# Classifying Standard Linguistic Processing Functionalities based on Fundamental Data Operation Types

**Yoshihiko Hayashi and Chiharu Narawa**

Graduate School of Language and Culture, Osaka University
1-8 Machikaneyama, Toyonaka, Osaka, 5600043 Japan
hayashi@lang.osaka-u.ac.jp, narawa@lang.osaka-u.ac.jp

**Abstract**

It is often argued that a set of standard linguistic processing functionalities should be identified, with each of them given a formal specification. We would benefit from the formal specifications; for example, the semi-automated composition of a complex language processing workflow could be enabled in due time. This paper extracts a standard set of linguistic processing functionalities and tries to classify them formally. To do this, we first investigated prominent types of language Web services/linguistic processors by surveying a Web-based language service infrastructure and published NLP toolkits. We next induced a set of standard linguistic processing functionalities by carefully investigating each of the linguistic processor types. The standard linguistic processing functionalities was then characterized by the input/output data types, as well as the required data operation types, which were also derived from the investigation. As a result, we came up with an ontological depiction that classifies linguistic processors and linguistic processing functionalities with respect to the fundamental data operation types. We argue that such an ontological depiction can explicitly describe the functional aspects of a linguistic processing functionality.

**Keywords:** language service; language processing functionality; linguistic data type; language service ontology

## 1. Introduction

It is often argued that a set of standard linguistic processing functionalities should be identified (Hayashi, 2011), with each of them given a formal specification. We would benefit from the formal specifications; for example, the semi-automated composition of a complex language processing workflow could be enabled in due time. Then, it is natural to ask, how can we extract a reasonable set of standard linguistic processing functionalities, and how can we formally specify each of them?

Because a linguistic processing functionality can be partly specified by the input/output data types, we carefully looked at the input/output data specification of tools/libraries provided by well-known Natural Language Processing (NLP) toolkits, as well as published language Web services. However the input/output data type is not the only aspect that fully characterizes a linguistic processing functionality. Therefore we further investigated a variety of fundamental data operation types that should be required to achieve a linguistic processing functionality.

Putting these investigations together, we came up with a three-layer model that can serve as a conceptual framework for classifying a standard set of linguistic processing functionalities. In this paper, we provide a sketch of an ontological depiction for specifying a standard set of linguistic processing functionalities based on the three-layered model. We also try to validate primary data types in NLP, which also have been derived from the investigation, while arguing that our proposal may be simple yet effective toward a formal specification of the standard linguistic processing functionalities.

## 2. Sources of the Investigation

To first identify a set of prominent linguistic processor types and the associated linguistic processing functionalities, we investigated a Web-based language service infrastructure and several NLP toolkits.

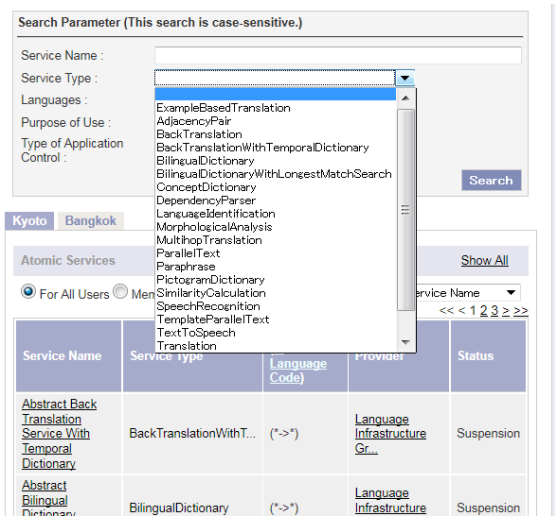


Figure 1: Language services in the Language Grid

### 2.1. The Language Grid

The Language Grid (Ishida, 2011)[1] is a collaborative language service infrastructure on the Web. Aimed at supporting activities of intercultural collaboration, it provides access to a range of Web-service-based language resources and linguistic processors. The Web site[2] gives a complete list of language service types and actual services registered as displayed in Figure 1. As of Feb 15, 2012, 23 different service types are available.

[1] http://langrid.org/
[2] http://langrid.org/service_manager/

Table 1: Major Language Grid service types classified into five groups

| Service Type group | Representative Service Types |
|---|---|
| Media Conversion | Speech Recognition, Text-To-Speech |
| Translation | Translation, Paraphrase |
| Linguistic Analysis | Language Identification, Morphological Analysis, Dependency Parser |
| Language Resource Access | Adjacency Pair, Bilingual Dictionary, Concept Dictionary, Pictogram Dictionary, Parallel Text |
| Miscellaneous | Quality Estimation, Similarity Calculation |

Table 2: Major processor types identified from the NLP toolkits

| *Linguistic Processor Type* | OpenNLP | Stanford NLP | FreeLing | LingPipe |
|---|---|---|---|---|
| Language Identifier | | | ✓ | ✓ |
| Sentence Splitter | ✓ | ✓ | ✓ | ✓ |
| Tokenizer | ✓ | ✓ | ✓ | ✓ |
| POS Tagger | ✓ | ✓ | ✓ | ✓ |
| Lemmatizer | | ✓ | ✓ | |
| Morphological Analyzer | | | ✓ | |
| Sense Tagger | | | ✓ | ✓ |
| Chunker | ✓ | | ✓ | |
| NE Classifier | ✓ | ✓ | ✓ | ✓ |
| Coreference Resolver | ✓ | ✓ | ✓ | |
| Dependency Parser | | ✓ | ✓ | |
| Phrase Structure Parser | ✓ | ✓ | ✓ | |

Although the Web site simply provides a flat list of language service types, we have grouped the major language service types into five groups, as shown in Table 1, out of which Media Conversion, Translation, and Linguistic Analysis are considered in the rest of this paper.

The services classified as Media Conversion perform so-called media conversion to/from text strings, and the Translation services project the meaning expressed in the input text string onto the output text string, which differs from the input string with regard to language and/or style. By its purpose and nature, the Language Grid provides a variety of Translation services, which include Multi-hop Translation for cascaded translation, and Back Translation, which can be employed as a useful tool for assessing the translation qualities.

Currently, two types of Linguistic Analysis are provided in the Language Grid: Morphological Analyzer and Dependency Parser for Japanese. We should remark here that morpho-syntactic analysis is required to tokenize, POS-tag, and lemmatize Japanese sentences. Therefore, the tokenization functionality, for example, is typically implemented by a morphological analyzer, which also performs POS tagging as well as lemmatization.

In the Web site, WSDL documents for describing these service types are publicized so that the user can invoke a specific service that can achieve his/her goal by relying on the SOAP service invocation protocol. However, as frequently argued, a WSDL document cannot formally specify the meaning of the input/output data of a Web service (Fensel et al., 2011). This surely is a potential burden for (semi-)automatically composing a composite Web service.

### 2.2. NLP toolkits

Table 2 displays a set of linguistic processor types, each of them has been extracted by surveying the documentation of the following well-known NLP toolkits: Apache OpenNLP[3], Stanford Core NLP[4], FreeLing[5], and LingPipe[6]. The table thus shows which processor type is provided by each toolkit.

Note that the names of the linguistic processor types were chosen by the authors of this paper while reviewing the documentation of each toolkit, thus they may not match the original names used by the toolkits. For example, in FreeLing, "Sentence Detection" is the name for the Sentence Splitter processor type in the table.

It should also be noted that, in the investigation, we particularly focused on linguistic analysis functionalities, leaving out of the table other (somewhat application-oriented and/or ad-hoc) functionalities provided by some of the toolkits. Among the mentioned NLP toolkits, FreeLing provides a variety of processors including modules for performing tasks of statistical machine learning.

## 3. A Linguistic Processor *implements* Linguistic Processing Functionalities

By combining the results shown in the Table 1 and Table 2, we consider the following as the current set of standard linguistic processor types. Note that the set is never frozen

[3] http://opennlp.apache.org

[4] http://nlp.stanford.edu/software/corenlp.shtml

[5] http://nlp.lsi.upc.edu/freeling/

[6] http://alias-i.com/lingpipe/

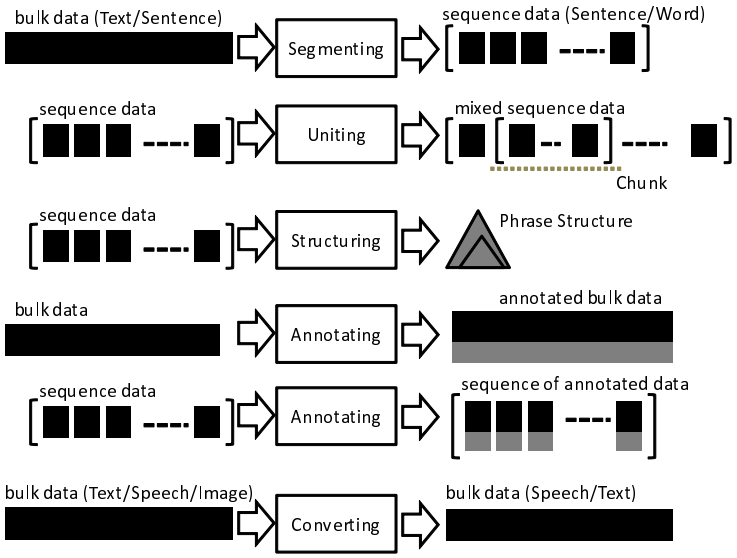


Figure 2: Fundamental data operation types

and may expand as new types are discovered.

Linguistic Processor Types::={Language Identifier, Sentence Splitter, Tokenizer, POS Tagger, Lemmatizer, Sense Tagger, Morphological Analyzer, Chunker, NE Recognizer, Coreference Resolver, Dependency Parser, Phrase Structure Parser, Speech Recognizer, Text-To-Speech Converter, Translator, Paraphraser}

Further, we have constructed a set of linguistic processing functionalities as follows.

Linguistic Processing Functionality Types ::= {Language Identification, Sentence Splitting, Tokenization, POS Tagging, Lemmatization, Sense Tagging, Chunking, NE Classification, Coreference Resolution, Dependency Parsing, Phrase Structure Parsing, Speech Recognition Text-To-Speech Conversion, Translation, Paraphrasing}

By comparing the two sets, it is easily noticed that a linguistic processor type, in general, implements a linguistic processing functionality, whereas two of them, which are underlined in the processor type set, implement multiple functionalities, and the corresponding names do not appear in the functionality set. The relationships are as follows:

- Morphological Analyzer *implements* Tokenization, POS Tagging, and Lemmatization;
- NE Recognizer *implements* Chunking and NE Classification.

## 4. A Linguistic Processing Functionality is *achieved by* Fundamental Data Operations

By carefully examining the input/output data specification of the published NLP tools/libraries/services mentioned so far, we have induced a fundamental set of data operation types that can be employed for classifying standard linguistic processing functionalities.

Figure 2 schematizes the data operation types characterized by the input/output abstract data types. Table 3 classifies which linguistic processing functionality type is achieved by which data operation type.
Note that each linguistic processing functionality, in general, is associated with a fundamental data operation type, whereas Lemmatization involves two operation types, Annotating and Converting. That is, a lemma form is generated from the word token, which would then be incorporated into the annotation, probably using a `lemma` tag.
Below, we describe each of the fundamental data operation types in turn.

**Segmenting:** A linguistic processing functionality assuming this operation type performs segmentation of the input bulk data into a sequence of the segmented elements. The linguistic processing functionality types achieved by this operation type are Sentence Splitting and Tokenization. The former splits a text string into a set of sentences, whereas the latter splits a sentence string into a set of word tokens. We therefore recognize `Text`, `Sentence`, and `Word` as elements of the *NLP primary data types*.

**Uniting:** In contrast to the segmenting operation, this operation type identifies a set of non-overlapping continuous regions from a sequence of primary data elements; these regions are usually referred to as chunks. We thus add `Chunk` as one of the NLP primary data types. Note that the resulting output can be a mixed sequence of primary data elements and sequences of primary data elements (that is, a nested sequence). The only linguistic processor functionality type achieved by this operation type is Chunking.

**Structuring:** For the moment, this operation type is performed solely by the Phrase Structure Parsing functionality, whose input is a sequence of primary data (`Word` or `Chunk`), and the output is a `Phrase Structure`, which is also nominated as one of the primary data types. Usually a phrase structure is represented by a tree structure, constrained by the following relationship between input and output: every element in the input sequence should appear

Table 3: Linguistic processing functionalities classified by data operation types

| *Linguistic Processing Functionality* | Segmenting | Uniting | Structuring | Annotating | Converting |
|---|---|---|---|---|---|
| Language Identification | | | | ✓ | |
| Sentence Splitting | ✓ | | | | |
| Tokenization | ✓ | | | | |
| POS Tagging | | | | ✓ | |
| Lemmatization | | | | ✓ | ✓ |
| Sense Tagging | | | | ✓ | |
| Chunking | | ✓ | | | |
| NE Classification | | | | ✓ | |
| Coreference Resolution | | | | ✓ | |
| Dependency Parsing | | | | ✓ | |
| Phrase Structure Parsing | | | ✓ | | |
| Speech Recognition | | | | | ✓ |
| Text-to-Speech Conversion | | | | | ✓ |
| Translation | | | | | ✓ |
| Paraphrasing | | | | | ✓ |

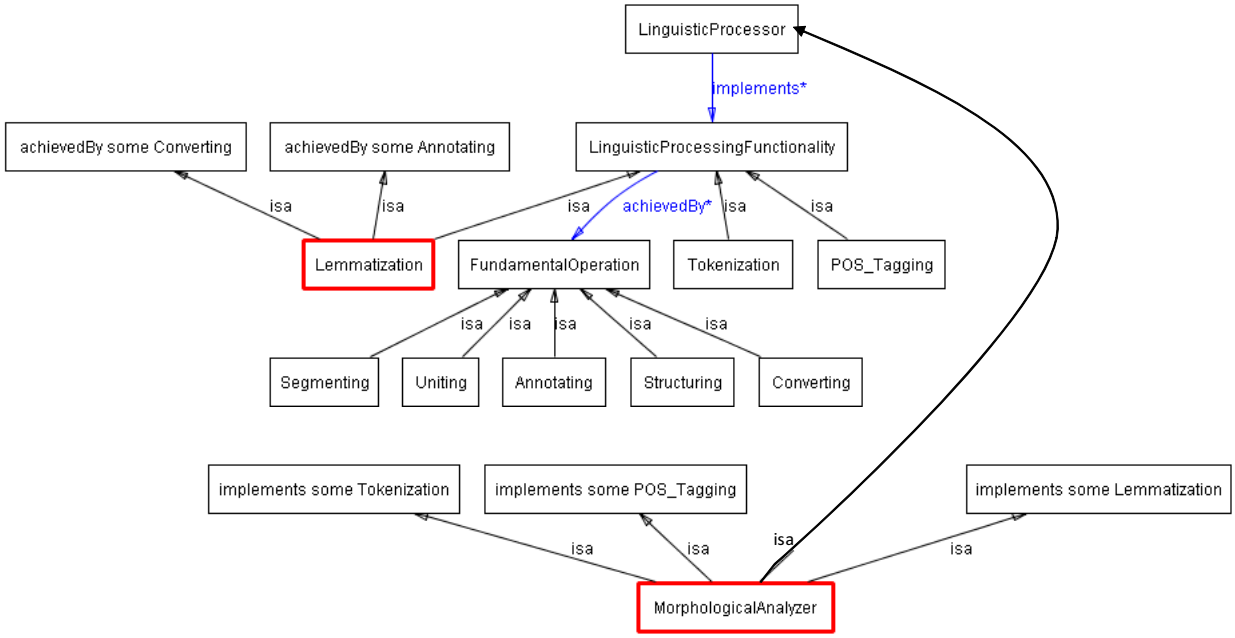


Figure 3: Ontological depiction for specifying linguistic processing functionalities and the associated elements.

as a leaf node of the phrase structure tree.

**Annotating:** As shown in Figure 2, this operation type is embodied in two ways: (a) annotating bulk data, or (b) annotating each element in the sequence data. A broad range of linguistic processing functionality types are classified as being achieved by this operation type. For example, Language Identification performs bulk data annotation (assigns the language name to a given text), while POS Tagging and Dependency Parsing perform sequence data annotation. We distinguish `Annotation` as a special abstract data type that can modify any NLP primary data type. This category is further discussed in the next section.

**Converting:** In principle, this operation type achieves the generation of output bulk data from the input bulk data. An important principle of Converting is that the input and output data should be different in some aspect. For example, the functionality type Text-to-Speech Conversion generates `Speech` data, represented with some encoding schema, from the input `Text`. A prominent linguistic processing functionality type achieved by this category is obviously Translation, which generates text in the designated target language from the input text in a specified source language, while retaining the meaning expressed in the input text.

## 5. A Three-layered Model for Specifying Linguistic Processing Functionalities

Based on the above findings and studies, a model is presented that consists of the three layers of linguistic processor type, linguistic processing functionality, and fundamental data operations as a conceptual framework for specifying a standard set of linguistic processing functionalities. These levels are summarized as follows.

- A linguistic processor *implements* one or more linguistic processing functionalities.
- A linguistic processing functionality is *achieved by* one or more fundamental data operations.
- Fundamental data operations are classified into the modeled types shown in Figure 2.
- Linguistic processors and linguistic processing functionalities are partly characterized by the input/output data types, which can be abstracted by the NLP primary data types and the Annotation data type.

Based on this model, Figure 3 provides the ontological depiction[7], which graphically represents the results of this study. In the figure, only the classes around Lemmatization and Morphological Analyzer are detailed. It can be verbalized as follows.

- A morphological analyzer is a kind of linguistic processor which implements Tokenization, Lemmatization, and POS Tagging, which are linguistic processing functionalities.
- The linguistic functionality of Lemmatization is achieved by Converting and Annotating, which are fundamental data operation types.

## 6. Remark about the `Annotation` abstract data type

`Annotation` is not a primary data type, rather it is a distinguished abstract data type that can modify any primary data. Although this paper does not deal with any mappings between abstract types and the actual data structure types, the class of `Annotation` data is usually given by a feature structure or attribute-value pairs.

The value range of an attribute can characterize the annotation type because it varies depending on the linguistic processing functionality type. Typically, the range would be a set of symbols in a system; for example, the RFC3066 language tag set or Penn Treebank POS tag set. However, other value types have to be considered. For example, the Lemmatization functionality annotates a word token with the lemma form string, which would have been generated by the processor. Therefore, the value range is not limited by an explicitly specified set. Another example is the Dependency Parsing functionality; it annotates each word in the sentence with its syntactic head, possibly along with the dependency relation label. That is, the value of, say, the `syntactic head` attribute ranges over possible word indices in the input sentence. In summary, the possible types of value range are symbols in a system, generated strings, and references to other linguistic data elements in the input linguistic data.

## 7. Concluding Remarks

This paper classified a standard set of linguistic processing functionalities through a careful survey of a Web-based language service infrastructure and several NLP toolkits. This paper also developed an ontological depiction that classifies the linguistic processing functionalities with respect to the fundamental data operation types.

The presented view may deviate somewhat from the mainstream view of the field, which holds that every linguistic processing function must be assigned to a type of linguistic annotation. The most prominent standard embodying this view is LAF/GrAF (Ide and Romary, 2004; Ide and Suderman, 2007), in which only two primary data types reside: primary data and annotation. We understand that this framework is highly useful in achieving so-called linguistic data interoperability. Nevertheless, we think that the proposed ontological classification of linguistic processing functionalities can play a role in their formal descriptions because it explicitly dictates the functional aspects of a linguistic processor.

For future work, we plan to extend and refine the presented abstract types as new kinds of linguistic processors are revealed. In addition, we would explore the possibility of augmenting the language service ontology (Hayashi, 2011) by adopting the presented abstract types. One of the problems that we were aware of in developing the language service ontology was that it is difficult to formally represent the relationships or constraints that should hold between the input and the output of a linguistic processor with the conventional RDF/OWL foundations. In this respect, we will consider adopting Z-notation (Spivey, 2001), which was developed for formal specification of computer systems, to formally describe the functional specification.

[7] This figure was created by using Protégé-OWL ontology editor with Ontoviz plugin.

## 8. Acknowledgements

The presented work was largely supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE) of the Ministry of Internal Affairs and Communications of Japan.

## 9. References

Dieter Fensel, Federico Michele Facca, Elena Simperl, and Ioan Toma. 2011. *Semantic Web Services*. Springer.

Yoshihiko Hayashi. 2011. Prospects for an Ontology-Grounded Language Service Infrastructure. *Proc. of IJCNLP 2011 Workshop on Language Resources, Technology and Services in the Sharing Paradigm*, pp.1–7.

Nancy Ide, and Laurent Romary. 2004. International Standard for a Linguistic Annotation Framework. *Journal of Natural Language Engineering*, Vol.10, No.3–4, pp.211–225.

Nancy Ide, and Keith Suderman. 2007. GrAF: A Graph-based Format for Linguistic Annotations. *Proc. of ACL 2007 Linguistic Annotation Workshop*, pp.1–8.

Toru Ishida (Ed.). 2011. *The Language Grid, Service-Oriented Collective Intelligence for Language Resource Interoperability*, Springer.

Mike Spivey. 2001. The Z Notation: a reference manual. `http://spivey.oriel.ox.ac.uk/mike/zrm/`