Managing metadata variability within a hierarchy of annotation schemas

Ionuţ Cristian Pistol¹, Dan Cristea¹,²

¹ Faculty of Computer Science, University “Al. I. Cuza” of Iaşi, Romania
{ipistol, dcristea}info.uaic.ro
² Institute for Computer Science, Romanian Academy, Iaşi, Romania

Abstract. The paper describes the theoretical basis of the ALPE¹ model, a hierarchy of annotation formats used to guide the automatic computation of processing flows capable of performing complex linguistic processing tasks. The hierarchy is comprised of a core, which is a direct acyclic graph whose nodes represent XML annotation formats, and a halo which contains additional annotation formats. The core hierarchy also serves as a standardization hub for annotated documents. The focus of the paper is the description of the new additions to the model, allowing the integration and usage of non-XML formats in processing flows and new equivalence relations between XML formats. This type of hierarchy is implemented in the ALPE system, which is also briefly described in the paper.

1 Introduction

In the latter years, the field of Natural Language Processing witnessed the emergence of a significant effort concerning the standardization and usability aspects of developed processing tools and resources. Projects such as CLARIN² and FLaReNet³, among others, intend to offer both developers and users of language resources and tools a management solution for the growing set of resources available. The primary objectives of these projects are to provide reusability in new contexts for existing resources and to guarantee maximum visibility and reusability for newly developed resources. An easy widening of the original setting of usage means a multiplication of the visibility of a tool and, finally, of the productivity of the research activity. In terms of managing linguistic processing tools, previous efforts lead to the development of linguistic processing meta-systems, most significant ones being GATE⁴ [3] and UIMA⁵ [4]. Both systems allow access to a set of independently developed NLP tools, integrated into an environment offering means to create and use processing chains adding linguistic metadata to an input corpus. They allow integration of new tools and new processing chains, but these functionalities require programming experience.

¹ Automated Linguistic Processing Environment
² http://www.clarin.eu/
³ http://www.flarenet.eu/
⁴ http://www.gate.ac.uk/
⁵ www.research.ibm.com/UIMA/
ALPE [1,2] is another such system, intended to define a framework which facilitates the integration of LT tools of different origins. In ALPE, linguistic resources and tools are placed with respect to a directed acyclic graph whose nodes are annotation schemas and the hierarchical links are subsumption relations between schemas. Processing tools are attached to the edges in the graph and ALPE uses these tools to compute processing chains, once specific pairs of start-destination formats are given. ALPE offers several advantages over existent systems with a similar goal, as it is able to identify the annotated format of the input file, then to automatically compute the processing steps required to bring an input file to the required output format, and, eventually to run this chain if costs/IPR conditions are fulfilled.

Section two of this paper briefly describes the base ALPE hierarchy and section three describes the enhancement of the base hierarchy with processing power and “clouds” of equivalent formats. The conclusions, as well as the further planned developments are described in section four.

2 The base hierarchy

The following section briefly describes the base of the ALPE hierarchy model, which is the extended hierarchy of annotation schemas. For a more detailed and commented description, please consult [1,2].

2.1 The core hierarchy

The basis of the core hierarchy is a directed acyclic graph (DAG) which configures the metadata of linguistic annotation in a hierarchy of XML schemas. Nodes in this graph are called core nodes. Each core node corresponds to a single XML annotation format. In the following, we will consider core nodes as equivalent to a XML annotation schema and we will occasionally refer to them as such.

We note as $T(A)$, where $A$ is a core node, the set of elements (tags) defined in the XML annotation format corresponding to the core node $A$. We note as $t(A)$, where $A$ is a core node and $t \in T(A)$ is the set of attributes belonging to the element $t$ as it appears in the core node $A$.

Edges connecting core nodes are called core edges. If there is a core edge linking a core node $A$ with a core node $B$ (we will say also that $A$ is formally subsuming $B$, noted as $A \sqsubseteq B$) then the following conditions hold simultaneously:

- any element (tag-name) of $A$ is also in $B$: $T(A) \subseteq T(B)$;
- any attribute in the list of attributes of a tag-name in $A$ is also in the list of attributes of the same tag-name of $B$: $t(A) \subseteq t(B)$ for all $t \in T(A)$. 
For the hierarchy in Figure 1 it holds that node A subsumes nodes B, C and D, and node B subsumes node D. As such, a hierarchical relation between a core node A and one descendent B describes B as an annotation schema which is more informative than A. In general, either B has at least one tag-name which is not in A, and/or there is at least one tag-name in B such that at least one attribute in its list of attributes is not in the list of attributes of the homonymous tag-name in A. The direction of the core edge connecting nodes A and B is given by the subsuming relation, with the subsuming node being the origin of the core edge and the subsumed node the destination.

Figure 1: An ALPE core hierarchy

Theorem: The subsumption relation s is a partial order relation on the set of core nodes.

2.2 The haloed hierarchy

In addition to the core nodes and edges, which strictly observe the specified restrictions, we can include in an extension to the core hierarchy, called a halo, other types of nodes and edges such as:

- Nodes representing other annotation formats than XML. We can consider each node in the core hierarchy representing not just a specific format, but rather a class of annotation formats, whose representative is an XML format. This is the case when a variety of annotation formats, used in practice, are all intended to code the same information, but in different ways. These formats can be represented in the hierarchy as halo nodes;

- Edges originating or ending in halo nodes. These edges can either originate or end in the core hierarchy, or they can be completely outside the core hierarchy. The semantic value of these edges is to mark the semantic subsumption between the source and destination nodes, relation considered at an abstract level as opposed to the formalized subsumption relation. Semantic subsumption means that the information encoded in the origin node’s format is part of the information encoded in the destination node’s format, but this inclusion cannot be strictly formalized using XML elements and attributes.
The core hierarchy and the halo form a full hierarchy. The halo nodes and edges expand the core hierarchy outside the limits of XML, allowing other types of annotation formats to be represented in the full hierarchy. In Figure 2 is shown an example of a full hierarchy based on the core hierarchy in Figure 1. With continuous lines are marked core nodes and edges (and the core hierarchy), and with interrupted lines halo nodes and edges. Nodes A, B, C and D are core nodes and nodes E, F, G and H are halo nodes.

As for the core edges, the following hold true:

Axiom: There exists at most one edge between two halo nodes.

Axiom: There exists at most one edge between any two nodes in the full hierarchy.

These axioms say that between any two nodes in the full hierarchy there is at most one subsumption relation, either formal or semantic.

In all core hierarchies we introduce an obligatory root core node. This node corresponds to the basic XML format, with only a root element. The definition of this node leads to the following theorem:

Theorem: The root core node of a hierarchy subsumes all nodes in the hierarchy.

Theorem: The core hierarchy is a connected graph (disregarding the edges orientation).

In order to guarantee the connectivity of the full hierarchy, we introduce the following axiom:

Axiom: From each halo node there is at least one core node which can be reached (disregarding the orientation of the edges).

This axiom basically says that no halo nodes are disconnected from the core hierarchy: in order for an annotation format to be included as a node in the hierarchy it has to have at least one other node already in the hierarchy which it subsumes (or is subsumed by) - either formal subsumption (introduced in 2.1) or semantic subsumption. The previous theorem and this axiom lead to the next theorem:
Theorem: The full hierarchy is a connected graph (disregarding orientation of the edges).

The proof is direct: the core nodes are connected (as proven by the previous theorem) and each halo node is connected to at least one core node. This means that from each halo node all core codes can be reached. Since all halo nodes can be reached from at least one core node, as a corollary to the previous axiom, this means that from each core node all other nodes can be reached. Also, the axiom and the following conclusion lead to the fact that from each halo node all other nodes can be reached. Thus, the full hierarchy is a connected graph.

3 The hierarchy augmented with processing power

3.1 Adding processing power to the hierarchy

If there is an edge (either core or halo) between nodes A and B, there should be a process which takes as input a file observing the restrictions imposed by the node A and produces as output a file observing the restrictions imposed by the node B. While doing this type of processing the module might make use also of some additional resources outside the hierarchy, such as language models and lexicons. A graph of annotation schemas on which processing modules have been marked on edges is called augmented with processing power (or simply, augmented).

An edge to which there is at least one processing module attached will be called a processing edge. A single edge in the graph can have multiple processing modules attached to it, if those modules observe the same restrictions regarding their input and output formats. If there is no known processing module attached to an edge, that edge is called a carrier edge. For reasons due to the way processing paths are computed in the hierarchy, all edges in the hierarchy can be considered carrier edges and can be included as such in the resulting processing flows.

A node to which points a single carrier edge (and no other edges) is called an isolated node.

3.2 The processing edges

Each node in the hierarchy has a unique identification key (ID). This is introduced to ensure a clear reference to each node.
The edges, being them of either type, processing or carrier, are noted as E(A,B), where A is the ID of the input (origin) node, and B is the ID of the output (destination) node. This notation is unique for each edge in the hierarchy due to the uniqueness of the node IDs and the restriction that between any two nodes there can exist only one edge.

![Diagram of a processing edge and attached processing modules]

**Figure 3:** A processing edge and the attached processing modules

The processing modules are noted as p_i(A,B) where i is the unique index of the processing module, A is the origin node of the edge to which the module is attached and B is the destination node of the edge to which the module is attached.

Processing modules will be used to introduce a series of operations on the augmented hierarchy. A processing module will be regarded as part of a more complex operation performed on an input file in order to produce an output file. These complex operations are called flows in ALPE and will be detailed in section 3.4. The operators for these operations (as well as for the processing modules when taken as a function in an expression) will be either nodes in the hierarchy (representing files observing the schemas corresponding to those nodes) or other processing modules applied on an input node (resulting another node).
3.3 Conversion edges and synclouds

An edge connected a halo node with another node of the full hierarchy is called a conversion edge. All conversion edges have at least one attached processing module. That module is basically a wrapper capable of converting a format into another. Part of the encoded information in the source node is rewritten in another format in the destination node. No new information is added.

If both source and destination nodes are core nodes this means that at least one XML element or attribute is written in another way (change of element name, converting an attribute into an element, merging attributes or elements, etc.). If one of the connecting nodes is in the halo, the module attached to the conversion edge rewrites either XML into a non-XML format or the other way around, depending on the direction of the edge. Conversion edges are created when corresponding processing modules are integrated in the hierarchy.

All nodes that can be reached from the same node using only conversion edges (disregarding the direction of the edges) form a syncloud (synonymy clouds). All nodes in the same syncloud contain the same information, but encoded differently. As shown in figure 4, part of the syncloud can be found in the halo, and part in one node belonging to the core hierarchy.

Figure 4: A syncloud
3.4 Operations on the hierarchy

As shown somewhere else [1,2], two types of operation complement the model:

- Hierarchy building operations: initialization (initializes a new hierarchy), classification (finds the place of an input XML schema onto an existing hierarchy) and integration (adds a new processing module onto the hierarchy).
- Processing operations: pipelining (combines two individual modules on the same sub document, the second in sequence using the output of the first one), simplification (simplifies an annotated file to the format of another subsuming format in the hierarchy) and merging (merges two or more annotations applied to the same hub document onto one).

On the augmented hierarchy are defined flows, as processing operations of the above types performed over an input file corresponding to the annotation described in a node in the hierarchy (or a node, for simplification).

A processing flow can be defined as any combination of pipeline, simplify, merge operations and void flows, applied to nodes. The processing flows can be computed automatically for an input file and a specified output format. The process of automatic computation of flows is partially described in [1].

3.5 Flows and synclouds

The main benefit of the introduction of synclouds in the ALPE hierarchy model is the accommodation of various available processing modules using non-XML documents as either input or output.

![Figure 5: Flows in synclouds](image)
A simple computed flow, on the left of figure 5, shows only the general picture of an ALPE hierarchy, where a node identifies a distinct annotation format. Figure 5a depicts the case of the processing modules considered on the left being actual modules between the core nodes of the synclouds. Figure 5b shows that a flow can include processing edges between XML and non-XML formats in a syncloud. These modules can be pipelined with XML processing modules using wrappers attached to edges between the core nodes and other nodes in the syncloud.

Also, as in 5c, flows can actually exist outside core nodes and include only processing modules using non-XML intermediate formats, allowing the straightforward integration in ALPE of flows produced by other systems, such as GATE or UIMA, with the only change being the addition of the input/output wrappers.

4 Conclusions and further work

ALPE is thought to offer compatibility with other similar systems as GATE and UIMA. The proposed model allows seamless integration of annotation formats, processing modules and processing flows, requiring at most the existence of wrappers, if the involved annotation formats are not encoded as XML core codes. Since UIMA is the prominent system comparable to ALPE and since both GATE and UIMA are now open-source, we study the possibility of integrating ALPE in either system. This will be done after ALPE is fully implemented (now it’s in the final stages of implementation). After a proof-of-concept can be made for ALPE, we plan to attract other researchers interested in developing it further, either as a stand-alone system or integrated in broader scope systems, such as UIMA.

Adopting ALPE as a management and access environment for the resources employed and developed in a computational linguist project proposing the development of multilingual resources and tools, such as CLARIN [6], has the potential of benefiting both the project and the interested user. One important further development of ALPE will be a web-service allowing users to build, configure and use ALPE hierarchies on the web, either as a limited password-protected resource or a global linguistic resources collection. This type of hierarchy is able to manage multilingual resources as well as resources which require a fee to be paid before usage. Each user will be able to contribute its own tools and annotated resources, as well as using processing chains adapted to its specifications, both in terms of input and output formats and cost and performance issues.

Standards usually appear late. In order for an annotation convention to become a standard it should be adopted by a community of people. This happens over a long interval of time, at the moment when a theory has passed the delicate exam of empirical validation. Therefore, there is a strong need for accepting new formats, which should work together with well accepted ones. We need a mechanism able to “understand” the notations, to detect the semantics beyond the notations, to infer the meaning of notations and to establish semantic links between new formats before standards appear. The precision of this automatic discovery could not be extremely high since the inference techniques should include also heuristics, a little bit of AI. When a new mechanism is invented and there is no standard in the hierarchy to describe it, these

6 http://www.mpi.nl/clarin/
heuristics aim at finding the right place in the hierarchy for the new schema. The first step would be the clear definition of the semantics of a standard. A promising new model of describing annotation semantics, the Linguistic Annotation Format [5], has the potential of clearly defining semantic links between various annotation formats. We are currently in the process of integrating a version of this model as a way to formally describe nodes in the ALPE core hierarchy and as a possible base for an automatic detection of semantic links between formats.

This paper touches the above aspect as it proposes a model of integrating XML schemas with non-standardized formats. Non-XML formats take the form of syn-clouds as clusters of formats around recognized XML standards. Processing flows can be computed automatically as combination of processing modules and wrappers.

References