Extending a Tool Resource Framework with U-Compare

Michael Rosner*, Andrew Attard*, Paul Thompson, Albert Gatt*, Sophia Ananiadou****

*University of Malta, Dept ICS
Msida MSD2080, Malta
{mike.rosner, andrew.attard, albert.gatt}@um.edu.mt
** School of Computer Science
University of Manchester, Oxford Road
Manchester M13 9PLUK
{paul.thompson, sophia.ananiadou}@manchester.ac.uk

Abstract

This paper deals with the issue of two-way traffic between on the one hand, language resources that have been conceived from a local perspective, i.e. from within a local project or institution, and on the other, a shared framework conceived from a global perspective that supplies such resources for local re-use or enhancement. We believe that a key enabler to such traffic is the choice of an appropriate sharing platform, and here we illustrate the point with respect to a constellation of EU projects that is attempting to enhance the quality and scope of shared resources, and a local project that has some already-developed local functionality. The paper first introduces the underlying projects, then goes on to discuss the proposed platform (U-Compare) whose use is then illustrated for a small module developed for a local project.

Keywords: language resources, sharing, sharing platform

1. Introduction

Access to suitable Language Resources (LRs) is a *sine qua non* for the development of Language Technology. But LRs of the right kind do not always occur naturally and frequently require LTs for their creation. The preparation of an interestingly-large POS-tagged corpus, for example, requires an accurate POS-tagger, unless we happen to have an army of human specialists on tap, which we assume is not the case most of the time. So one cannot consider LRs in isolation from the LTs used to create them. In fact the picture also includes a collection of users from different sectors including academia and industry who may themselves contribute content, as shown in Fig. 1.

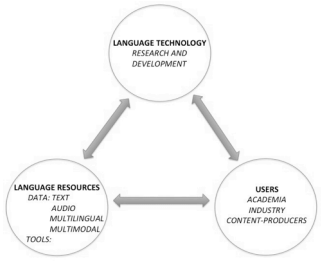

Fig. 1: Resources, Users and Technologies

This dynamic symbiosis between Resources, Users and Technologies is reflected in the philosophy of the Multilingual Europe Technology Alliance (META), which forms the backdrop to the work described here. In this paper we are focussed on the provision of *tools* as a subclass of LRs spanning a range of functionalities including automatic annotation, parsing, statistical analysis etc. The notion of resources-as-tools is not new, having first been proposed under the name BLARK (Basic Language Resource Kit) by Krauwer (2003). However, it has taken on a renewed importance as efforts towards the development of language technologies become ever more globalised.

One of the main problems when dealing with tool resources is how to guarantee interoperability, both with other tools, and with the data that these tools consume and produce. As we shall explain below, one approach to dealing with this problem is to ensure that all tools operate within a single type system which defines the type of all static data, and the input/output types (or *signatures*) of all the tools.

Such a type system is fine if we are starting from scratch, since everything can be designed to operate within it, but typically this is not the case. A more usual scenario is that we have a mix of tools and data that have been developed at different times, for different projects, in various states of readiness. We might, for example, have a perfectly good parser developed for project A, but in order to use it the grammar data which was developed for project B has to be in a particular format. In short, if we solve the interoperability problem by creating a type-based framework, we are then faced with another problem: how to deal with legacy resources and tools.

This paper concerns just such a problem. Roughly, we have a collection of projects aimed at creating a shared framework for the distribution of language resources in general, and we have an under-funded local project which has been running for some years which has nevertheless developed some tool resources for Maltese. Now the question is how to upgrade those resources and incorporate them into the shared framework.

The structure of the paper is as follows: section 2 describes the background projects from which the work has been conducted. Section 3 mentions MLRS, the local project that has developed its own functionality. Section 4 is devoted to the proposed platform (UIMA and U-Compare), and section 5 shows how the local functionality can be integrated into the platform. The discussion in section 6 assesses the possible impact of proposed framework, and section 7 concludes.

2. The Project Constellation

The work reported in this paper takes place within a constellation of EU projects that together are contributing towards the realisation of META. The mission of the Alliance is to further language technologies as a means towards facilitating communication and cooperation across language barriers. The constellation comprises three language-resource-oriented projects: METANET4U, CESAR, META-NORD, and a network of excellence, META-NET, to which they all relate and which to some extent ties them all together.

2.1 META-NET

META-NET is an EU-funded Network of Excellence that supports META through three main objectives aimed at (i) building a community with a shared vision, (ii) building an open resource-sharing framework called META-SHARE, and (iii) building bridges to neighbouring technology fields. The work described here has a direct bearing on the second of these objectives.

2.2 The Language Resource Projects

The three projects are similar in so far as they seek to contribute to META-SHARE by targeting the upgrading, extension, linking, and distribution of language resources. They are also organised along more or less the same lines, as elaborated below. They differ primarily in the language groups handled by each. Broadly, METANORD handles the Nordic Languages and CESAR, the languages of Eastern Europe. METANET4U, with which the present paper is mainly concerned, deals with Spanish, Portuguese, Maltese, English and Romanian.

2.3 METANET4U

The work in METANET4U is split into a number of tasks, each of which is distributed amongst the partner Universities. The main responsibilities include:

(i) Analysis and Selection of Language Resources
(ii) Enhancing Language Resources
(iii) Cross National Collaboration and Pilot Service
(iv) Outreach, Awareness and Sustainability

In this paper, we are mainly concerned with (ii) and (iii). Although most of the language resources that have been selected for uploading to META-SHARE are useable in some sense, there are various shortcomings that have to be addressed.

For data-oriented resources, these include: cleaning datasets, removal of inconsistencies, particularly where this can be done automatically; ensuring that the data complies with, or can be mapped to, a form that complies with existing standards for character representation, annotation using compliant tagsets etc.; improving descriptive documentation, both informal (text description) and metadata, developed according to a standard developed for META-SHARE. Tool resources present a different set of problems. Obviously, there are operational bugs to be removed, but the most important area of improvement concerns interoperability.

3. Maltese Language Resource Server (MLRS)

The MLRS project (Rosner *et al.*, 2008) was initiated in 1997, with the twin goals of creating a corpus for the Maltese language, together with an associated machine-readable lexicon. The project, which is ongoing, has been intermittently supported by funding from different sources[1]. The current version of the system, which was released online[2] (Gatt and Borg, 2011) in May 2011, is centred around a text corpus of about 100 million words, represented in a standard format and implemented using the IMS Open Corpus Workbench (Hardie, forthcoming). This provides for the definition of certain corpus-related services which target Maltese linguistic research in the first instance, and includes certain preprocessing steps, which facilitate the translation of different input formats into the standard representation, word frequency calculations, and concordancing. POS tagging is currently under development. The medium term aim here is to gradually ascend the "semantic food chain" and to include chunking, named-entity recognition, text classification etc.

Presently, MLRS can be regarded as a tool with a fairly high level of "macro" functionality that presents itself to the user as a query engine over corpora that can be used mainly for linguistic research. However, this macrofunctionality is constructed from a number of "microfunctionalities" - components that implement, for example, tokenisation, sentence and paragraph splitting, source document format translation, stop word removal, POS tagging (eventually). The main problem, from the perspective of META-SHARE, is that the interface does not expose these microfunctionalities and so they are not accessible individually.

The challenge we address is how to remedy this situation - i.e. to retain the macro functionality whilst in addition allowing the micro functionalities to be exposed and reused within META-SHARE.

4. UIMA and U-COMPARE

Architecturally, the solution to this problem is a framework that enables microfunctionalities to be composed systematically.

In effect, such a framework has already been developed for building powerful analysis techniques such as information retrieval, information extraction, textual inference, automated reasoning for "unstructured information", i.e. data lacking an accepted data model which does not obviously fit into relational tables. Unstructured Information Management Architecture[3] (UIMA) is an architecture and software framework which facilitates the integration of arbitrary components to work in collaboration within the same application. This section

[1] University of Malta and Malta Council for Science and Technology

[2] http://mlrs.research.um.edu.mt/

[3] Apache UIMA is an Apache-licensed open source implementation of the UIMA specification, which is being developed concurrently by a technical committee within OASIS (http://www.oasis-open.org/committees/uima/).

gives an overview of the UIMA framework, and following that an overview of U-Compare.

4.1 UIMA

UIMA makes it possible to put together a workflow of different components, each of which would be responsible for carrying out a specific type of analysis over a document (text file, audio or video). Such components are referred to as *Analysis Engines* (AEs). AEs can be either primitive or aggregate. In the former case, the AE would be hosting one annotator, whereas aggregate AEs are defined to contain other AEs within themselves.

The annotator itself is the constituent inside the AE which contains the analysis algorithm. The annotator's role is to annotate regions within the document and to create *annotation objects* of a particular type. For example, a commonly used pre-defined UIMA type is Annotation. This type is used to label regions within the document, specifying the beginning and end position of the region. These offsets are stored in the type's features. Features can be seen as properties associated with a particular type. Thus, in the case of the annotation type, two of its features are *begin* and *end* offsets. UIMA has a set of such pre-defined basic types, and gives the developer the possibility to extend these for a richer Type System. Hence, AEs search within the document for the types of objects defined by the respective assigned type system. An important feature within the UIMA architecture is the Common Analysis Structure (CAS). Throughout the execution of the workflow, all the generated annotations are recorded and shared amongst the other AEs through the use of the CAS. So when we talk about searching the document, we actually mean searching within the CAS for objects that collectively hold all information about the document in the form of annotations.

It is easy to see how tools can be made to work together given such a setup. In order to create a workflow in which, for example, two tools are chained together so that the output of one is compatible with the input of the other. This is achieved by ensuring that input and output expectations of the tools are observed with respect to the CAS.

4.2 U-COMPARE

UIMA is a domain-independent framework. In order to apply it to a specific domain, users need to create a type system defining the data types used by tool resources. Based on the UIMA framework, U-Compare[4] defines just such type system that is specifically oriented towards the domain of text mining. A U-Compare compatible component is a UIMA component which makes use of the U-Compare type system, or extends its types, in order to read objects from the CAS and to record and share results.

[4] U-Compare is a joint project between the University of Tokyo, the Center for Computational Pharmacology (CCP) at the University of Colorado Health Science Center, and the National Centre for Text Mining (NaCTem) at the University of Manchester.

U-Compare provides an integrated platform, currently containing an extensive repository of ready-to-use natural language processing components, all of which operate within its compatible type system. Furthermore, through an intuitive graphical user interface, the user is able to construct, edit and compare the performance of different workflows (Kano *et al*., 2009) - as suggested by its name.

Because U-Compare includes a considerable level of language-handling functionality (and in fact currently holds the largest single repository of type-compatible UIMA components), it has been proposed by METANET4U as a promising starting point for the creation and distribution of LT tools within the META-SHARE framework (Ananiadou et al., 2011). The proposal is not without challenges, however. The U-Compare type system has been designed for the text mining domain. However, the demands of META-SHARE are considerably more general, since the set of resource types includes not just written resources, but also audio, video and multimodal modalities as well as more structured resources like lexicons and grammars.

Therefore, a major issue for investigation is the extent to which the existing type system is the U-Compare type system adequate. In this paper we report on a first experiment that incorporate a small piece of the micro-functionality of MLRS as a test case.

5. Integration

5.1 Converting Modules to UIMA Components

MLRS, mentioned in section 3, hosts an evolving set of textual resources and natural language processing services for Maltese. As an example, it currently includes a tokeniser which has been specifically designed to handle the peculiarities of Maltese tokenisation. Amongst these we count, for example, the use of the hyphen, apostrophe, clitic pronouns etc.

If METANET4U is to export a Maltese tokeniser to META-SHARE, there would be considerable advantages in making it U-Compare compliant in order to exploit the advantages such as incorporation into workflows, generation of statistics, comparison of performance etc.

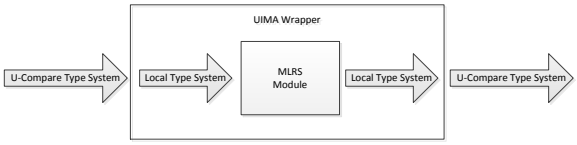

Fig. 2: UIMA Wrapper

In order to make the tokeniser compliant with U-Compare, one can consider two options: either re-implement the components or create a UIMA wrapper, i.e. write code to map the U-Compare types to the local types used by MLRS for input, and then vice versa for outputting tokenisation results. In this case and in general, reimplementation is undesirable, so we need to look at the UIMA wrapper solution. This is depicted in Fig. 2.

The MLRS tokeniser being considered makes use of a sentence splitter for Maltese and then tokenises each sentence. The existing tokeniser reads the data to be

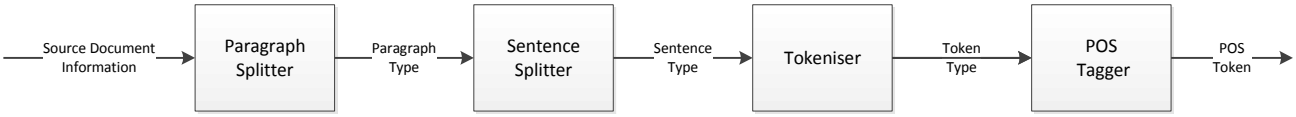

Fig. 3: POS Tagger Workflow

tokenised from a file, executes the tokenisation task, and writes the result to an XML file.

The new component (the outer box in Fig. 2) will need to be modified with regards to its input and output requirements; but the main computation of the component will remain the same. Hence, instead of reading the input from a file, the component will now retrieve it from the CAS, and instead of writing the results to a file, the component will now write the results to the CAS again. This will enable the components executing later in the workflow to use these results by reading them from the CAS.

As explained in *The Apache UIMA Development Community* (2010) the steps which need to be completed are:

1. Define the CAS types the annotator will use
2. Generate the Java classes for these types
3. Write the actual annotator code
4. Create the AE descriptor

Since our component needs to be U-Compare compatible, steps 1 and 2 are already done. The annotator will be making use of the U-Compare type system, and the corresponding Java classes are already implemented.

Step 3 is where the type conversion comes in. Assuming that a tokeniser instance would take as input a string containing the whole document, this method would (i) retrieve the document from the CAS, and store it in a string variable, (ii) create an instance of the tokeniser and (iii) pass the string as argument.

Since the document object obtained from the CAS is now converted to a string – the type which the tokeniser was already using for computation – now the execution could take place normally. Upon completion of the tokenisation, for each token identified: (i) a U-Compare token object is instantiated (ii) *begin* and *end* offsets of the token are set, corresponding to its position in the document and (iii) the token object is added to the CAS.

Step 4 involves writing the AE descriptor. This is an XML file which specifies the component's properties and requirements, and which could be easily completed using UIMA pluggable tools for Eclipse[5].

Here we specify (i) that the AE is subscribed to the U-Compare Type System, (ii) the location of the tokeniser (iii) its input and output capabilities, i.e. the input would be left empty, since the component would retrieve the whole document, rather than some specific types and the output would correspond to U-Compare token object, hence it would be of type *Token*.

We have carried out these steps successfully for the MLRS tokeniser, which means that, in principle, we have shown how the MLRS tokeniser can be made available within U-Compare without any change to the U-Compare type system. Furthermore, if the UIMA framework is accepted within META-SHARE, we have also demonstrated a basis for extending the functionality of the latter.

[5] A Java Integrated Development Environment (http://www.eclipse.org/)

6. Discussion

In order to assess where we go from here, we need to take a step back. We have to bear in mind that MLRS has been conceived locally, and that local functionality has been developed, as exemplified by the tokeniser. This functionality is now in principle available to META-SHARE i.e. to anybody wishing to tokenise documents written in Maltese.

We envisage two-way traffic between modules developed locally and those available through META-SHARE. Hence, where appropriate, not only should the results of local developments be made available to META-SHARE, as we have seen with the tokeniser, but also, where appropriate, components available through META-SHARE should be freely available for the development of local functionality.

MLRS currently includes a POS tagger, but there are two major shortcomings: (i) it is not very accurate and (ii) the architecture is monolithic. We envisage that the first problem will be solved by providing better training data, and an effort to address this is currently under way. We are proposing to address the second problem using the U-Compare framework involving a UIMA workflow.

Fig. 3 shows a possible UIMA workflow, consisting of four components, for implementing such a POS tagger. The labels on the arrows name the corresponding U-Compare types. Source Document Information (the first input) is the document which is given as input to be analysed. The components' outputs are all U-Compare types.

The advantage of such an arrangement is that the each individual component is reusable and exportable. The realisation of such a workflow could be achieved in a number of different ways.

- The individual modules could be fashioned from already existing MLRS components. For instance, the underlying functionality of the box marked POS tagger already exists. However, it needs to be wrapped in the manner described in Fig. 2.
- The individual modules could be imported from META-SHARE. In this case the drag and drop interface available within U-Compare can be used to insert the respective processing element into the workflow

Alternatively some mixture of these two approaches might be used to develop a new, U-Compare-compatible processing element involving an interaction between existing processing elements that cannot be easily constructed in terms of a workflow.

An example of this might be a Named Entity Tagger for Maltese (which in fact we are planning to develop in future). One could imagine that the construction of such a tagger might not involve just a pipeline of existing processes - but more complex interactions between , say, an (existing) POS tagger, a locally produced gazetteer linking names and to entities, and a (shared) semantic classifier for distinguishing between locations and organisations using sentence context. In such a case, it could prove most efficient to simply program the interaction in terms of the underlying platform and create a new module for subsequent export to META-SHARE.

Before concluding, as already pointed out, ongoing work is currently focusing at increasing the number of Maltese tool resources. Since the study carried out in this paper revolves around *available* resources, this conversion procedure was necessarily carried out on a fairly simple component. However, other more sophisticated components have been successfully wrapped up to be U-Compare compliant, including corpus readers, as well as semantic and syntactic tools[6], and we propose to adopt this methodology to other components for Maltese as and when they are developed.

7. Conclusion

We have presented a project cluster which aims to improve LT across a wide range of languages by developing, amongst other things, a framework for sharing language resources and LT tools. We have illustrated how U-Compare can be used to elevate the status of an existing module from something strictly internal to something that can be shared by the community at large. Conversely, we have suggested that in future, the adoption of a framework like U-Compare can be used to facilitate the creation of new functionality by adopting a mix and match approach which freely draws from shared and locally produced modules, thus contributing to the shareable resources.

8. Acknowledgment

The work described above has been carried out under the auspices of the METANET4U project jointly funded by the EU (grant number 270893) and the Universities of Malta and Manchester.

References

Ananiadou, S., Thompson, P., Kano, Y., McNaught, J., Attwood, T. K., Day, P. J. R., Keane, J., Jackson, D. and Pettifer, S. (2011). Towards Interoperability of European Language Resources. *Ariadne*, 67.

Ferrucci D, Lally A, Gruhl D, Epstein E, Schor M, Murdock JW, Frenkiel A, Brown EW, Hampp T, Doganata Y., "Towards an Interoperability Standard for Text and Multi-Modal Analytics". *IBM Research Report* RC24122 2006

Gatt, A. and Borg, C., Using the MLRS Interface, Institute of Linguistics, University of Malta, 2011. Available at http://mlrs.research.um.edu.mt/corpusquery/cwb/doc/mlrs_userdoc.pdf

Hardie, A. (forthcoming). "CQPweb - combining power, flexibility and usability in a corpus analysis tool". http://www.lancs.ac.uk/staff/hardiea/cqpweb-paper.pdf (draft)

Kano Y., McCrohon L., Ananiadou S., Tsujii J. (2009). Integrated NLP Evaluation System for Pluggable Evaluation with Extensive Interoperable Toolkit. In: *Proceedings of the NAACL HLT Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing, Boulder, Colorado, Association for Computational Linguistics, pp. 22-30.* Boulder, Colorado, Association for Computational Linguistics.

Kano, Yoshinobu, William A. Baumgartner Jr., Luke McCrohon, Sophia Ananiadou, K. Bretonnel Cohen, Lawrence Hunter and Jun'ichi Tsujii *U-Compare: share and compare text mining tools with UIMA*. Bioinformatics. 25(15), pp. 1997-1998, 2009; doi: 10.1093/bioinformatics/btp289

Krauwer, Steven, The Basic Language Resource Kit as the First Milestone for the Language Resources Roadmap, Survey Lecture delivered at SPECOM, October 2003, Moscow, (also available at http://www.elsnet.org)

Rosner, M., Caruana, J., and Fabri, R., Maltilex: A computational lexicon for maltese. In M. Rosner, editor, Computational Approaches to Semitic Languages: Proceedings of the Workshop held at COLING-ACL98, Université de Montréal, Canada, pages 97–105, 1998.

[6] Navigate to *http://u-compare.org/components/index.html* for the full list of U-Compare components