



Linguistic Annotation Framework

Nancy Ide

2006-08

This document is partial but outlines the major aspects of LAF in order to provide a basis for discussion.

1. Introduction

As noted in Cole, *et al.*, 1997, years of research and development in computational linguistics and language engineering have yielded many stable results, which have in turn been integrated into language processing applications and industrial software. Especially over the past fifteen years, researchers and developers have increasingly understood the need to define common practices and formats for linguistic resources, which serve HLT development as the primary source for statistical language modeling. To answer this need, numerous projects have been launched to lay the basis for standardization of resource representation and annotation--e.g., the Text encoding Initiative (TEI)¹, the Corpus Encoding Standard (CES and XCES)², the Expert Advisory Group on Language Engineering Standards (EAGLES) and the International Standard for Language Engineering (ISLE)³—as well as software platforms for resource creation, annotation, and use--MULTEXT⁴, LT XML⁵, GATE⁶, NITE⁷, ATLAS⁸). However, although in practice consensus has begun to emerge, definitive standards have not yet been put in place. In large part this is as it should be: advances in technology together with the emergence of a solid body of web-based standards have dramatically impacted and re-defined many of our ideas about the ways in which resources will be stored and accessed over the past several years. Perhaps more importantly, the ways in which language data— together with “communicative” data of any kind, including gesture, facial expression, speech characteristics—are processed and analyzed will certainly continue to change, as more and more emphasis is put on immediate processing of (often multi-modal) streamed

¹ <http://www.tei-c.org>

² <http://www.xml-ces.org>

³ <http://www.ilc.cnr.it/EAGLES96/home.html>

⁴ <http://www.lpl.univ-aix.fr/projects/multext>

⁵ <http://www.ltg.ed.ac.uk/software/xml>

⁶ <http://gate.ac.uk/>

⁷ <http://www.dfki.de/nite/main.html>

⁸ <http://www.nist.gov/speech/atlas/>

data. Whatever the scenario, if we intend to make HLT work in the larger arena of universal availability and accessibility, data, its annotations, and processing results will have to be represented in some way that allows exploitation by the full array of language processing technologies.

The growth of the web and the explosion in the number of electronic documents to be handled and maintained within the industrial sector has created an immediate and urgent need for generic language processing components for document indexing and classifying, information extraction, summarization, topic detection, etc., in both mono- and multi-lingual environments, together with robust machine translation and facilities for man-machine multimodal communication. While progress will continue, the field has nonetheless reached a point where we can see clear to a reasonable representation and processing model that should fulfill the needs of HLT for at least the foreseeable future. Indeed, commonality that can enable flexible use and reuse of communicative data is essential for the next generation of language processing applications, if we are to build a global information environment.

The Linguistic Annotation Framework (LAF) builds on existing technologies and schemes to codify best practices as a set of standards for representing and processing language-related information, in order to leverage the growth of language engineering. LAF is intended to achieve the following general goals:

- Provide means to use and reuse linguistic data across applications, at all levels of linguistic description from surface mark-up of primary sources to multi-layered processing results;
- Facilitate maintenance of a coherent document life cycle through various processing stages, so as to enable enrichment of existing data with new information and the incremental construction of processing systems.

2. Background and Scope

Over the past 20 years, numerous projects and initiatives have worked toward the development of standards for one or more of the components pictured above, as well as for a general architecture that would enable efficient representation of the resources themselves together with the "links" establishing the inter-dependencies among them. Among the most notable are the TEI, CES and XCES, and MATE/NITE for the representation of primary data and annotations; EAGLES/ISLE for annotation content; OLIF⁹, SALT¹⁰, and ISLE for various kinds of lexical/terminological data; RDF/OWL and Topic Maps for knowledge structures; Dublin Core and the Open Archives Initiative (OAI)¹¹ for general metadata; MPEG7, IMDI, and OLAC for domain-specific metadata; Corba¹² and the W3C's SOAP¹³ and web services work for access protocols; and

⁹ <http://www.olif.net/>

¹⁰ <http://www.loria.fr/projets/SALT/>

¹¹ <http://www.openarchives.org/>

¹² <http://www.corba.org/>

¹³ <http://www.w3.org/TR/soap/>

MULTEXT, Edinburgh's LT framework, TIPSTER¹⁴, GATE, and ATLAS for general architecture. Most of these projects actually address several of what we can regard as the multiple "dimensions" of language resource representation, including (at least) the following:

Rendering formats and mechanisms, such as SGML, XML, Lisp-like structures, annotation graphs, or a particular database format.

Annotation content, including categories of annotation information for linguistic phenomena (e.g., modality, aspect, etc.) and the values that can be associated with each category.

General architectural principles for language resources, such as the now widely-accepted notions of pipeline architecture and stand-off annotation.

These dimensions are inter-dependent: for example, the choice of a representation format will have repercussions for content, first of all because relations among pieces of information may be expressed implicitly through the structures provided by the format, the most common of which is a hierarchical structure for grouping and/or defining part/whole relations. Some formats impose other constraints—for example, Lisp-like formats provide a hierarchical structure but do not readily accommodate labeling the structures to distinguish their functions (e.g., grouping, listing alternatives, etc.), as one might do in XML by simply giving a tag a meaningful name. Similarly, implementing stand-off annotation with XML dictates use of XML paths, pointers, and links. As a result, format and content have in past projects often been treated as a whole, rather than addressing them separately.

Annotation of linguistic data may involve multiple annotation steps, for example, morpho-syntactic tagging, syntactic analysis, entity and event recognition, semantic annotation, co-reference resolution, discourse structure analysis, etc. Annotation at lower linguistic levels typically serves as input to the higher-level annotation process in an incremental process. Depending on the application intended to use the annotations, lower-level annotations may or may not be preserved in a persistent format. For example, information extraction software often annotates linguistic features required to generate the final annotation, without preserving the intermediate information. In other situations, the annotation process may not be strictly incremental; for example, when handling streamed data (text, video, and audio, a stream of sensor readings, satellite images, etc.) the processor analyzes language data in a linear, time-bound sequence, and therefore annotations may be temporarily partial during processing if long-distance dependencies between seen and unseen segments of the data exist.

At present, most annotated resources are static entities used primarily for training annotation software, as well as corpus linguistics and lexicography. However, in the context of the Semantic Web, annotations for a variety of higher-level linguistic and communicative features will increasingly be preserved in web-accessible form and used by software agents and other analytic software for inferencing and retrieval. This dictates that the LAF not only relies on web technologies (e.g., RDF, OWL) for representing annotations, but also that "layers" of annotations for the full range of annotation types

¹⁴ <http://www.fas.org/irp/program/process/tipster.htm>

(including named entities, time, space, and event annotation, annotation for gesture, facial expression, etc.) are at the same time separable (so that agents and other analytic software can access only those annotation types that are required for the purpose, and mergeable (so that two or more annotation types can be combined where necessary). They may also need to be dynamic, in the sense that new and/or modified information can be added as necessary.

In order to ensure that the LAF architecture reflects state-of-the-art methods drawn from consensus of the research community, a group of experts¹⁵ was convened in November, 2002, to lay out its overall structure. The group, which included researchers with extensive experience in the development of annotation schemes at a variety of linguistic levels together with developers of major resource-handling software (GATE, ATLAS, Edinburgh LT tools), defined the following general principles to guide the LAF development:

- The data model and document form are distinct but mappable to one another
- The data model is parsimonious, general, and formally precise.
- The document form is largely under user control.
- The mapping between the flexible document form and data model is via a rigid dump-format. The responsibility of converting to the dump format is on the producer of the resource.
- Mapping is operationalized via either a schema-based data-binding process or schema-derived stylesheet mapping between the user document and the dump format instantiation. The mapping from document form to the dump format is documented in an XML Schema (or the functional equivalent thereof) associated with the dump format instantiation.
- It must be possible to isolate specific layers of annotation from other annotation layers or the primary (base) data; i.e., it must be possible to create a dump format instantiation using stand-off annotation
- The dump format supports stream marshalling and unmarshalling

¹⁵ Participants: Nuria Bel (Universitat de Barcelona), David Durand (Brown University), Henry Thompson (University of Edinburgh), Koiti Hasida (AIST Tokyo), Eric De La Clergerie (INRIA), Lionel Clement (INRIA), Laurent Romary (LORIA), Nancy Ide (Vassar College), Kiyong Lee (Korea University), Keith Suderman (Vassar College), Aswani Kumar (LORIA), Chris Laprun (NIST), Thierry Declerck (DFKI), Jean Carletta (University of Edinburgh), Michael Strube (European Media Laboratory), Hamish Cunningham (University of Sheffield), Tomaz Erjavec (Institute Jozef Stefan), Hennie Brugman (Max-Planck-Institut für Psycholinguistik), Fabio Vitali (Universite di Bologna), Key-Sun Choi (Korterm), Jean-Michel Borde (Digital Visual), Eric Kow (LORIA).

3. Terms and Definitions

Annotation

In this document the term *annotation* refers to the process of adding linguistic information to language data (“annotation of a corpus”) or the linguistic information itself (“an annotation”), independent of its representation. For example, one may annotate a document for syntax using a LISP-like representation, an XML representation, etc.

Representation

The term *representation* refers to the format in which the annotation is rendered, e.g. XML, LISP, etc. independent of its content. For example, a phrase structure syntactic annotation and a dependency-based annotation may both be represented using XML, even though the annotation information itself is very different.

Types of Annotation

We distinguish two fundamental types of annotation activity:

1. *segmentation* : delimits linguistic elements that appear in the primary data, including
 - a. continuous segments (appear contiguously in the primary data)
 - b. super- and sub-segments, where groups of segments will comprise the parts of a larger segment (e.g., a contiguous word segments typically comprise a sentence segment)
 - c. discontinuous segments (linking continuous segments)
 - d. landmarks (e.g time stamps) that note a point in the primary data

In current practice, segmental information may or may not appear in the document containing the primary data itself. Documents considered to be *read-only*, for example, might be segmented by specifying byte offsets into the primary document where a given segment begins and ends.

2. *linguistic annotation* : provides linguistic information about the segments in the primary data, e.g., a morpho-syntactic annotation in which a part of speech and lemma are associated with each segment in the data. Note that the identification of a segment as a word, sentence, noun phrase, etc. also constitutes linguistic annotation. In current practice, when it is possible to do so, segmentation and identification of the linguistic role or properties of that segment are often combined (e.g., syntactic bracketing, or delimiting each word in the document with an XML tag that identifies the segment as a word, sentence, etc.).

Stand-off annotation

Annotations layered over a given primary document and instantiated in a document separate from that containing the primary data. Stand-off annotations refer to specific locations in the primary data, by addressing byte offsets, elements, etc. to which the annotation applies; or they refer to other annotations. Multiple stand-off annotation

documents for a given type of annotation can refer to the same primary document (e.g., two different part of speech annotations for a given text). There is no requirement that a single XML-compliant document may be created by merging stand-off annotation documents with the primary data; that is, two annotation documents may specify trees over the primary data that contain overlapping hierarchies.

4. Design Principles

The Linguistic Annotation Framework is designed based on the following requirements:

- LAF must enable users to represent their data and annotations in a variety of formats of their own choosing.
- LAF must accommodate all varieties of annotation and data (including, e.g., time-stamped speech, streamed data, etc.).
- LAF must be easy to use so that the community will adopt it.

To meet these requirements, we identified an abstract data model for annotations, such that annotation formats conforming to the model—regardless of differences in their superficial representation—are trivially mappable to one another. The model is represented in a “dump” format that is intended to function in the same way as an interlingua functions for machine translation--i.e., as a representation of universal concepts into and out of which realizations in different languages are mapped for the purposes of translation. Here, the different languages are different user-defined formats.

The overall LAF design is based on a few straightforward principles:

- **Separation of data and annotations.** Language data is regarded as “read-only” and contains no annotations. All annotations are contained in stand-off documents linked to the primary data. Note that we define primary data as the source data obtained by the user, which may include markup (e.g. HTML tags, time stamps, etc.), but to which no linguistic annotations that will be rendered into the dump format have been superimposed.¹⁶ Treating the primary data as read-only avoids maintenance problems associated with stand-off approaches, where modification of the data may cause links into it to break.
- **Separation of user annotation formats and the exchange (“dump”) format.** Users may use any format for annotations, including not only XML but also formats such as LISP-like structures or tab-delimited information, etc. The only requirement is that the information in the user’s annotation format is automatically mappable to the feature structure data model instantiated by the dump format.
- **Separation of structure and content in the dump format.** In many annotation schemes, content and structure are not clearly differentiated. In such cases, structural relations among parts of the annotation content may be ambiguous. The most obvious

¹⁶ We recognize that the line between source data and annotations is a highly debatable matter; our definition here is driven by the practical concerns of dealing with web and legacy data. We believe that in most cases, there is a common sense distinction between data and annotations that can be applied.

example is LISP-like formats, which use parentheses to group information, with no indication of whether the group represents an inclusive list, a prioritized list, a set of alternatives, etc. The only way to determine which applies is to examine the data; if, for instance, the list describes syntactic frames or part of speech for a given lexical item, it is probably a set of alternatives, but human knowledge is required to decide this and program a script to treat it appropriately. LAF requires that all annotation information included in the original format be made explicit in the dump format representation; this way, fully automatic transduction from the dump format representation or other user formats is ensured.

The dump format is isomorphic to the underlying abstract data model, which is built upon a clear separation of the *referential structure* of the annotations and the *annotation content*, that is, the linguistic information itself.

The referential structure is used to associate annotations with regions in primary data or with other annotations. The referential structure forms a graph consisting of elementary structural nodes to which one or more feature structures are attached, providing the semantics (content) of the annotation. Feature structures are represented as specified in ISO TC37 SC4 document N188¹⁷. These specifications implement the full power of feature structures and define inheritance, unification, and subsumption mechanisms over the structures, thus enabling the representation of linguistic information at any level of complexity. The specifications also provide a concise format for representing simple feature-value pairs, which suffices to represent many annotations, together with means to define feature libraries. Because the FS encoding is fully specified in N188, we focus here on the specification for the referential structure; note, however, that the specifications here supercede those given in N188 section 5.10 (“Linking Text and Analysis”) in the dump format representation.

Primary data contains no annotations and is regarded as “read-only”. LAF insists on the existence of a *primary segmentation* of the primary data that identifies contiguous sequences of characters comprising indivisible logical units, where a *character* is defined as a contiguous byte sequence of a user-defined length.¹⁸ A primary segmentation consists of a set of disjoint edges over primary data. The vertices are virtual, located between each character in the data, and sequentially numbered. Each edge (x,y) in the graph delimits a non-divisible region of primary data. Multiple edge sets may be defined over primary data, specifying segmentations at different levels of granularity. Figure 1 shows a primary segmentation with edges $(0,3)$, $(4,9)$, and $(10,16)$.

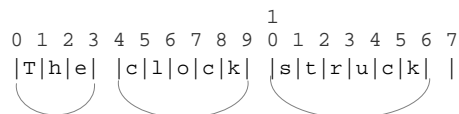


Figure 1. Primary segmentation

¹⁷ See ISO TC37 SC4 document N188, Feature Structures-Part 1: Feature Structure Representation (2005-10-01), available at <http://www.tc37sc4.org/>

¹⁸ As specified in ISO 10646/Unicode.

For text, the most common primary segmentation is the token, over which, for example, word forms (which may or may not consist of contiguous tokens) may be defined for the purposes of morpho-syntactic annotation. However, edges in a primary segmentation can be defined over any span of contiguous primary data, regardless of its length.

To enable reference to a primary segmentation, we define an *edge graph* as follows: given an edge set, E , over primary data, we create an edge graph E' such that for each edge (x,y) in E , there is a vertex xy in E' . Annotations are associated with regions of primary data by referencing the edge graph vertices; annotations never reference the primary data directly. Edges in E' are defined when annotations reference a vertex in E' , which may or may not be contiguous.¹⁹ When zero²⁰ or more vertices are referenced by a single annotation, the effect is to create a new edge in E' covering the referenced vertices. By default, the effect is to concatenate the information covered by the referenced vertices; this can be overridden to specify that the vertices are to be regarded as an ordered list or a collection (“bag”). [do we need set?]

Edges defined in an annotation may be referenced from other annotations, by treating the edge as a vertex and associating a feature structure with it. The strategy described above is applied recursively, thus creating an n -order edge graph involving the referenced subset of edges in E' . In this way, annotations may be layered indefinitely, resulting in a well-formed graph with clearly defined properties, to which common graph traversal and analysis algorithms may be applied.

As many annotations as desired, of either the same type or different types, can reference the same segmentation or be layered over lower-level annotations. For example, additional morpho-syntactic annotations could reference the segmentation above; a co-reference annotation could reference the NP annotation; and a syntactic annotation could reference the same NP annotation. Figure 2 shows how several segmentation and annotation documents can be layered and inter-leaved over primary data.

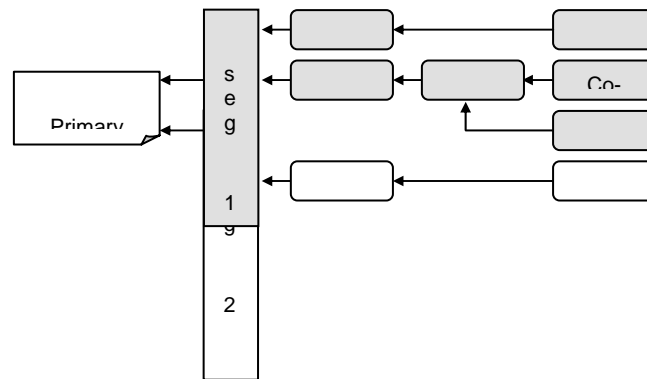


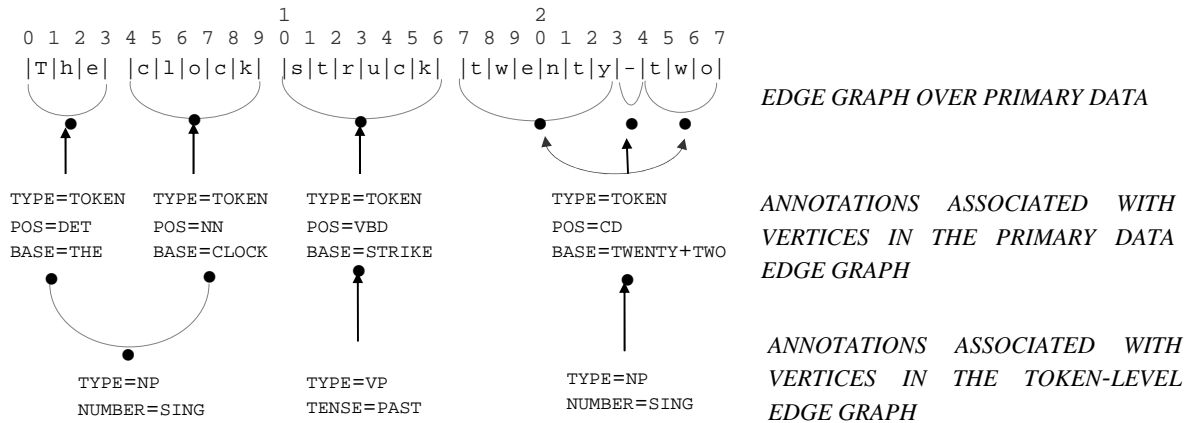
Figure 2. Dump format scenario with two segmentations and multiple annotations

¹⁹ Note that this deviates from the typical edge graph model, in which edges are assumed to exist between any two vertices (x,y) and (u,v) in E' , if and only if $y = u$. In our model, edges in E' must be defined explicitly and need not have a common element.

²⁰ A zero reference is used in the special case where the annotation applies to information that is not present in the data.

Example

The following graphical representation shows an edge graph over primary data and annotations for Token/POS and noun and verb phrases. New edges are defined for the compound “twenty-two”, which this segmentation identifies as three separate edges, and for the noun phrase “the clock”. The edge created for the token “twenty-two” is treated as a vertex in the NP annotation. The VP annotation references the vertex created by the token annotation for “struck”.



To ground this example, we provide a sample XML encoding:

```

<!-- edges over primary data -->
<edge id="e1" from="0" to="3"/>
<edge id="e2" from="4" to="9"/>
<edge id="e3" from="10" to="16"/>
<edge id="e4" from="17" to="23"/>
<edge id="e5" from="23" to="24"/>
<edge id="e6" from="14" to="27"/>

```

The *from* and *to* attributes specify the start and end nodes for the edge; the actual byte offset within the data is a function of the character encoding. Each edge can now be treated as a vertex by an annotation.

The following shows a fragment of an annotation for morpho-syntax that provides information associated with a vertex in the edge graph defined over the primary segmentation:

```

<edge id="t2" refs="e2">
  <fs type="token">
    <f name="base" sVal="clock"/>
    <f name="pos" sVal="NN"/>
  </fs>
</edge>

```

The `<edge>` element refers to the pre-defined edge (treated as a vertex) with id *e2* in the primary segmentation. The feature structure included within the element provides the annotation, in this case, two simple feature/value pairs specifying lemma and part of speech. This newly-defined edge may now be treated as a vertex by other annotations.

To create a new edge from two or more vertices, the relevant id's are provided in list as the value of the *targets* attribute on the relevant `<edge>` element. The following example shows an annotation for the noun phrase “the clock”:

```
<!-- edge graph for "The" and "clock" -->
<edge id="np1" refs="t1 t2">
  <fs type="NP">
    <f name="number" sVal="singular"/>
  </fs>
</edge>
```

The effect of this notation is to concatenate the edges referenced in the *refs* attribute and create a new edge from the start node of the first to its end node, through the start node of the second to its end node, etc.

5. Data Categories

Semantic coherence in annotations is provided in part by a registry of *data categories* maintained in RDF/OWL format. Data categories include both feature (attribute) names and the values associated with them. The Data Category Registry (DCR) is fully described in document ???.

Dump format instantiations of annotations specify data categories in one of the following ways:

1. direct reference to a data category defined in the DCR
2. reference to features or values in a pre-defined feature structure library
3. use of a user-specific category name

In cases (2) and (3), one of the following must be provided:

- a schema providing the mapping of categories used in the annotation to categories in the DCR, or
- a formal specification of newly-defined categories in DCR entry format

6. Mapping to the Abstract Format

[This section will provide an extensive set of examples of mappings from some common formats to the dump format]

Note: there should be two defined schemes for feature structure specification: one which uses the most common and simple mechanisms and the full specification given in N188, in which case an annotation will include an indication of which set it uses: “FS Full” or “FS Light”. Alternatively, the parts of the FS specification that are used in a particular annotation should be documented with the dump format instantiation. The reason for this is to make it easy for those writing scripts to map into/out of the dump format to know what they have to deal with, and what may or may not be representable in an in-house scheme.

7. APPENDIX. XML specification of the dump format