., describes further this architecture, introducing the concepts of NLP atomic operations.

5.2. Input/output standardization

The federal dictionary as well as the other repositories define a resource classification and standardization. We consider another kind of standardization that plays a crucial role in resource interoperability. We refer to input/output standardization. UIMA components declare their input and output with respect to a type system, therefore we need a standard format to address data transfer from one resource to another. The UIMA platform provides an out-of-the-box solution of this standardization: the inline annotation is performed at the CAS level. This annotation is saved in feature-like format, indexed and managed by the UIMA platform. However, the content of the CAS can be serialized and accessed by external tools as well as stored in databases. In this way we can access, on demand, standoff annotation. From a linguistic perspective, the standardization is achieved by using the data category registry (DCR) (Wright, 2004). This DCR behaves as a *graph* central point other format are mapped onto (Nancy and Keith, 2007).

5.3. NLP atomic operations and pipelines

This section defines NLP atomic operations and (abstract) NLP pipelines. In previous sections we have shown the conditions to build an NLP pipeline from user requested annotations. On the other hand, UIMA provides primitive analysis engines and aggregate ones. It is straightforward to identify a primitive engine with an NLP atomic operation, while aggregate engines are identified as NLP pipeline.
The federal dictionary allows for a generalization of an NLP pipeline from one built upon single resources to one built on resource types . In such a case the NLP pipeline is abstract, since used resources are not physical ones, but only generic resources of a specific type. The meta workflow schema is responsible to record typical NLP pipeline. UIMA renders this schema and pipes each single resource to another. Figure 3 shows single modules or atomic NLP operations piped one to another to define the final NLP pipeline.

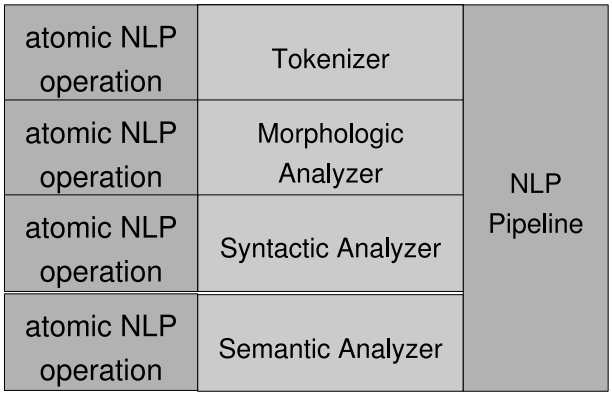

Figure 3: Abstract NLP Pipeline and atomic operations

The meta workflow schema assembles each module in an NLP pipeline, accordingly to *operations* and/or *resources* selected by a user. This pipeline is capable to provide different annotation levels: *words (tokens)*, their *morphology*, how they behave from a *syntactic perspective* and, finally, when available, their *semantics*.
Each annotation level is carried out by specific modules: *Tokenizers*, *Morphological Analyzers*, *Syntactic Analyzers* and *Semantic Analyzers*, respectively.
Linguistic annotations flow from module to module following UIMA component specifications summarized in the UIMA type systems and CAS object.
Each annotation is maintained in the CAS object. This guarantees that NLP modules work on the same token and/or the same span of text.

5.4. Linguistic Issues

The UFRA framework supports the ability to define spans of text that can be analyzed is a single "spot". This is useful when two different spans of texts, e.g. two sentences, overlap one another, partly overlap and so on, and annotation tags carried out for tokens in a sentence depend on tags carried out on other tokens in the same sentence and/or on tokens in the other sentence. This typical linguistic issue is addressed UFRA in different ways: a standoff annotation and other more complex situations, such as span texts analyzed in a single spot. The standoff annotation is managed by using the CAS consumer techniques (Götz and Suhre, 2004) that allows for saving texts and their annotations in a storage repository such as databases, files. On the other hand, the single spot annotation is managed directly in memory by using the features-like techniques implemented both in CAS and in type systems.

6. Case Study

A first case-study in the perspective of the work described so far is represented by the mapping (Del Grattaet al., 2008) between the TIMEML event classes and the SIMPLE lexicon. In this case-study an input text is analyzed by a *Temporal Annotator* which searches the TIMEML category by integrating the SIMPLE lexicon with a set of heuristic-based rules to tag a given word sense with the correct TIMEML category.
As reported in (Del Grattaet al., 2008), in this case study we adopted the inline annotation techniques. We defined a *sentence annotator* capable for managing each token in the sentence, in order to use annotations defined for previous or next tokens with respect to the current token analyzed. This strategy allows the heuristics to access each annotation level in the sentence. Some annotations can be used while others can not. This case study shows how the pipeline-drive architecture actually works:

Document level annotation scheme The user selects the language (Italian), the annotation levels (temporal annotation) and the lexicon (SIMPLE) to start the temporal annotation. The UFRA framework matches these requirements against the rule-based repositories to verify their compatibility;

NLP Pipeline Setup Information retrieved from the repositories lead the platform to built the NLP pipeline which better meets the user requirements. Resulted NLP pipeline is capable to add the temporal annotation levels to the input document(s) integrating information from SIMPLE and a heuristics tableau.

NLP Execution The NLP pipeline built by UFRA is, finally, executed in the UIMA platform by means of specific software methods: the CAS process methods, defined for the selected aggregate analysis engines, are responsible to call appropriate resource(s) for the resulted NLP pipeline as well as to manage data structures previously created on documents. In the case-study, UIMA temporal annotator is an Aggregate Analysis Engine which requires a Sentence annotator and a Tokenizer. The CAS process method executes these tools to annotate the input text with results retrieved from the heuristics tableau.

7. Conclusion

We have presented UFRA, a UIMA approach to FDBS. The UIMA platform is responsible for the definition of a set of Type Systems modeled on resource types instead of on specific resource instance, e.g: the generic *Tagger* instead of that specific tagger. The rules to integrate and to merge language resources have been stored in a central rule-based repository which is responsible to verify the user's requirements against these rules and to build the NLP pipeline that better meets the requirements.
The CAS objects, defined by these Type Systems, index, access, manage and annotate the input document(s) according to the capabilities of the original resource types and behave as if they were *instances* of the selected language resources.
The resulting architecture is a pipeline-driven one, since language resources are assembled and merged according to NLP pipeline defined upon user requirements and resources structure.

8. Acknowledgements

Part of this work is carried out as part of the CLARIN project, an EU STREP project funded under the FP7.
We would also like to thank the three anonymous reviewers for their precious comments and suggestions.

9. References

R. Del Gratta, T. Caselli, I. Prodanof, N. Ruimy, and N. Calzolari. 2008. TimeML: An ontological mapping onto UIMA type systems. In *ICGL2008: First International Conference on Global Interoperability for Language Resources.*, Hong Kong.

The Clarin Project. 2008. Common language resources and technology infrastructure. http://www.clarin.eu/.

D. Broeder and P. Wittenburg. 2006. The IMDI metadata framework, its current application and future direction. *IJMSO*.

M. Erdos and S. Cantor. 2001. The Shibboleth architecture. http://shibboleth.internet2.edu/.

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

OpenLdap Foundation. 2007. Open ldap. http://www.openldap.com/.

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

D. Heimbigner and D. McLeod. 1985. A federated architecture for information management. *ACM Trans. Inf. Syst.*, 3(3):253–278.

T. Ishida. 2007. Nict, language grid & department of social informatics, kyoto university. http://langrid.nict.go.jp/.

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

J. Pustejovsky, J. Littman, B. Knippen, R. Gaizauskas, A. Setzer, and R. Saurí. 2006. TimeML annotation guidelines. *http://www.timeml.org/site/publications/specs.html*.

N. Ruimy. 2006. Computational multi-layered italian lexicon for hlt applications. In *Proceedings XII EURALEX International Congress, Atti del Congresso Internazionale di Lessicografia*.

U. Schäfer. 2007. *Integrating Deep and Shallow Natural Language Processing Components - Representations and Hybrid Architectures.*

A. P. Sheth and J. A. Larson. 1990. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236.

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