Managing Language Resources and Tools using a Hierarchy of Annotation Schemas

Dan Cristea
Faculty of Computer Science, University "Al. I. Cuza" of Iaşi, Romania
Institute for Computer Science, Romanian Academy, Iaşi, Romania
dcristea@info.uaic.ro

Ionut Cristian Pistol
Faculty of Computer Science, University "Al. I. Cuza" of Iaşi, Romania
ipistol@info.uaic.ro

Abstract

This paper describes the concept and usage of ALPE (Automated Linguistic Processing Environment) a system designed to facilitate the management and deployment of large and dynamic collections of linguistic resources and tools. ALPE can build linguistic processing chains involving the annotation formats and the tools integrated into a hierarchical structure. The particularities and advantages of integrating ALPE in a project involving the development and usage of multiple linguistic resources are the main topics of this paper.

1. Introduction

Making sure that corpora, resources and tools are reusable in different contexts than that of the originating project is one of the recent main topics of interest in the Natural Language Processing community. Re-using a resource initially developed for a specific project usually fails for one of two reasons: either the resource is not enough documented (the format is not known to the re-user), or the resource is not directly accessible (the location of the resource is not known to the re-user). Making sure a project's results are well organized and accessible ensures a better impact and a longer lasting significance, as more people will be able to use the developed resources and tools.
One of the latest developments in NLP, and one which promises to have a significant impact for future linguistic processing systems, is the emerging of linguistic annotation meta-systems, which make use of existing processing tools and implement some sort of processing architecture, pipelined or otherwise.
In this paper we describe ALPE, a system offering a new perspective to the task of exploiting NLP meta-systems, by helping a community of users to have an integrated look at a whole range of tools that are able to communicate on the basis of common formats.
For annotated linguistic resources several standardization efforts have been made, such as XCES[1] and TEI[2]. However, the proposed standardizations are not universally accepted, most research projects developing resources according to their own described formats. More recent developments, such as GOLD[3], propose unification methods for the various annotation formats. Due to such methods one can easily transform the name space of a corpus in order to make it compatible to her/his own targets. Several systems tried to facilitate the access to existing processing tools and to ease their usage. The more prominent ones are GATE[4] and UIMA[5]. Both systems make easier the access to a set of independently developed NLP tools which are already parts of an environment offering means to create and use processing chains intended to add linguistic metadata to an input corpus. GATE (Cunningham et al., 2002, Cunningham et al., 2003) is a versatile environment for building and deploying NLP software and resources, allowing for the integration of a large amount of built-ins in new processing pipelines that receive as input a single document or corpus. UIMA (Ferrucci and Lally, 2004) offers the same general functionalities as GATE, but once a processing module is integrated in UIMA it can be used in any further chains without any modifications (GATE requires wrappers to be written to allow two new modules to be connected in a chain). Since the appearance of UIMA, the GATE developers have made available a module that allows GATE and UIMA processing modules to be interchangeable, basically merging the "pool" of modules available.
ALPE, a new NLP meta-system still in development, allows a user, even with very limited programming capabilities, to automatically exploit already walked-on processing paths or to configure new ones on-the-spot, by exploiting the annotation schemas at intermediate steps. ALPE is based on the hierarchy of annotation schemas described in (Cristea and Butnariu, 2004). In this model, XML annotation schemas are nodes in a directed acyclic graph, and the hierarchical links are subsumption relations between schemas. In (Cristea et al., 2006) is described how the graph may be augmented with processing power by marking edges linking parent nodes to daughter nodes with processors, each realising an elementary NLP step.
Section two of this paper presents the theory behind the ALPE system, and section three describes the significant features of ALPE, relevant in the context of a large scale research project, employing multiple layers of annotation schemas and various tools. Section four makes a brief comparison between ALPE and the two most prominent NLP meta-systems (GATE and UIMA). The conclusions, as well as the further planned developments are described in section five.

2. The Underlying Model

2.1 Linguistic Metadata Organised in a Hierarchy

We base our model on the direct acyclic graph (DAG) described in (Cristea and Butnariu, 2002), which

[1] www.xml-ces.org/
[2] www.tei-c.org/
[3] http://www.linguistics-ontology.org/gold.html
[4] http://www.gate.ac.uk/
[5] www.research.ibm.com/UIMA/

configures the metadata of linguistic annotation in a hierarchy of XML schemas. Nodes of the graph are distinct XML annotation schemas, while edges are hierarchical relations between schemas. By interacting with the graph, a user can modify it from an initial trivial shape, which includes just one empty annotation schema, up to a huge graph accommodating a diversity of annotation and processing needs. If there is an oriented edge linking a node A with a node B in the hierarchy (we will say also that A subsumes B or that B is a descendant of A) then the following conditions hold simultaneously:

- any tag-name of A is also in 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.

As such, a hierarchical relation between a node A and one descendant 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. We will agree to use the term *path* in this DAG with its meaning from the support graph, i.e. a path between the nodes A and B in the graph is the sequence of adjacent edges, irrespective of their orientation, which links nodes A and B. As we will see later, the way this graph is being built triggers its property of being connected. This means that, if edges are seen undirected, there is always at least one path linking any two nodes.

2.2 The Hierarchy Augmented with Processing Power

In NLP, the needs for reusability of modules and the language and application independence impose the reuse of specific modules in configurable architectures. In order for the modules to be interconnectable, their inputs and outputs must observe the constraints expressed as XML schemas.

When processes are placed on the edges of the graph of linguistic metadata, the hierarchy of annotation schemas becomes a graph of interconnecting modules. More precisely, if a node A is placed above a node B in the hierarchy, there should be a process which takes as input a file observing the restrictions imposed by the schema A and produces as output a file observing the restrictions imposed by the schema B.

In (Cristea et al., 2006) a graph (or hierarchy) of annotation schemas on which processing modules have been marked on edges is called augmented with processing power (or simply, augmented). The null process, marked Ø, is a module that leaves an input file unmodified.

2.3 Building the Hierarchy

Three hierarchy building operations are introduced in (Cristea et al., 2006): initialize-graph, classify-file and integrate-process. In this section we briefly present them.

The *initialize-hierarchy* operation receives no input and outputs a trivial hierarchy formed by a ROOT node (representing the empty annotation schema). Once the graph is initialised, its nodes and edges are contributed by classifying documents in the hierarchy or manually.

The *classify-file* operation takes an existing hierarchy and a document marked with an XML metadata and classifies the schema of the document within the hierarchy. The operation results in a (possibly) updated hierarchy and the location of the input schema as a node of the hierarchy. If the input document fully complies with a schema described by a node of the hierarchy, the latter remains unchanged and the output indicates this found node; otherwise a new node, corresponding to the annotation schema of the input document, is inserted in the proper place within the hierarchy.

Integrate-process is an operation aiming to properly attach processes to the edges of a hierarchy of annotation schemas, mainly by labelling edges with processors, but also by adding nodes and edges and labelling the connecting edges.

Apart from these basic operations that allow building a hierarchy from scratch or modifying an existing one by exploiting the annotation incorporated in files, a graphical interface allows the user to also define new nodes manually, which ALPE will place at proper places in the hierarchy automatically. But building a hierarchy can be made independent of any explicit interaction with the system by a user. It is still not unusual that an interaction results also in an augmentation of an existing hierarchy with nodes, corresponding to user's input and/or output file. Through multiple interactions, an initial minimal hierarchy which is accessed by a community of users can thus be developed.

2.4 Operations on the Augmented Graph

Three main operations can be supported by the Cristea et al. (Cristea et al., 2006) model.

If an edge linking a node *A* to a node *B* (therefore *B* being a descendant of *A*) is marked with a process *p*, it is said that ***A* pipelines to *B* by *p***. Equally, when a file corresponding to the schema *A* is pipelined to *B* by *p*, it will be transformed by the process *p* onto a file that corresponds to the restrictions imposed by the schema *B*. This arises in augmenting the annotation of the input file (observing the restrictions of the schema *A*) with new information, as described by schema *B*.

For any two nodes *A* and *B* of the graph, such that *B* is a descendant of *A*, it is said that ***B* can be simplified to *A***. When a file corresponding to the schema *B* is simplified to *A*, it will lose all annotations except those imposed by the schema *A*. Practically, a simplification is the opposite of a (series of) pipeline(s) operation(s).

The **merge** operation can be defined in nodes pointed by more than one edge on the hierarchical graph. It is not unusual that the edges pointing to the same node are labelled by empty processors. The merge operation applied to files corresponding to parent nodes combines the different annotations contributed by these nodes onto one single file corresponding to the schema of the emerging node.

With these operations, the graph augmented with processing power is useful in two ways: for goal-driven, dynamic configuration of processing architectures and for transforming metadata attached to documents. Automatic configuration of a processing architecture is a result of a navigation process within the augmented graph between a start node and a destination node, the resulted processes being combinations of branching pipelines (serial simplifications, processing and merges). In terms of processing, the difference with respect to GATE and UIMA, both allowing only pipeline processing in which the whole output of the preceding processor is given as input to the next processor, is that in the described model the required processing may result in a combination of branching pipelines. This is due to the introduction of the merge operation which is able to combine two different annotations on the same file. Once the process is computed, then it can be applied on an input file displaying a certain metadata in order to produce an output file with the metadata changed as intended. These two files comply with the restrictions encoded by the start node and, respectively, the destination node of the hierarchy.

Since the graph is connected, there should always be at least one path connecting these two nodes. The paths found are made up of oriented edges and, depending on whether the orientation of the edges is the same as that of the path or not, we will have pipeline operations or simplification operations. A **flow** is a combination of paths between the start and the destination node that configures the processing which transforms any file observing the specifications of the start node (schema) onto a file observing the specifications of the destination node (schema).

Once the entry and exit points in the hierarchy have been determined and processing flows (combination of paths in the graph) have been devised, all the rest is done by the hierarchy augmented with the processing power in the manner described above. This way, the processing needed to arrive from the input to the output is computed by the hierarchy as sequences of serial and parallel processing steps, each of them supported in the hierarchy by means of specialized modules. Then the process itself is launched on the input file.

2.5 ALPE

ALPE is a system implementing the described model. Besides implementing all the previously described features, ALPE brings several additions.

The core modules

ALPE includes 11 core modules, used in any ALPE hierarchy (the hierarchy augmented with processing power, as described) but not attached to any edge. These core modules perform built-in tasks such as language

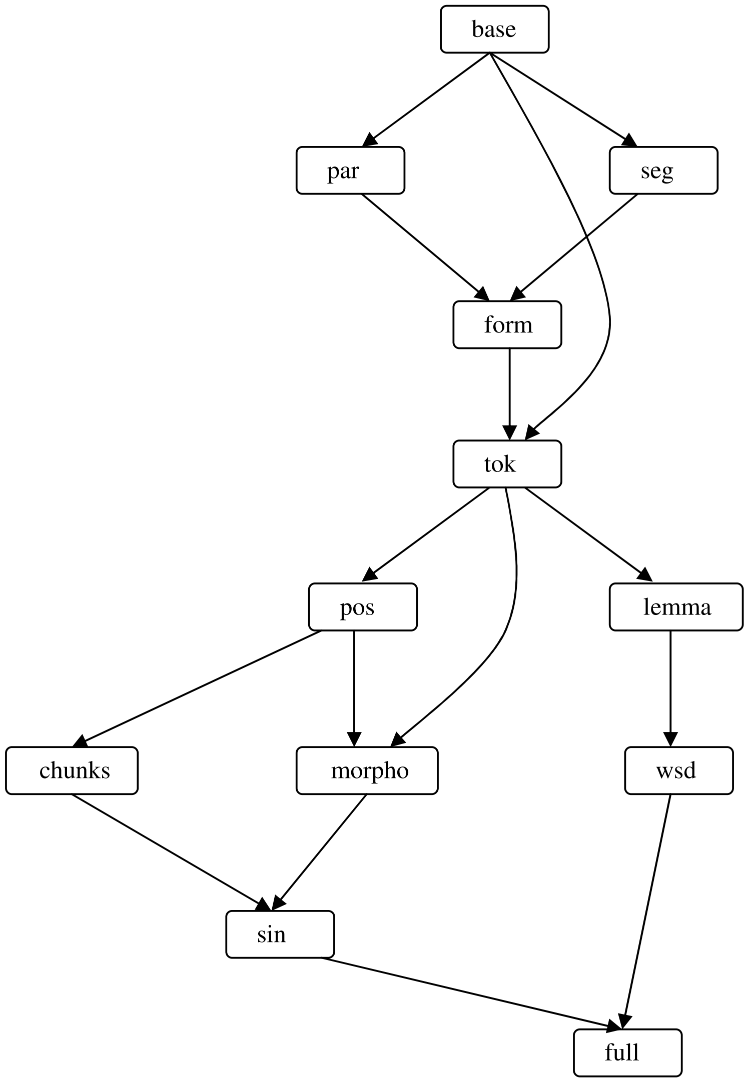

Figure 1: The ALPE core hierarchy

identification, but also implement the basic operations in the hierarchy (among others, flow computation, merging and simplifying). These core modules are used in any ALPE hierarchy and are not replaceable by user tools. They ensure that any ALPE hierarchy implements the basic behaviour, as described in this paper.

The core hierarchy

One of the main problems in developing a new NLP system is selecting a relevant and useful annotation format for the developed resources. Establishing a hierarchy of generally used XML metadata is not one of ALPE's main purposes, but having most annotated documents adhere to some common format brings obvious benefits both to the developer of new NLP software and to the user who would have an easier time finding the tools required for a particular annotation task. As base for any new ALPE hierarchy is offered a core hierarchy, with 12 annotation schemas ranging from basic XML format to a full XCES (Ide et al., 2000) linguistic annotation specification[6]. The intermediate formats are designed to conform to specific requirements for document annotation, such as tokenization, POS-tagging, NP-chunking, etc. as well as combination of these markings. Figure 1 shows the ALPE core hierarchy. All nodes are subsets of the XCES standard for annotated data, and the subsumption relation is observed between all pairs of nodes linked through an edge.

The 12 nodes in figure 1 correspond to XML annotation schemas as follows:

- *base*: subset of XCESAna including just *cesAna* tags – corresponding to a basic XML format;
- *par*: adds the *par* tag to the parent node – corresponding to an XML with marked paragraphs;
- *seg*: adds the *s* tag to the parent node – corresponding to an XML with marked sentences;
- *form*: a merge of the subsuming formats – corresponding to an XML with marked formatting (paragraphs and sentences) information;
- *tok*: adds the *tok* and *orth* tags to the parent node – corresponding to a tokenized text;
- *pos*: adds the *ctag* tag to the parent node – corresponding to a pos-tagged text;
- *lemma*: adds the *base* tag to the parent node – corresponding to a lemmatized text;
- *chunks*: adds the *chunk* and *chunklist* tags to the parent node – corresponding to a (Noun/Verb) phrase-chunked text;
- *morpho*: adds the *msd* tag to the parent node – corresponding to an XML displaying morphological metadata;
- *wsd*: adds a *wsd* tag for semantic disambiguation;
- *sin*: merges the parent nodes – corresponding to an XML displaying full syntactic information;
- *full:* merges all parent nodes.

The purpose of the core hierarchy is to offer both a starting point to any new hierarchy as well as anchors for any new linguistic annotation formats that a user would like to include. When the XML formats of the user's input and output files are not identical with schemas belonging to the hierarchy (for instance, due to differences in the tags name space or to configurations of attributes that convey in different ways the same information) then the user has to provide convertors (wrappers) able to accommodate his notations with those corresponding to nodes of the hierarchy.

The user's needs and the selection of flows

The ALPE augmented hierarchy can be used in many ways. Suppose a user wants to process an XML file from one input format to some output format. In principle, any such processing task involves a transformation by some module capable to receive the input format and to output the required final format. The ALPE philosophy details such a processing task in relation with the pair of input-output schemas by establishing the way these schemas interrelate from the point of view of the subsumption relation. Two cases can be evidenced: either the two schemas do observe a subsumption relation or not. When they do, then the node corresponding to the input file can be connected through a direct descending or ascending edge to the one corresponding to the output file. It will be descending if the output schema results from the input schema through some adds, and it will be ascending if in order to obtain the output, simplification applied to the input are required. When the two schemas are not in a subsumption relation, then there should be a node such that either both are subsumed by it, or both subsume it.

ALPE comes with a core hierarchy whose nodes act as a grid of fixed bench-marks with respect to which the locations of the input and output schemas are set out. When the pair of users' schemas matches two nodes of the core hierarchy, then processing can be drawn in terms of known (built-in) interconnected modules. When a match (modulo, as noticed above, the XML elements name space and/or differences in configurations of attributes still conveying the same information) of one or even both of user's schemas against nodes of the hierarchy is not possible, then the non-matching schemas should be seen as new nodes of the hierarchy. In this case it is the user's responsibility to locate also the processes which will be assigned to the new edges which will interconnect the new nodes onto the hierarchy.

ALPE designs a solution to the user's problem by first computing all possible chains of edges which link the input schema to the output schema and, if needed, executing them.

Each computed flow is characterized by a set of features. These features include properties such as: flow length (defined as number of processing steps involved), cost (for instance, if processing involving one or more modules presupposes financial costs), the estimated precision of execution, and the estimated time of execution. The user can then select and run the flow most suitable to his needs.

3. Features

In this section we will describe a set of features implemented in ALPE often wished for in environments working with linguistic resources and tools. We will see how these features emerge from the model described above. Many of these features are key elements of the

[6] http://www.cs.vassar.edu/XCES/dtd/xcesAna.dtd

future European linguistic infrastructure, as seen by CLARIN[7].

Multilinguality

In modern NLP, algorithms are separated from linguistic details. This way, a module designed to perform a specific task can be put to work on any language if fuelled with appropriate language resources. This is the case, for instance, with POS-taggers (see, for instance, TNT (Brants, 2000)), which are powered by specific language models (frequency of n-grams of POS tags). A syntactic parser should be powered by the grammar of a language to be effective in parsing sentences of that language. A shallow parser, which usually implements an abstract automata machinery, could recognize noun phases of one language if powered by a resource consisting of a set of regular expressions specific to that language.

To implement multilinguality within the proposed model means to map the edges of the augmented graph on a collection of repositories of configuring resources (language models, sets of grammar rules, regular expressions, etc.) which are specific to different languages. This can be achieved if the edges of the graph labelled with processes are indexed with indices corresponding to languages. This way, to each particular language an instance of the graph can be generated, in which all edges keep one and the same index – the one corresponding to that particular language. This means that all processors of that particular language should access the configuring resources specific to that language in order for the hierarchy to work properly. For instance, in the graph instance of language *Lx*, the edge corresponding to a POS-tagger has as index *Lx*, meaning that it accesses a configuring resource file that is specific to language *Lx* (that language model).

It is a fact that different languages have different sets of processing tools developed, English being perhaps the richer, presently. Ideally, the blame for the lack of a tool in a specific language should be put on the lack of the corresponding configuring resource, once a language independent processing module is available for that task. It is also the case that differences exist in processing chains among languages. For instance one language could have a combined POS-tagger and lemmatizer while another one realizes these operations independently, pipelining a POS-tagger with a lemmatization module. These differences are reflected in particular instances of sections of the graph, which, although reproduce the same set of nodes, do not allow but for certain edges linking them. The missing edges inhibit pipelining operations

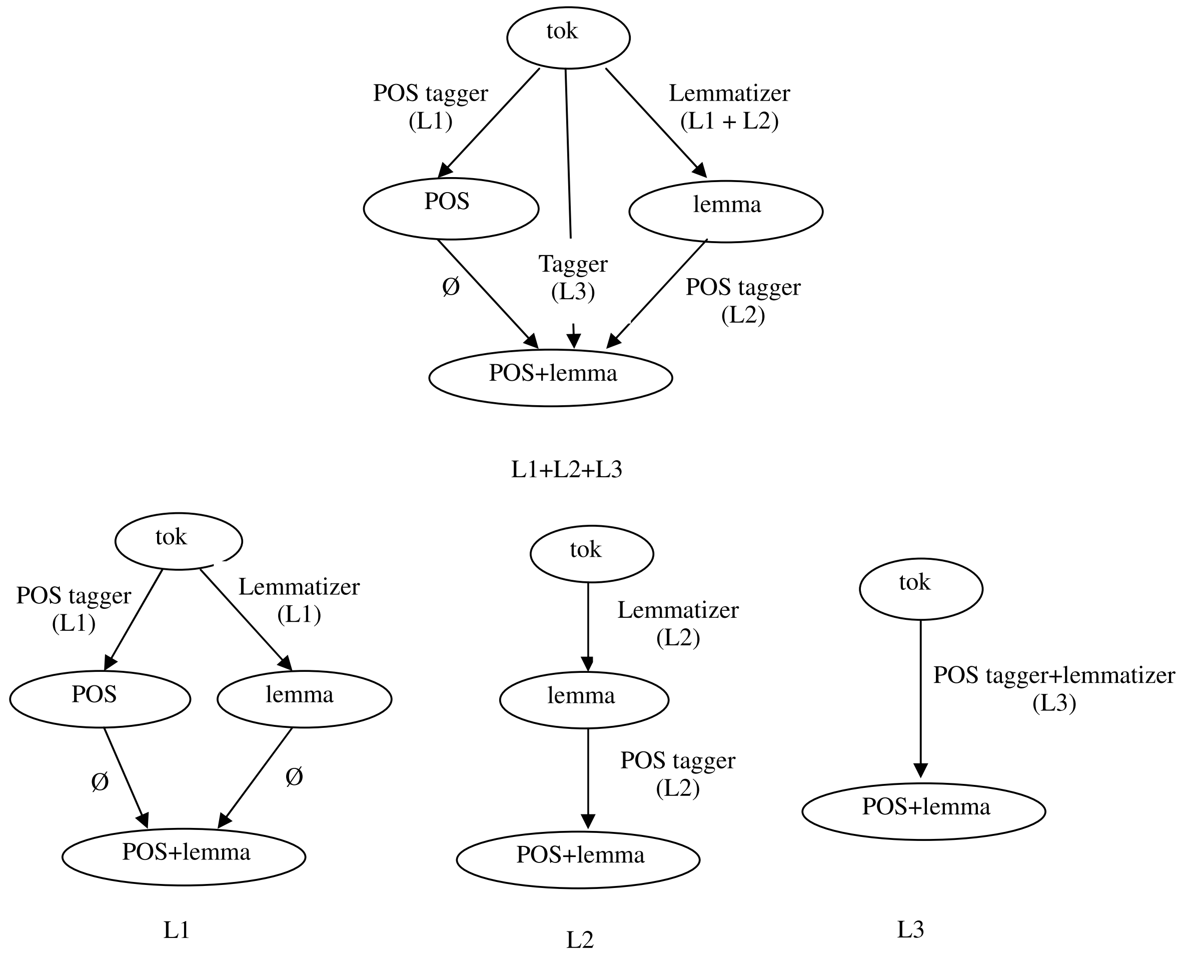

Figure 2: Computation of different flows for specific languages

[7] http://www.clarin.eu

along them, but are suited for simplification operations.
In figure 2 is given a simple example of how ALPE handles multiple languages integrated in the same hierarchy. The first hierarchy (marked as L1+L2+L3 in the figure) has four nodes (annotation schemas):

- *tok*: XML which marks lexical tokens;
- *POS*: XML marking tokens and their part-of-speech;
- *lemma*: XML marking tokens and their lemmas;
- *POS+lemma*: XML with tokens, POS and lemma information.

These four nodes correspond to simple processing stages for linguistically annotated documents. The ALPE hierarchy fragment representation (shown on the L1+L2+L3 section of Figure 2) indicates the subsuming relations between the respective nodes and the attached tools. For each tool, in parenthesis, it is indicated the languages for which the tool is available. In the sections marked L1, L2 and L3, respectively, of Figure 2 are sketched the corresponding instantiations of this sub-hierarchy for the three languages.
The user can provide an input document (XML with marked lexical tokens) and specify the required output format as being the final node (suppose *POS+lemma*). ALPE determines the language of the input document (as being L1, L2 or L3). If the input document belongs to the language L1, the computed flow will include only tools available for that language. Thus the only possible flow will use the *POS tagger* and the *Lemmatizer* tools, then merge their results into the output format. For the second language the flow will use a different *POS tagger* tool, one that requires as input a file corresponding to the *lemma* node. So the computed flow will run first the *Lemmatizer*, then the *POS tagger* on the result. For the third language, a tool is available that can directly annotate an input file in the *tok* format up to the required output.
We can look at the ALPE hierarchy as having three layers, one for each language. The three language specific hierarchies can look completely different for each language, but are still able to compute and run the same flows as the combining hierarchy. The three layers are aligned by nodes which display the same XML structure.

Manual versus automatic annotation

We have seen how automatic annotation is supported by the augmented graph. But how can manual annotation be accommodated within this approach?
Usually, in order to train processing modules in NLP, developers use manually annotated corpora. To create such corpora, they make use of annotation tools configured to help placing XML elements over a text, and to decorate them with attributes and values. As such, if annotation tools do, although in a different way, the same jobs which can be performed by processing modules, it is most convenient to associate them with edges in the graph in the same way in which processing modules are associated with these edges.
Meanwhile, it is clear that manual annotation cannot be chained in complex processing architectures in the same way in which automatic annotation can. In order to differentiate between automatic and manual processes, as encumbered by pairs of schemas observing the subsumtion relation, it results that edges should have facets, for instance AUT and MAN. Under the AUT facet of a POS-tagging edge, for instance, the automatic POS-tagger should be placed, while under the MAN facet – the POS-tagging annotation tool should be placed.
The configuration files of these tools can usually be separated from the tools themselves. We can say that the corresponding configuration files particularise the annotation tools, which label edges of the graph, in the same way in which language specific resources particularise processing modules.

IPR and cost issues

Intellectual property rights can be attached to documents and modules as access rights. Only a user whose profile corresponds to the IPR profile of a resource/tool can have access to that file/service. As a result, while computation of processing chains within the hierarchy is open to anybody, the actual access to the dynamically computed architectures could be banned to users which do not correspond to certain IPR profiles of certain component modules or resources they need.
More than that, some price policies can be easily implemented within the model. For instance, one can imagine that the computation of a flow results also in a computation of a price, depending on particular fees the chained Web servers charge for their services.
Out of this, it is also imaginable the graph as including more than one edge between the same two nodes in the hierarchy. This can happen when different modules performing the same task are reported by different contributors. When these modules charge fees for their services, it is foreseeable also an optimization calculus with respect to the overall price over the set of paths that can be computed for a required processing.

Facing the diversity of annotation styles

It is a fact that, today, a huge diversity of annotation variants circulates and is being used in diverse research communities. It is far from us to belief that a Procustean Bed policy could ever be imposed in the CL or NLP community, that would aim for a strict adoption of standards for the annotated resources. On the other hand, it is also true that efforts towards standardization are continually being made (see the TEI, XCES, ISLE, etc. initiatives). Moreover, Semantic Web, with its tremendous need for interconnection and integration of resources and applications on communicating environments, boosts vividly the appeal for standardization. It is therefore foreseeable that more and more designers will adopt recognized standards, in order to allow easy interoperability of their applications. A realistic view on the matter would bring into the focus the standards while also providing means for users to interact with the system even if they do not rigorously comply with the standards.
We have seen already that, by classification, any schema could be placed in the hierarchy. Of course, classification could increase in an uncontrollable way the number of nodes of the hierarchy. The proliferation could be caused not so much by the semantic diversity of the annotations, as by the differences in name spaces (names of tags and attributes).
Technically, this can be achieved by temporarily creating links between the new schema classified by the hierarchy, as a new node, and its corresponding schema in the hierarchy. Processing along such a link is different than

the usual behaviour associated to the edges of the graph and is specific to wrappers. It describes a translation process, in which the annotation is not enriched, but rather names of XML elements and attributes are changed. Ideally, the processing abilities of the hierarchy should include also the capability to automatically discover wrapping procedures. This task is not trivial since it would require that the hierarchy "understands" the intentions hidden behind the annotation, displaying, this way, some kind of semantic processing capabilities which is not easy to implement. However, recent initiatives as GOLD make us believe that significant steps forward in this direction are near us.

4. Evaluation

4.1 ALPE vs. GATE and UIMA

In this section we will compare functionalities of ALPE with those of GATE and UIMA, systems which can give very similar results with our.

First of all, ALPE is intended primarily to facilitate the user's interaction with the system, allowing for an programming non-expert to integrate resources and tools. As a standalone linguistic processing environment, the user is presented with a visual representation of a hierarchy of annotation formats and has basically three main choices: s/he can add a new resource to the hierarchy (for example enabling an already integrated processing module to work for another language by adding a corresponding language model), add a new processing tool (attached to an existing edge, or attached to a newly created edge) or compute and use a processing chain (providing the input file and selecting the output format). GATE offers a user interface adequate for creating and using processing chains. Chains have to be built manually and presuppose an intimate knowledge of the system. UIMA is even more oriented to the NLP professional, offering little in terms of visual user interaction. A direct comparison that would put on stage quantitative evaluations is difficult to be made for these kinds of systems. Perhaps a better prospect would be a qualitative comparison performed by a significant pool of users, providers as well as consumers of language resources and tools. In the following, we make just an estimative comparison, but a qualitative evaluation versus human performance is planned.

Every one of the three main functionalities (adding a new resource, adding a new tool, and computing and using a processing chain) is easier to perform in ALPE. Both UIMA and GATE require some formal description to be written for each new resource integrated into the system, while ALPE generates these formal descriptions automatically. When adding a new processing tool, ALPE has much more permissive restrictions with regard to what tool can be integrated: it basically has to be either a webservice or a command line, executable under Windows or Linux. GATE allows the user to integrate at least Java and Perl based tools, and this is done by writing some dedicated code, a task which is however above the capabilities of some users. UIMA is even more restrictive, allowing only C++ based tools to be integrated, and only after significant implementations and changes to the original code. However, an extension allowing modified Perl, Python and TCL modules to be integrated is available.

An evident advantage of ALPE over both GATE and UIMA is that the processing chains in ALPE are automatically computed, therefore requiring no human intervention. Moreover, they can be created between any two formats defined in the hierarchy (providing the modules decorating the connecting edges are available, otherwise there are signalled as missing). ALPE deals with multilinguality, thanks to its core module that performs language identification for each input file, then selects to corresponding tools and language resources, if available. GATE and UIMA are mainly focused on English (GATE incorporating also modules dedicated to some other languages), but the user has to make sure to select the proper modules when designing a processing chain for a document in other language than English.

Let us consider the example of a use-case in which the user has two processing tools s/he wants to use on the same input file and to merge the results in an output file. Using ALPE, this user has to specify the input/output formats of the modules, then let the system integrate the tools as arches linking the corresponding nodes in the hierarchy (in the case when one of both of these formats are not currently part of the hierarchy, they will become as such), then input the file and specify the required output format (node). Using GATE, the user has to implement the integration of the tools to make them available to the processing chain building interface, then build and run two processing chains, one for each tool, then merge the results outside GATE (since it does not allow parallel processing and merging of annotations). UIMA performs this task basically in the same way as GATE, requiring even more implementation when integrating the new tools, but allows annotation merging.

4.2 Qualitative evaluation

In order to evaluate ALPE versus human computational linguistic specialists, we have developed an ALPE augmented hierarchy configured for a current research project involving documents in 9 European languages (Bulgarian, Czech, English, German, Dutch, Maltese, Polish, Portuguese and Romanian) and using a significant number of language processing tools[8]. All documents have to be annotated according to 6 main annotation formats (and 8 optional ones), resulting a significant hierarchy of standards. This hierarchy is already implemented and serves as a management and access facility for the collected documents.

At the time of writing this paper, an ALPE core hierarchy specific to the mentioned project is implemented for English and Romanian.

5. Conclusions

We think that the model we propose and its first implementation, as the ALPE system, encapsulate different organisational, standardisation and processing features which make it interesting for the goals of a project like CLARIN.

In this proposal we have been concerned with the following features of functionality, also identified as of

[8] LT4eL – an FP6 project (www.lt4el.eu)

primary importance in CLARIN[9]:

- **unique access gate and distributivity**: although distributed in different places, LR and LT could be, in the vision described in this paper, identified through a single access gate;
- **metadata policy**: primary text and speech documents should be given the possibility to be accompanied by metadata describing human and/or automatic annotation over them. The ALPE conventions allow for the metadata to have a form which make it easily removable when the primary raw documents are needed of being recuperated;
- **independence of representation**: it is clear that the XML representation adopted by ALPE allows for LR to be manipulated in such a way as to benefit of the same treatment irrespective of the particular metadata conventions;
- **quick access**: ALPE comes very close to the objective that CLARIN LR and LT be accessed instantaneously from all over Europe;
- **conversion services**: the ALPE approach incorporates features that allows easy conversion operations from and onto different representations;
- **processing services**: the ALPE portal provides processing services for enrichment and or simplification of metadata attached to LR;
- **versioning**: the portal allows manipulation of different versions of data as well as of the metadata accompanying the texts;
- **multilinguality**: the structure allows uniform treatment of documents in different languages, as well as of parallel texts;
- **IPR issues**: the structure provides means of dealing with IPR.

In this paper we have described a model of dynamical building of processing architectures based on a hierarchy of XML schemas and an implementation – called ALPE. We have argued that ALPE brings some advantages over other known systems with similar objectives, mainly coming from a plus in manoeuvrability and complete automation of the configuring tasks. It is also shown how ALPE, has brought already significant advantages in the context of a multilingual research project. In this context ALPE has automatically configured complex processing chains involving several modules and documents in different languages. The features brought by the addition of an ALPE type hierarchy into a complex project contribute significantly to acquire multilinguality, distributivity, versioning of language resources, automatic and manual annotation, management of IPR and cost issues, as well as managing diversity of annotation styles, features that the CLARIN project considers of extreme importance.

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.

[9] We foresee that other requirements, as, for instance, discovery of resources and tools, preservation of resources, archiving services, content discovery, distribution, authentication and authorization, could also be designed around the structure we propose.

Acknowledgments

Part of the work for the paper was supported by the ROTEL (CEEX project) AMCSIT contract no. 29/03.10.2005, the CLARIN INFRA-2007-2.2.1.2 project, and the FP6 LT4eL project.

References

T. Brants (2000): TnT: a statistical part-of-speech tagger. In Proceedings of the sixth conference on Applied Natural Language Processing, Seattle, Washington, pag: 224 – 231.

D. Cristea, C. Butnariu (2004): Hierarchical XML representation for heavily annotated corpora. In *Proceedings of the LREC 2004 Workshop on XML-Based Richly Annotated Corpora*, Lisbon, Portugal.

D. Cristea, C. Forăscu, I. Pistol. (2006): Requirements-Driven Automatic Configuration of Natural Language Applications. In Bernadette Sharp (Ed.): *Proceedings of the 3rd International Workshop on Natural Language Understanding and Cognitive Science - NLUCS 2006*, in conjunction with ICEIS 2006, Cyprus, Paphos, May 2006. INSTICC Press, Portugal. ISBN: 972-8865-50-3.

H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan. (2002): GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the ACL (ACL'02)*. Philadelphia, US.

H. Cunningham, V. Tablan, K. Bontcheva, M. Dimitrov. (2003): Language engineering tools for collaborative corpus annotation. *Proceedings of Corpus Linguistics* 2003, Lancaster, UK.

D. Ferrucci and A. Lally. (2004): UIMA: an architectural approach to unstructured information processing in the corporate research environment, *Natural Language Engineering* 10, No. 3-4, 327-348.

N.Ide, Bonhomme P., Romary L. (2000) : XCES: An XML-based Encoding Standard for Linguistic Corpora, *Proceedings of the Second International Language Resources and Evaluation Conference*. Paris: European Language Resources Association