A Linguistic Service Ontology for Language Infrastructures

Yoshihiko Hayashi
Graduate School of Language and Culture, Osaka University
1-8 Machikaneyama-cho, Toyonaka, 560-0043 Japan
hayashi@lang.osaka-u.ac.jp

Abstract

This paper introduces conceptual framework of an ontology for describing linguistic services on network-based language infrastructures. The ontology defines a taxonomy of processing resources and the associated static language resources. It also develops a sub-ontology for abstract linguistic objects such as expression, meaning, and description; these help define functionalities of a linguistic service. The proposed ontology is expected to serve as a solid basis for the interoperability of technical elements in language infrastructures.

1 Introduction

Several types of linguistic services are currently available on the Web, including text translation and dictionary access. A variety of NLP tools is also available and public. In addition to these, a number of community-based language resources targeting particular domains of application have been developed, and some of them are ready for dissemination. A composite linguistic service tailored to a particular user's requirements would be composable, if there were a language infrastructure on which elemental linguistic services, such as NLP tools, and associated language resources could be efficiently combined. Such an infrastructure should provide an efficient mechanism for creating workflows of composite services by means of authoring tools for the moment, and through an automated planning in the future.

To this end, technical components in an infrastructure must be properly described, and the semantics of the descriptions should be defined based on a shared ontology.

2 Architecture of a Language Infrastructure

The linguistic service ontology described in this paper has not been intended for a particular language infrastructure. However we expect that the ontology should be first introduced in an infrastructure like the Language Grid [1], because it, unlike other research-oriented infrastructures, tries to incorporate a wide range of NLP tools and community-based language resources (Ishida, 2006) in order to be useful for a range of intercultural collaboration activities.

The fundamental technical components in the Language Grid could be: (a) external web-based services, (b) on-site NLP core functions, (c) static language resources, and (d) wrapper programs.

Figure 1 depicts the general architecture of the infrastructure. The technical components listed above are deployed as shown in the figure.

Computational nodes in the language grid are classified into the following two types as described in (Murakami et al., 2006).

- A *service node* accommodates atomic linguistic services that provide functionalities of the NLP tool/system running on a node, or they can simply have a wrapper program that consults an external web-based linguistic service.
- A *core node* maintains a repository of the known atomic linguistic services, and provides service discovery functionality to the possible users/applications. It also maintains a workflow re-

[1] Language Grid: http://langrid.nict.go.jp/

Proceedings of the ACL 2007 Demo and Poster Sessions, pages 145–148,
Prague, June 2007.

pository for composite linguistic services, and is equipped with a workflow engine.

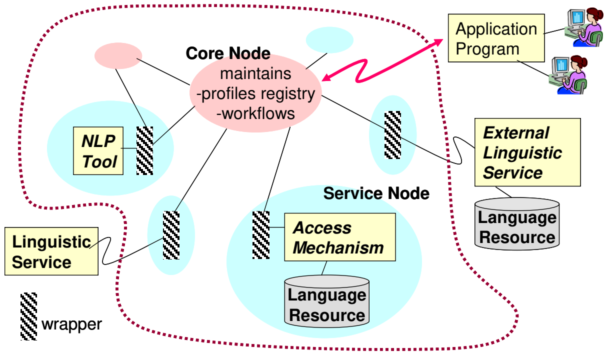

Figure 1. Architecture of a Language Infrastructure.

Given a technical architecture like this, the linguistic service ontology will serve as a basis for composition of composite linguistic services, and efficient wrapper generation. The wrapper generation processes are unavoidable during incorporation of existing general linguistic services or dissemination of newly created community-based language resources. Tthe most important desideratum for the ontology, therefore, is that it be able to specify the input/output constraints of a linguistic service properly. Such input/output specifications enable us to derive a taxonomy of linguistic service and the associated language resources.

3 The Upper Ontology

3.1 The top level

We have developed the upper part of the service ontology so far, and have been working on detailing some of its core parts. Figure 2 shows the top level of the proposed linguistic service ontology.

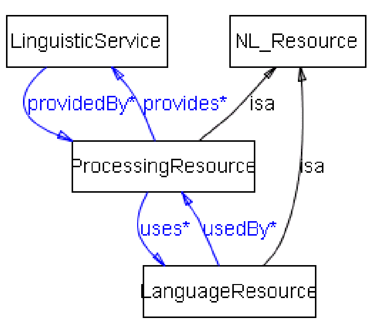

Figure 2. The Top Level of the Ontology.

The topmost class is **NL_Resource**, which is partitioned into **ProcessingResource**, and **LanguageResource**. Here, as in GATE (Cunningham, 2002), processing resource refers to programmatic or algorithmic resources, while language resource refers to data-only static resources such as lexicons or corpora. The innate relation between these two classes is: a processing resource can use language resources. This relationship is specifically introduced to properly define linguistic services that are intended to provide access functions to language resources.

As shown in the figure, **LinguisticService** is *provided by* a processing resource, stressing that any linguistic service is realized by a processing resource, even if its prominent functionality is accessing language resources in response to a user's query. It also has the meta-information for advertising its non-functional descriptions.

The fundamental classes for abstract linguistic objects, **Expression**, **Meaning**, and **Description** and the innate relations among them are illustrated in Figure 3. These play roles in defining functionalities of some types of processing resources and associated language resources. As shown in Fig. 3, an expression may *denote* a meaning, and the meaning can be further *described by* a description, especially for human uses.

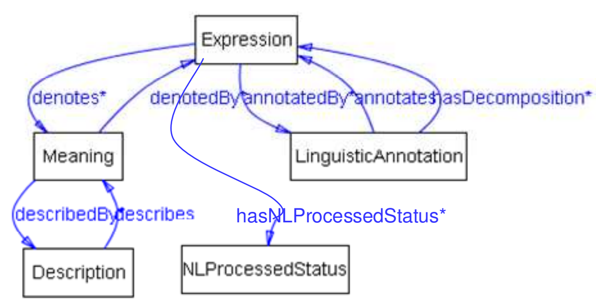

Figure 3. Classes for Abstract Linguistic Objects.

In addition to these, **NLProcessedStatus** and **LinguisticAnnotation** are important in the sense that NLP status represents the so-called IOPE (Input-Output-Precondition-Effect) parameters of a linguistic processor, which is a subclass of the processing resource, and the data schema for the results of a linguistic analysis is defined by using the linguistic annotation class.

3.2 Taxonomy of language resources

The language resource class currently is partitioned into subclasses for **Corpus** and **Dictionary**. The immediate subclasses of the dictionary class are: (1) **MonolingualDictionary**, (2) **Bi-**

lingualDictionary, (3) **Multilingual-Terminology**, and (4) **ConceptLexicon**. The major instances of (1) and (2) are so-called machine-readable dictionaries (MRDs). Many of the community-based special language resources should fall into (3), including multilingual terminology lists specialized for some application domains. For subclass (4), we consider the computational concept lexicons, which can be modeled by a WordNet-like encoding framework (Hayashi and Ishida, 2006).

3.3 Taxonomy of processing resources

The top level of the processing resource class consists of the following four subclasses, which take into account the input/output constraints of processing resources, as well as the language resources they utilize.

- **AbstractReader**, **AbstractWriter**: These classes are introduced to describe computational processes that convert to-and-from non-textual representation (e.g. speech) and textual representation (character strings).
- **LR_Accessor**: This class is introduced to describe language resource access functionalities. It is first partitioned into **CorpusAccessor** and **DictionaryAccessor**, depending on the type of language resource it accesses. The input to a language resource accessor is a query (**LR_AccessQuery**, sub-class of **Expression**), and the output is a kind of 'dictionary meaning' (**DictionaryMeaning**), which is a sub-class of meaning class. The dictionary meaning class is further divided into sub-classes by referring to the taxonomy of dictionary.
- **LinguisticProcessor**: This class is further discussed in the next subsection.

3.4 Linguistic processors

The linguistic processor class is introduced to represent NLP tools/systems. Currently and tentatively, the linguistic processor class is first partitioned into **Transformer** and **Analyzer**.

The transformer class is introduced to represent **Paraphrasor** and **Translator**; both rewrite the input linguistic expression into another expression while maintaining the original meaning. The only difference is the sameness of the input/output languages. We explicitly express the input/output language constraints in each class definition.

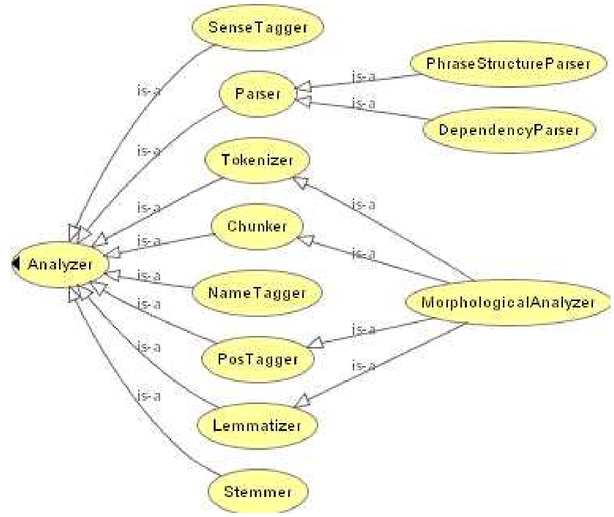

Figure 4. Taxonomy of Linguistic Analyzer.

Figure 4 shows the working taxonomy of the analyzer class. While it is not depicted in the figure, the input/output constraints of a linguistic analyzer are specified by the **Expression** class, while its precondition/effect parameters are defined by **NLProcessedStatus** class. The details are also not shown in this figure, these constraints are further restricted with respect to the taxonomy of the processing resource.

We also assume that any linguistic analyzer additively annotates some linguistic information to the input, as proposed by (Cunningham, 2002), (Klein and Potter, 2004). That is, an analyzer working at a certain linguistic level (or 'depth') adds the corresponding level of annotations to the input. In this sense, any natural language expression can have a layered/multiple linguistic annotation. To make this happen, a linguistic service ontology has to appropriately define a sub-ontology for the linguistic annotations by itself or by incorporating some external standard, such as LAF (Ide and Romary, 2004).

3.5 NLP status and the associated issues

Figure 5 illustrates our working taxonomy of NLP processed status. Note that, in this figure, only the portion related to linguistic analyzer is detailed. Benefits from the NLP status class will be twofold: (1) as a part of the description of a linguistic analyzer, we assign corresponding instances of this class as its precondition/effect parameters, (2) any instance of the expression class can be concisely ‘tagged’ by instances of the NLP status class, according to how ‘deeply’ the expression has been linguistically analyzed so far. Essentially, such information can be retrieved from the attached linguistic annotations. In this sense, the NLP status class might be redundant. Tagging an instance of expression in that way, however, can be reasonable: we can define the input/output constraints of a linguistic analyzer concisely with this device.

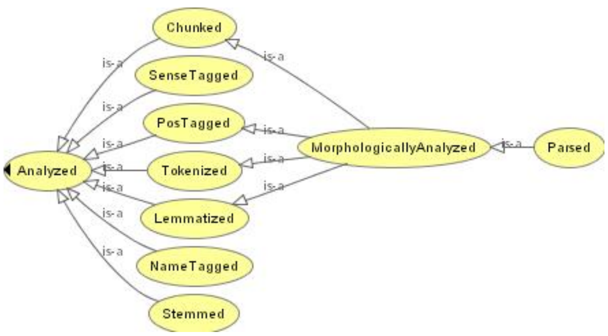

Figure 5. Taxonomy of NLP Status.

Each subclass in the taxonomy represents the type or level of a linguistic analysis, and the hierarchy depicts the processing constraints among them. For example, if an expression has been parsed, it would already have been morphologically analyzed, because parsing usually requires the input to be morphologically analyzed beforehand. The subsumption relations encoded in the taxonomy allow simple reasoning in possible composite service composition processes. However note that the taxonomy is only preliminary. The arrangement of the subclasses within the hierarchy may end up being far different, depending on the languages considered, and the actual NLP tools, these are essentially idiosyncratic, that are at hand. For example, the notion of ‘chunk’ may be different from language to language. Despite of these, if we go too far in this direction, constructing a taxonomy would be meaningless, and we would forfeit reasonable generalities.

4 Related Works

Klein and Potter (2004) have once proposed an ontology for NLP services with OWL-S definitions. Their proposal however has not included detailed taxonomies either for language resources, or for abstract linguistic objects, as shown in this paper. Graça, et al. (2006) introduced a framework for integrating NLP tools with a client-server architecture having a multi-layered repository. They also proposed a data model for encoding various types of linguistic information. However the model itself is not ontologized as proposed in this paper.

5 Concluding Remarks

Although the proposed ontology successfully defined a number of first class objects and the innate relations among them, it must be further refined by looking at specific NLP tools/systems and the associated language resources. Furthermore, its effectiveness in composition of composite linguistic services or wrapper generation should be demonstrated on a specific language infrastructure such as the Language Grid.

Acknowledgments

The presented work has been partly supported by NICT international joint research grant. The author would like to thank to Thierry Declerck and Paul Buitelaar (DFKI GmbH, Germany) for their helpful discussions.

References

H. Cunningham, et al. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proc. of ACL 2002*, pp.168-175.

J. Graça , et al. 2006. NLP Tools Integration Using a Multi-Layered Repository. P*roc. of LREC 2006 Workshop on Merging and Layering Linguistic Information.*

Y. Hayashi and T. Ishida. 2006. A Dictionary Model for Unifying Machine Readable Dictionaries and Computational Concept Lexicons. *Proc. of LREC 2006*, pp.1-6.

N. Ide and L. Romary. 2004. International Standard for a Linguistic Annotation Framework. *Journal of Natural Language Engineering*, Vol.10:3-4, pp.211-225.

T. Ishida. 2006. Language Grid: An Infrastructure for Intercultural Collaboration. *Proc.* of *SAINT-06*, pp. 96-100, keynote address.

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

Y. Murakami, et al. 2006. Infrastructure for Language Service Composition. *Proc. ofçSecond International Conference on Semantics, Knowledge, Grid.*