Multilingual Linguistic Workflows

Dan Cristea[1,2], Ionuţ Cristian Pistol[1]

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

1. Introduction

The field of Natural Language Processing (NLP) has seen important developments over the later years, most significantly in efforts intended to raise the quality and quantity of resources, to enhance the performance and diversity of tools and to open accessibility to both resources and tools as largely as possible. The demands of multilinguality at the level of language technology impose the necessity of reusing for different languages the processing modules performing specific linguistic tasks. The language a module is able to interpret becomes thus commanded by the resources it is fuelled with. Such a view on interoperability requires a standardization of the processing steps and an efficient building and execution of workflows.

But the issue of language technology addressing the needs of multilinguality has a lot more facets then strict reusability of resources and tools, as for instance, the easiness of adopting resources in resource-poor languages from resource-rich languages, abstracting the ways in which language dependency issues are regarded in approaches involving language processing tasks, or a uniform way of looking at annotations schemas provided by different schools and overpassing language barriers.

Standardization of metadata formats and of the NLP software were, among others, the goals of projects such as CLARIN[1] and

[1] http://www.clarin.eu/external/

FLaReNET[2], as well as several national and international workshops and conferences. Meta-systems, capable to offer to a diversity of users access to libraries of processing modules, as well as interfaces that help building complex processing architectures out of these modules, are two of the most wanted behaviours in the NLP field. Systems, such as GATE[3] (Cunningham et. Al, 2002) and UIMA[4] (Ferruci and Lally, 2004), and research efforts such as PANACEA[5] and "Heart of Gold"[6] represent some of the most prominent efforts in this direction.

Almost all of the most influential NLP frameworks respond very well to requirements specific to different languages. To take just one notorious example, UIMA is used as an integration and unifying framework in many multilingual projects. The project ATLAS[7], for instance, builds complex processing chains that perform translation and summarisation of documents in 7 languages: Bulgarian, Croatian, English, German, Greek, Polish and Romanian, and uses UIMA as a compatibility standard. Another project, METANET4U[8], among other things, updates, enhances and disseminates a large spectrum of language resources and tools in at least 6 languages: Catalan, English, Spanish, Maltese, Portuguese and Romanian (Branco et al., 2011), and UIMA is there also in the central interest of the consortium. More and more resources in more and more languages are accumulated and/or advertised on big portals[9]. The more numerous these resources will be, the bigger the need of interconnectivity in complex multilingual applications.

ALPE (the Automated Linguistic Processing Environment) has been reported (Pistol and Cristea, 2009; Pistol, 2011) as being a

[2] http://www.flarenet.eu/
[3] http://gate.ac.uk/
[4] http://uima.apache.org/
[5] http://www.panacea-lr.eu/en/
[6] http://www.delph-in.net/heartofgold/index.html
[7] PSP-ICT grant #250467, http://www.atlasproject.eu/atlas/project/en

[8] PSP-ICT grant #27089, http://www.meta-net.eu/projects/METANET4U/

[9] See, for instance, the META-SHARE initiative of the META consortium.

format representation and processing environment which makes use of annotation schemas arranged hierarchically in a hyper-graph in order to compute automatically workflows out of a pool of processing components. The model, called the *Formats and Modules Hierarchy* (FMH) is designed to help users to build complex processing architectures, by involving minimum of expert skills.

Although rather well studied from different perspectives (Pistol, 2011), the FMH model has potentials yet unexplored sufficiently attaining the multilingual aspects of language processing. In this paper we describe the FMH model with a special emphasis on its capacities to deal with multilingual aspects of linguistic processing.

The paper is organised as follows. Section 2 shortly presents the FMH model and section 3 shows how processing flows are treated in the model. The next two sections discuss in detail a couple of applications, with the emphasis on the model's capacity to handle the specific needs of multilinguality, and the last section presents conclusions and further work.

2. The Formats and Modules Hierarchy

In NLP applications, very often, a task is accomplished through a pipeline of chained linguistic processing modules, each adding a supplementary level of details to the data it receives in the input. In certain cases, a processing chain can fork, such that independent processing be performed in parallel and the results, when accomplished on all branches, be collected on a common file. At times, it could also be necessary a certain pruning of the notations accumulated in the intermediate files, in order to get rid of some bookkeeping notations and to retain only the data relevant for the following steps. Conversions could also be applied to intermediate files in order to make them compatible to the format accepted by the subsequent modules. In all these cases, the intermediate results take the form of notations applied to text files (marked in XML, Lisp, table forms, etc.).

For reasons of space, we give here only a brief description of the model, further details being found in (Cristea et al., 2006; Cristea and Pistol, 2009). At the core of the FMH model stays a hyper-graph, with nodes being abstractions of annotation formats (called schemas) and edges corresponding to various linguistic and conversion tasks. Simplifications are realized always along reverse directions of linguistic edges, sometimes even short-circuiting more processing steps. They are not explicitly marked on the hyper-graph, but are seen by the navigation algorithm when searching for workflows. Conversions neither add nor delete information and, as such, are specially marked edges. Hyper-edges appear when two or more annotation schemas can be merged to produce one that includes all common information, as well as all specific information of the contributing nodes. In this case, the destination node of the hyper-edge is called a merge node. A path in a hyper-graph links a start node to a destination node and corresponds to a processing workflow. It may include linguistic processors, simplifications, conversions and merges.

The constituent bricks of the annotation schemas are:

- a list of tags (element names in XML, list names in LISP, etc.);
- the specific attributes describing each annotation tag (attributes of an XML element, fields of a list in LISP, etc.);
- any restrictions the annotation tags and their attributes should observe, for instance, if a tag can be found only if paired with another one, or if it is always nested in another tag.

Although annotation schemas can support other formats than XML, it is easily arguable that these can be mapped (losslessly) to XML equivalents, for instance by considering the class identifiers (or table names in relational databases) as XML elements and the attribute-value pairs (specified features) as XML attribute-values. This is a frequent assumption made whenever conversions are performed between non-XML and XML formats. As such, our discussions below will be drawn on examples considering only XML formats.

As said, in the FMH hyper-graphs, nodes represent the annotation formats. Edges between nodes are originating in two

sources: the subsumption relation (Cristea and Butnariu, 2004) and the conversion relation. The definition of the *subsumption relation* is adapted from the notion of subsumptions between feature structures (for instance, Gazdar and Mellish, 1989): a format A subsumes a format B ($A \subseteq_s B$) if B includes all elements and attributes of A, possibly also additional data (element names and attributes). Also, all annotated documents observing the restrictions described by the format B also observe the restrictions described by A. For example, for $A \subseteq_s B$, Figure 1 shows two XML annotated documents, the first conforming to a simple format (A), the other to a slightly more complex format (B).

As such, if two documents represent the same hub text and their XML annotation conform to two schemas A and B which are in a subsumption relation $A \subseteq_s B$, then they are comparable, in the sense that the one corresponding to B includes all the information that the document corresponding to A has (possibly also something more). The subsumption relation is anti-symmetric, reflexive and transitive and the strict subsumption is anti-reflexive.

```
<document>
<tok id="1">This</tok>
<tok id="2">is</tok>
<tok id="3">an</tok>
<tok id="4">example</tok>
<tok id ”5”>.</tok>
</document>
```

A

```
<document>
<seg id=”1”>
  <tok id="1" pos=”p”>This</tok>
  <tok id="2" pos=”v”>is</tok>
  <tok id="3" pos=”d”>an</tok>
  <tok id="4" pos=”n”>example</tok>
  <tok id=”5” pos=”m”>.</tok>
</seg>
</document>
```

B

Figure 1. Documents observing the restrictions of A and B, where $A \subseteq_s B$.

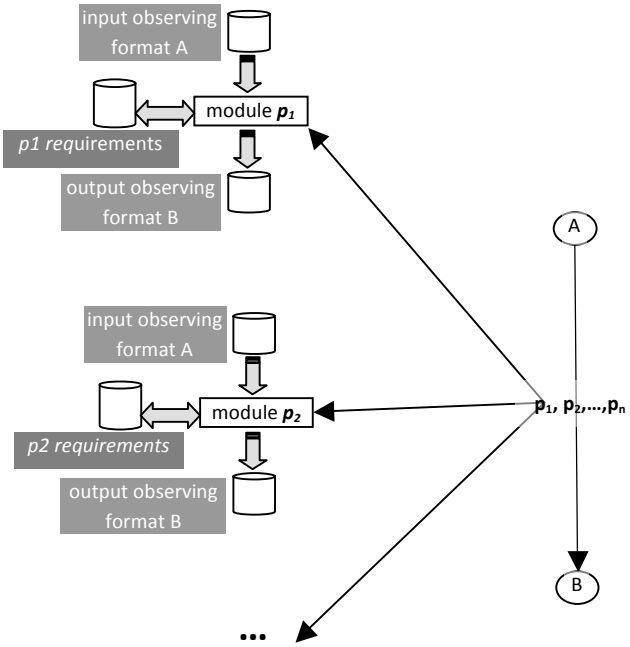

Figure 2: Multiple processing modules attached to the same processing edge

Among all pairs of schemas which are not comparable, one class distinguishes, namely schemas sharing the same semantic content, although in different forms. As such, if two schemas C and D represent the same information in different formats, a document corresponding to C has neither more nor less information than the one corresponding to D (when both documents share the same hub text). In this case we say that the nodes corresponding to C and D are in a *conversion relation*. The conversion relation is symmetric, reflexive and transitive.

Originating in these two distinct types of relations characterising pairs of nodes in the hyper-graph, the model defines four types of edges, all directional:

- *Processing edges* are found between all nodes A and B which are in a subsumption relation, namely $A \subseteq_S B$, and such that

the model knows about at least one processing module (called also a pipeline module) accepting as input the format A and producing as output the format B. This means that if a file conforming to the format A is available, the format B can be obtained after processing the file with the corresponding processing modules. Modules attached to the same edge can differ in terms of language restrictions, costs and/or running peculiarities, by the resources they require (as seen in Figure 2), etc. The direction of the edge is from A to B.

- *Simplification edges* are implicitly considered between all nodes B and A, where $A \subseteq_s B$. This means that if the format B is available, the format A can be satisfied by removing the additional data in B (thus, simplifying B to A). The direction of the edge is from B to A.
- *Merge edges*: a hyper-edge connects a set of formats (A_1,..., A_k) to another format B, which has the property of being the minimal upper bound of A_1,..., A_k (considering strict subsumption as an ordering relation). Thus, all annotation information in any of A_1,..., A_k, can also be found in B and there is no piece of information in the format B which could not be found also in at least one of the formats A_1,..., A_k. Speaking in terms of documents conforming to these schemas, if f_{A1},..., f_{Ak} are files observing the formats A_1,..., A_k, all sharing the same hub text, then a file f_B observing the merging format B can also be generated. The direction of the hyper-edge is from A_1,..., A_k to B.
- *Conversion edges* are found between all nodes A and B which are in a conversion relation and such that the model knows about at least one processing module accepting as input the format A and producing as output the format B. The difference between *processing* and *conversion* edges resides in the nature of the transformation module. If the transformation does not modify the nature of the information in the input, but rather puts it in a different form, then we are in the case of a convertor (wrapper) and the edge linking the input to the output format is of a conversion type. In general, if a convertor can bring format A to format B, then there should exist a convertor performing the reverse transformation. The direction of the edge is from A to B.

In all the figures displaying FMH hyper-graphs in this paper subsumption relations to whom correspond processing modules are

indicated as arrowed thin full lines; when there is no processing module along a subsumption relation the edge is indicated as an arrowed dotted line; simplification edges are sometimes shown as arrowed interrupted lines; merge hyper-edges are drawn as thick arrowed lines connecting a group of nodes surrounded by an oval to a destination node; and conversion edges are marked (not appearing in this paper) as double lines arrows.

3. FMH processing flows

A processing flow is a sub-graph of a FMH hyper-graph which includes all and only the nodes and edges on a path linking a pair of nodes, called *start* and *destination*, in the direction of the edges. The path may include all the four categories of edges: processing, simplification, merge and conversion. Given a pair of nodes start-destination on a FMH hyper-graph, there could be found none, one or more processing flows. A processing flow models a possible set of processing activities, which, if applied to a hub document, can transform it from the format of the start node to the format corresponding to the destination node.

Flows are directed paths and, as no two edges between the same nodes and with the same orientation can exist in FMH, there is no ambiguity in describing flows as sequences of nodes. Some examples of flows on the FMH of Figure 3 are:

- start=A, destination=C_1: the flow includes the pipeline edges a and b traversing the sequence of nodes (A, B, C_1);
- start=C_2, destination=A: the flow includes a simplification edge connecting the nodes (C_2, A);
- start=A, destination=D: the flow includes two pipelines, a+b and c, and a merge hyper-edge to combine two intermediate formats, and traverses the nodes ((A, B, C_1), (A, C_2)), D).

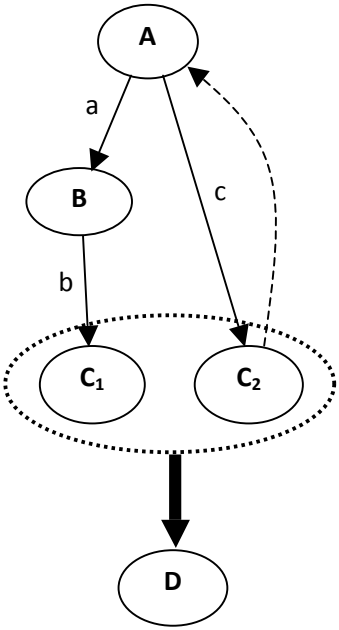

Figure 3: A simple FMH for flow exemplification

To describe processing flows, the following notations will be used:

- $A >_a B$ designates the use of a pipeline module *a* linking nodes *A* and *B* (reads "*A* pipelines to *B* by *a*");
- $B < A$ designates the use of a simplification process between nodes *B* and *A* (reads "*B* is simplified to *A*");
- $(A_1, ..., A_k) > B$ designates a merge process between the hyper-node $(A_1, ..., A_k)$ and *B* (reads "$A_1, ..., A_k$ merges to *B*"). The same notation applies if some or all of the terms of the merge are flows instead of simple nodes. In this case, instead of a node *X*, a whole flow having as destination the node *X* is noted in the merge;
- $A \rightarrow_a B$ designates the use of a conversion module *a* linking nodes *A* and *B* (reads "*A* converts to *B* by *a*");

The examples put in evidence above on Figure 3 are noted as follows:

- $A >_a B >_b C1$
- $C2 < A$
- $(A >_a B >_b C1, A >_c C2) > D$

A processing flow may be further characterized by several features, among which are flow length and flow density:

Flow length measures the total number of edges of the flow. This feature gives a rough estimation of the effort necessary to execute the flow. However, some of these edges represent operations performed by the model intrinsic machinery (simplifications, merges, conversions) and, as such, do not presuppose adding of information, i.e. processes. Then, *flow density* gives the number of processes involved in the flow. When the flow length differs significantly from the flow density it means that a great part of operations are performed by the model.

The flow computation algorithm in FMH produces all flows that link the start and destination nodes in the hyper-graph, not just the shortest path. The differences among processing flows stay not just in the number of processing steps involved but in a larger set of factors, some reflecting technical aspects and some reflecting the users' personal preferences. All usable processing flows are offered to the user, together with measurable parameters and other available data. Going further in the particularisation of flows and as we will see further in the examples of the following two sections, the flows themselves are not sufficient to define the intended processing. In many cases, the processing edges have associated more pipeline modules (Figure 2), among which only one has to be chosen and sometimes complex conditions need to be verified by the input and output files. The conditions can be expressed as compatibility restrictions of the input file with the start node, of the destination node with the task specifications, as language restrictions, momentary availability of web services, costs, software and hardware configuration constraints, users' access rights, cost constraints, etc. The verification of all these conditions and the selection of proper modules on pipeline and conversion edges should be done before a flow is actually executed and is realised in

the *instantiation* of the flow. More instantiations of a flow are possible. The conditions checked for each module are different than those encumbered by input/output formats, which are intrinsically validated by the flow.

The intention in displaying the examples in the following sections is to show that common multilingual scenarios as well as complex applications can be modelled in FMH.

4. Performing part of speech tagging

Part of speech tagging (further referred to as POS tagging) is a common pre-processing step for most linguistic workflows. There are multiple tools available for this task, many of them being capable to change the language they work for when fuelled by a language specific resource (usually a language model trained on gold corpora). The significance of the nodes in the FMH graph of Figure 4 is not important for the purpose of this example, and we will ignore their description here. All we should know is the start and destination nodes and these are: TXT, corresponding to the original text including no annotations, and respectively POS, which should be understood as including only POS information.

We have not marked the processing edges of Figure 4 with actual names of processing modules. Rather, we have noted the modules' language compatibility constrains: EN_i for English, RO_j for Romanian and LIT_k designating Language Independent Tools, therefore tools compatible with any language.

One way to look at FMH is as a methodology of processing which augments and, at times, simplifies the annotation added to a text along a processing chain, thus advancing the representation towards the envisioned output. As it appears on Figure 4, there are multiple flows (paths) between the input node (TXT) and the output node (POS), differing both in the number of steps and in language restrictions.

In the multilingual use-case we will first describe two linguistic workflows that have the same pair of nodes start-destination, but

displaying different paths, as put in evidence by different language specific tools. This leads to different number of processing steps for the two workflows. Figure 5 shows the hierarchy of Figure 4, in which a language filter, specifying as processing language Romanian, has been applied.

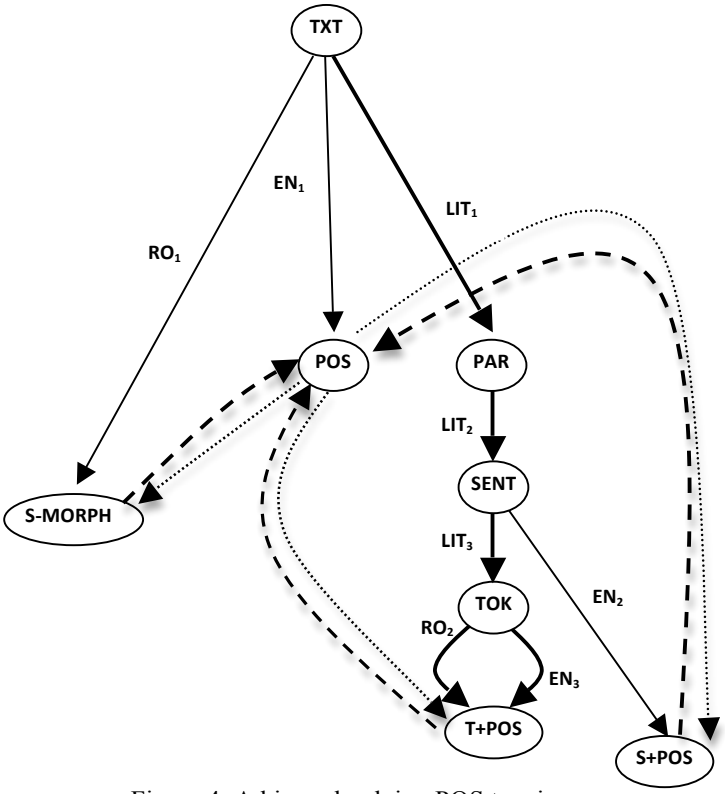

Figure 4: A hierarchy doing POS tagging

Two alternative flows bring an input document in Romanian from TXT to POS, the traversed nodes being:

(TXT, PAR, SENT, TOK, T+POS, POS)
and (TXT, S-MORPH, POS).

The instantiated processing flows are:

1. TXT>$_{LIT1}$PAR>$_{LIT2}$SENT>$_{LIT3}$TOK>$_{RO2}$T+POS<POS
2. TXT>$_{RO1}$S-MORPH<POS

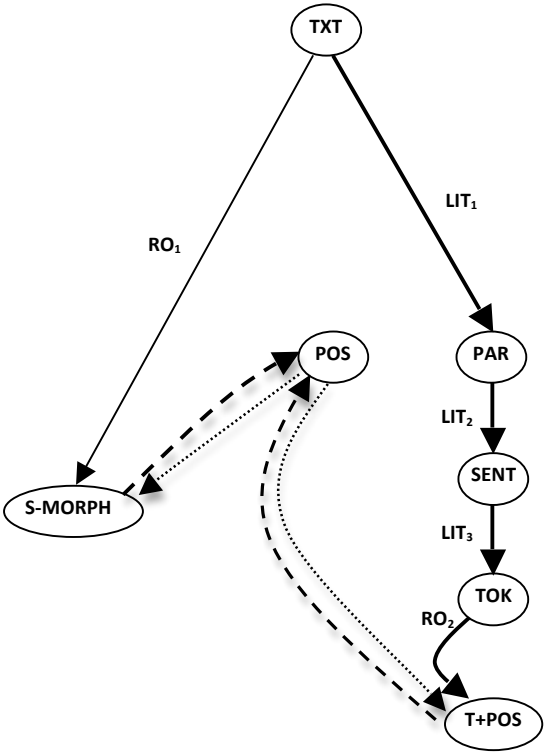

Figure 5: The workflow doing POS tagging for Romanian

Figure 6 shows three alternative flows bringing an input document in English from TXT to POS. The traversed nodes are:

(TXT, PAR, SENT, TOK, T+POS, POS)
(TXT, PAR, SENT, S+POS, POS)
and (TXT, POS)

The instantiated processing flows are:

1. TXT>$_{LIT1}$PAR>$_{LIT2}$SENT>$_{LIT3}$TOK>$_{EN3}$T+POS<POS
2. TXT>$_{LIT1}$PAR>$_{LIT2}$SENT>$_{EN2}$S-POS<POS
3. TXT>$_{EN1}$POS

The examples above put in evidence also an interesting situation found in multilingual applications: parts of workflows can look identical (for two different languages) in terms of processing steps. Compare, for instance the first resulted instantiations of each of the Romanian and English cases above: the first 3 steps (noted here $>_{LIT1}$, $>_{LIT2}$ and $>_{LIT3}$) are apparently identical. However, they imply processors which may differ in the language specific resources employed.

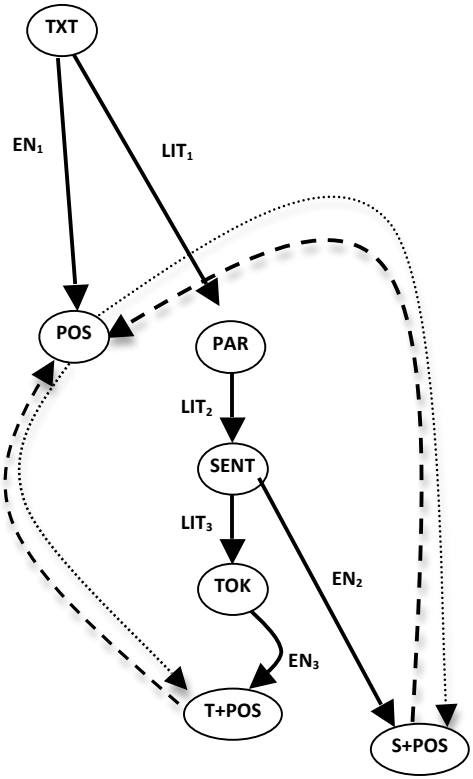

Figure 6: The workflow doing POS tagging for English

This example shows that in the same relatively compact hierarchy, the model can describe multiple alternative workflows, indicating the number of steps and the tools to be used in which step.

5. A Semantic Role Labelling application

The automatic labelling of semantic roles is one of the most complex examples of linguistic workflows, as the few systems available include a range of processing steps, from lexical level analysis (tokenization, POS tagging, etc.) to semantic analysis (usually word sense disambiguation, alignment with ontologies or other resources). The following example is inspired by the work described in (Trandabăţ, 2010; Trandabăţ, 2011) to mark semantic roles on a Romanian corpus. The author has used an aligned bilingual (English-Romanian) corpus and the semantic roles marked on the English version, from which the missing SRL markups have been imported onto the Romanian texts.

Some parts of the flow which were considered not relevant in the context of this example, as for instance the English semantic role labelling system, were left hidden in the example shown in Figure 7.

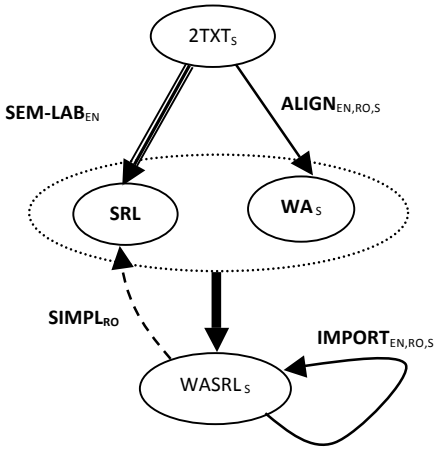

Figure 7: Importing semantic role labels

The nodes in Figure 7 have the following significance:

- 2TXT$_S$: contains two versions of the same text, therefore the two parts should be parallel translations, i.e. represent the same content (noted here with S). Markings that make explicit the two languages are supposed to surround the respective parts.

- SRL: the text that include semantic role markups. Suppose these are SR elements surrounding constituents of sentences;
- WA_S: schema containing word-aligning markups between two texts representing the same content S;
- $WASRL_S$: schema containing word-alignment and semantic role markups. It is not compulsory that all sentences contain SR markups.

We will explain now the edges:

- $SEM\text{-}LAB_{EN}$ signifies an unrevealed (perhaps long) processing chain which accomplishes semantic roles labelling on English texts. The language EN is an instantiation condition, therefore only sentences marked with the attribute-value pair language="EN" will be processed.
- $ALIGN_{EN,RO,S}$ is an edge performing a pipeline operation: the alignment of the sentences belonging to the two languages at word level. Again, the indexes, mark instantiation conditions: that the two languages should be EN and RO and the texts have the same content, S.
- $IMPORT_{EN,RO,S}$ represents the module importing the semantic roles from one part (EN) onto the parallel part (RO), provided the two parts represent the same content (S). The module presupposes to find SR markups on all language="EN" sentences.
- $SIMPL_{RO}$ is a simplifying operation that prunes off all markings except the SR elements of the sentences marked with language="RO".

As can be seen in Figure 7, there are three flows in the FMH of Figure 7 that link the mentioned pair of nodes:

1. $2TXT_S >_{SEM\text{-}LAB\_EN} SRL$

2. $((2TXT_S >_{SEM\text{-}LAB\_EN} SRL), (2TXT_S >_{ALIGN\_EN,RO,S} WA)) > WASRL <_{SIMPL\_RO} SRL$

3. $((2TXT_S >_{SEM\text{-}LAB\_EN} SRL), (TXT_S >_{ALING\_EN,RO,S} WA)) > WASRL >_{IMPORT\_EN,RO,S} WASRL <_{SIMPL\_RO} SRL$

Our task of marking semantic roles on RO texts is transposed in the model as the following requirement: find an instantiation of a

flow linking the start node $2TXT_S$ to the destination node SRL, such that given a parallel EN-RO document in the input to obtain the RO sentences annotated with SR elements. As seen, the instantiation should announce conditions on both the input and output files, as well as conditions to be verified by all processing steps. Suppose the input condition is satisfied for all three flows because a document containing a parallel EN-RO translation of a content S is presented to the start node $2TXT_S$.

Edge instantiation conditions for the first flow verifies the restrictions of the pipeline edge $>_{SEM\text{-}LAB\_EN}$, namely that sentences are marked with the attribute-value pair language=”EN”, which is true. Finally, the destination conditions are verified, namely that sentences marked with language = “RO” include also SR markings on them, which is false. This makes the first flow to be filtered out. The second flow evaluates to true the condition of $ALIGN_{EN,RO,S}$, that the two languages are EN and RO and the texts have the same content. However, after the merge, the special simplification edge $<_{SIMPL\_RO}$ will prune out all markings, resulting the null file, because no language=”RO” exists which include also SR markups. As such, the output condition is not fulfilled and the second flow fails too. Finally, the third flow includes one more condition induced by $IMPORT_{EN,RO,S}$, namely that all sentences marked language=”EN” include also SR markups, and this is verified. The $IMPORT_{EN,RO,S}$ pipeline produces sentences marked language=”RO” which include also SR markups. The $<_{SIMPL\_RO}$ will leave only them, and the destination condition is fulfilled.

This example shows one way in which the model can face an important feature of processing multilingual documents, the ability to deal with parallel corpora.

6. Conclusions

Many efforts in the current NLP research are concentrated to develop and adapt linguistic tools and resources that have an increased visibility and usability, and to help humanities and social sciences researchers to deal with the NLP technology. Both areas

require multilingual considerations, and this aspect is the main topic of this paper.

We have shown a model that needs two steps from the definition of a NLP problem until the preparation of the actual run: flow computation and instantiation. At times, the computed flows can be virtually the same irrespective of the language requirements and only the instantiation differentiate the specific behaviours. This is in line with the trend in modern NLP to separate algorithms from linguistic details. 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, which are powered by specific language models (frequency of n-grams). A syntactic parser can 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.

The Formats and Modules Hierarchy model supports a number of important features that characterise NLP processing. We list below some of the most significant properties of it:

- It allows a unified representation of both annotation formats and processing tools. An FMH hyper-graph functions as a framework for recording processing tools, based on which workflows can be designed and visualised.
- Linguistic resources can be clearly positioned within the hierarchy of formats, as the annotation schemas they observe.
- A hyper-graph can be shared by a community of researchers and is enriched any time a user “uploads” an annotated document/resource or a processing tool.
- Given a pair of input-output formats (called start-destination nodes in the hierarchy), the model proposes a set of processing workflows, by computing paths linking them, as combinations of linguistic processors, simplifications, conversions and merges.
- By instantiation, a set of flows can be reduced, possible down to one solution, while also fixing parameters of the processing components.

Moreover, when a multilingual behaviour is at value, the model can be characterised by the following set of features:

- Identical processing for different languages: in the model, two or more languages can share the same processing chain. However, the component modules may be instantiated differently by the language specific resources they require.
- Identical input-output schemas to accomplish the same task for different languages: in the model, two or more linguistic workflows can share the start-destination nodes pair, but include different paths, therefore offering distinct solutions. It is possible that during the instantiation of these flows, for instance induced by the specification of languages, to be revealed that the ambiguity in solution disappears for each of the involved languages, because the language constrains prune off specific parts of the flows.
- Processing of multi-language documents: as revealed by the last example in section 5, documents that include passages in different languages can be object to distinct processing.
- Snapshots of available processing power for particular languages: on a FMH hyper-graph which include tools for many languages, language filters can be applied to sieve only tools compatible with the selected languages. At times, this can produce disconnected hyper-graphs (nodes which cannot be reached), meaning that no solution exists.

We believe that the main benefit that our model brings to the multilingual NLP is an abstract and consistent way of looking at language dependency issues, that allows both generalisations and particularisations of approaches. However, a lot of research is still to be done. A revised deployment of the model is our next objective, as well as the implementation in the model of a large collection of NLP tools. The most important critics that someone, in our opinion, can raise with respect to the model is related to the intrinsic presupposed compatibility of processing tools, that the model considers given. Indeed, its success relies heavily on a concerted agreement with respect to the annotation conventions of linguistic phenomena. Otherwise the critical mass of tools necessary to build a sufficiently large hierarchy will not be reached. But, as it is known, although there is a huge need of standardisation of annotation schemas, standards still lack and there is sufficient

evidence to believe that they will not appear too soon. Moreover, because the domain is so active, which makes it advance in such a great speed, the research will always go ahead of any attempt of standardisation, and therefore, there will always exist tools employing new formats, which have not been standardised.

This is why, it is foreseeable that a line of research aiming to infer the semantic content of linguistic annotations should soon be opened. Copying human skills to recognise the significance of markings in a file from the first glance, such software will make possible not only the automatic classification of linguistic resources conforming to a general hierarchy, but also the automatic generation of convertors, in order to accomplish interoperability there where necessary.

Acknowledgments

This research has been partly supported by the ICT PSP projects METANET4U (under the grant no. 270893) and ATLAS (under the grant no. 250467).

References

1. Branco A., Trancoso I., Ananiadou S., Thompson P., McNaught J., Cristea D., Tufis D., Rosner M., Moreno A., Bel N. (2011): Specification of pilot services and applications, Document METANET4U-2011-D2.2, EC CIP project #270893, to appear on http://metanet4u.eu/
2. Cristea, D., Butnariu, C. (2004): Hierarchical XML representation for heavily annotated corpora. In *Proceedings of the LREC 2004 Workshop on XML-Based Richly Annotated Corpora*, Lisbon.
3. Cristea D., Forăscu C., Pistol I.C. (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, Lisbon, ISBN: 972-8865-50-3.
4. Cristea, D., Pistol, I. (2008): Managing Language Resources and Tools Using a Hierarchy of Annotation Schemas. Proceedings of the Workshop on Sustainability of Language Resources, LREC-2008, Marakesh.
5. Cunningham H., Maynard D., Bontcheva K., Tablan V. (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.
6. Ferrucci D. and Lally A. (2004): UIMA: an architectural approach to unstructured information processing in the corporate research environment, *Natural Language Engineering* 10, No. 3-4, 327-348.
7. Gazdar, G, Mellish, C. (1989): Natural Language Processing in Lisp, Addison-Wesley.
8. Pistol I.C, Cristea D. (2009): Managing Metadata Variability within a Hierarchy of Annotation Schemas, in *Proceedings of the 6th International Workshop on Natural Language Processing and Cognitive Science - NLPCS 2009*, Milan - May 2009, pp. 111-116, ISBN: 978-989-8111-92-0.
9. Pistol I.C. (2011): The Automated Processing of Natural Language, PhD thesis, "Alexandru Ioan Cuza" University of Iaşi.
10. Trandabăţ D. (2010): Natural Language Processing Using Semantic Frames, PhD Thesis, University Al. I. Cuza Iasi, available at: http://students.info.uaic.ro/~dtrandabat/thesis.pdf.
11. Trandabăţ D. (2011) Mining Romanian texts for semantic knowledge, in *Proceedings of Intelligent Systems and Design Application Conference*, ISDA2011, Cordoba, Spain.