Interoperability of corpus processing workflow engines: the case of. AlvisNLP/ML in OpenMinTeD

Mouhamadou Ba, Robert Bossy

▶ **To cite this version:**

Mouhamadou Ba, Robert Bossy. Interoperability of corpus processing workflow engines: the case of. AlvisNLP/ML in OpenMinTeD. Meeting of working Group Medicago sativa, May 2016, Portoroz, Slovenia. hal-01455853

HAL Id: hal-01455853
https://hal.science/hal-01455853v1

Submitted on 5 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Interoperability of corpus processing workflow engines: the case of AlvisNLP/ML in OpenMinTeD

Mouhamadou Ba, Robert Bossy
INRA – MaIAGE – Bibliome group
Domaine de Vilvert, 78352 Jouy-en-Josas, France
Mouhamadou.Ba@jouy.inra.fr, Robert.Bossy@jouy.inra.fr

Abstract

AlvisNLP/ML is a corpus processing engine developed by the Bibliome group. It has been used in several experiments and end-user applications. We describe its design principles and data and workflow models, then we discuss interoperability challenges in the context of the OpenMinTeD project. The objective of OpenMinTeD (EC/H2020) is to create an infrastructure for Text and Data Mining (TDM) of scientific and scholarly publications. In order to offer to the infrastructure users a single entry point and the widest range of tools as possible, the major European corpus processing engines will be made interoperable, including Argo, DKPro, and GATE. We show that AlvisNLP/ML can be fully integrated into the OpenMinTeD platform while maintaining its originality.

Keywords: Natural Language Processing, Processing Workflows, Software Interoperability

1. Introduction

AlvisNLP/ML is a corpus processing engine developed by the Bibliome group. It automates sequences of NLP and machine learning steps. AlvisNLP/ML is a critical software for conducting experiments in natural language processing, information extraction, and information retrieval. Moreover AlvisNLP/ML plays a key role in the deployment of several end-user services like semantic search engines (Bossy et al., 2008), corpus-based database and ontology population (Nedellec et al., 2014; Golik et al., 2012), and also in the preparation of the BioNLP-ST challenges (Bossy et al., 2012; Bossy et al., 2015).

In this paper we present AlvisNLP/ML and its components, and we discuss the plan to make AlvisNLP/ML interoperable with similar frameworks in the context of the OpenMinTeD EC/H2020 project. The goal of OpenMinTeD is to "create an open, service-oriented infrastructure for text and data mining (TDM) of scientific and scholarly content" (OpenMinTeD Consortium, 2016). The main technical ambition of OpenMinTeD is to make several corpus processing engines interoperable in order to offer the widest range of tools to the OpenMinTeD platform users. The engines provided by the consortium members include AlvisNLP/ML, GATE (Cunningham et al., 2013), Argo/U-Compare (Rak et al., 2012; Kano et al., 2009), DKPro Core (Eckart de Castilho and Gurevych, 2014), LAPPS (Ide et al., 2014). All of them are either built on top of the UIMA framework (Ferrucci and Lally, 2004), or already provide an interoperability layer to UIMA, therefore we will assume that interoperability issues are addressed in the context of UIMA components.

Section 2 describes AlvisNLP/ML data and processing models. Section 3 presents our perspective for the interoperability of AlvisNLP/ML components, and discusses potential challenges.

2. Description of AlvisNLP/ML

The design principles of AlvisNLP/ML focus on genericity, modularity, and support of reproducibility and ease of use for NLP experimentation (Nédellec et al., 2009). The typical target user is a researcher with basic computer skills but not necessarily proficient in software programming, their NLP knowledge can be advanced to moderate. AlvisNLP/ML has been used by a wide range of academic users: NLP specialists, knowledge engineers, computer scientists, and bioinformaticians. One key problem in NLP experiments is reproducibility because results depend on a large number of stacked intermediate processing steps for each of which several parameters and external resources have an impact on the result. AlvisNLP/ML attempts to address reproducibility by requiring the user to specify a processing sequence, its parameters and resources within a single file using a common language. In this way the conducted experiments are fully transferable.

2.1. Processing Model

The processing model of AlvisNLP/ML relies on a sequential execution of individual modules. Each module offers a core functionality and several modules can be combined in sequences in order to build complex corpus extraction and mining tasks.

The coordination of modules is achieved using a shared data structure model that is able to represent the corpus contents and annotations. The data structure is passed from one module to the following, so that each module is able to access the corpus and the result of previous modules, and to append more annotations to the benefit of following modules. The AlvisNLP/ML processing model is thus similar to UIMA, where the shared data structure is analogous to UIMA's CAS.

2.2. Data Model

The AlvisNLP/ML data model is composed of 3 main components: the shared data structure, primitive modules and plans.

- The shared data structure contains both the corpus contents and annotations produced by different tools.

- Primitive modules are atomic tools for processing the data structure contents. Primitive modules include tokenizers, named entity recognizers, syntactic parsers, machine learning tools, corpus importers, annotation exporters, etc.

- Plans (workflows) are sequences of primitive modules coordinated in order to build complex corpus processing tasks.

2.2.1. Shared Data Structure

The shared data structure is responsible for holding the corpus contents and structure as well as the annotations generated by each primitive module. It is an fixed-depth tree whose successive levels represent the corpus, documents, sections, annotations and tuples. A section represents a passage of text in a document, an annotation represents a span of the text contents (words and named entities), and a tuple represents a labelled collection of nodes (dependencies, constituents, semantic relations). Each node is further characterized by a set of *features* which are key/value pairs (*e.g.* POS tag, lemma, dependency label, cross-reference).

The data structure does not define types of annotations or entities. Their interpretation as words or dependencies, for instance, is entirely up to the workflow designer. This allows for a greater flexibility and the user to experiment different strategies. AlvisNLP/ML shares this notational approach with the BioC project (Comeau et al., 2013).

Finally the transmission of information between the modules relies on conventions over feature names and tuple argument labels. The conventions are local to the plan however values are set by default in modules that perform traditional NLP tasks ("word", "sentence", "pos", etc.)

2.2.2. Primitive Modules

The primitive modules are the elementary tools present in AlvisNLP/ML. They are independent and non-decomposable. A module is composed of an algorithm and an interface. The algorithm is the actual implementation of the module. It defines the operational task the module have to full-fill. The interface defines the parameters supported by a module and the description of the module. The module parameters are used to specify external resources, to configure the module behaviour, and to induce the portions of the shared data structure on which a module read or write.

2.2.3. Plans

A plan specifies a sequence of modules to be executed in order, and the value of the parameters for each module. The parameters are set with two goals in mind: configure the modules according to one's needs, and coordinate the modules so that they create and access the relevant parts of the shared data structure.

Plans are expressed in XML that the AlvisNLP/ML engine interprets by instantiating the specified modules, converting the parameters, performing a static validation of the plan, and executing the module algorithms. A typical plan is generally composed of three parts: a first part reads initial data from external sources (reader modules), a second part performs the specific text processing, and a third part aggregates and presents the results in a suitable format (export modules).

Plans can be parametrized and composed into larger plans, thus allowing the user to define and share custom libraries of plans.

3. Integrating AlvisNLP/ML in OpenMinTeD

To integrate AlvisNLP/ML in the future OpenMinTeD platform, two points have to be taken into account: the module registry, and module interoperability.

3.1. Module registry

The OpenMinTed platform will offer a registry that exposes modules from all partners. This registry allows users to browse and look for modules that fit their specific needs.

AlvisNLP/ML features a primitive registry of modules; it's sole responsibility is to to provide module instances, their documentation, and their parameter set when executing a plan. The OpenMinTeD registry however must allow the exploration and the comparison in a large federated pool of modules. To achieve this, a uniform description of modules from all providers through a meta-data standard is necessary.

Thus one of the challenges for the integration of AlvisNLP/ML will be to align the description of its modules to the OpenMinTeD standard. Currently AlvisNLP/ML modules are described in two parts: a documentation file, and source code annotations. The source code annotations allows the system to manage automatically the module name, module parameters, data types, and default values. The documentation, completed through source code annotations, is designed for human consumption. It helps users to understand the purpose and the customization of modules. To fit the standard, the existing description model of AlvisNLP/ML must be extended with additional aspects like flow control, functional classification of modules, and licensing.

3.2. Module interoperability

Modules of each partner must be made interoperable in order to avoid component "silos" and offer the user the most of each system. The OpenMinTeD platform will compose workflows with modules from different providers uniformly. That raises compatibility issues at different levels, for example at data, protocol or licensing levels.

AlvisNLP/ML modules are mutually compatible since they share the same data model. Concerning the compatibility between AlvisNLP/ML modules and modules from other partners, we foresee three major challenges: the shared data structure, the engine, and the specification of module parameters.

3.2.1. Mapping of the shared data structure

The shared data structure of AlvisNLP/ML must be mapped to one or several type-systems of our partners' engine. Fortunately all annotations and their associated information in type-systems fit into one or several elements of the data structure.

AlvisNLP/ML and partner's type systems represent the same core information, though they differ in its representation. Core elements such as *corpus*, *documents*, *sections*, *annotations*, *dependencies* or *relations* are present in most partner's type-systems. Most discrepancies between type-systems are nomenclatural differences. For instance, a *sentence* in DKPro is called *annotation* in AlvisNLP/ML, *sentence* is in fact an annotation (a class of annotations). In other cases one element in one type-system benefit from a more detailed breakdown in another type-system. For instance LAPPS uses individual elements to represent the *Location*, *Organisation* and *Date* whereas AlvisNLP/ML represents everything as *annotations*. The concrete mapping between an entity of the AlvisNLP/ML data structure and a element of a partner's type system falls into a combination of basic operations such as renaming, selecting element components, or composing/decomposing elements. Thus, the integration can be achieved by specifying a back and forth transformation in such a way that the AlvisNLP/ML engine automatically injects and extracts data into/from the data structure. The particular elements (e.g., audio, video), from partner's type systems, that AlvisNLP/ML does not use, can be managed during the mapping process as byte streams that will remain unprocessed and handed back at the end of of the processing.

3.2.2. Encapsulating the engine

AlvisNLP/ML has its own engine, while most of the partners systems are enacted by the UIMA engine. It is likely OpenMinTeD platform will be operated on UIMA. Therefore in order to be exposed as a OpenMinTeD service, a module will have to embed a AlvisNLP/ML engine. This poses software architecture challenges that must be taken care of, especially regarding monitoring and usage of server resources (CPU and filesystem).

Alternative scenarios take advantage of workflow engines like Taverna (Wolstencroft et al., 2013) or Galaxy (Giardine et al., 2005). AlvisNLP/ML modules would have to be embedded with the engine in the same way. In the case of Taverna, components are assumed to be distant and components communicate through a data exchange protocol. Taverna sees the components as "black-boxes", the engine underlying a component is not constrained.

3.2.3. Parameters

AlvisNLP/ML module parameters are strongly typed. Currently there are more than fifty different parameter types. On one hand this further specifies the role of the parameter in the module, thus helping the user to configure their workflow. For instance a parameter of type *regular expression* instead of *string*, self-documents about the expected values and even its function with regard to the module behaviour.

On the other hand it hinders the integration in a system that assumes parameters are either scalars (integer, string, or boolean), or collections of scalars. Since the definition of parameters is strongly tied to the components implementation, it is very unlikely that the range of parameter types will change from one system to another.

Complex and alternate parameter types can always be exposed as *string* and automatically converted. This does not entail any development since AlvisNLP/ML provides converters for all parameter types. However we can take advantage of strong typing to automatically complete the component documentation, or to generate appropriate user interfaces for configuring components.

4. Conclusion

OpenMinTeD is an ambitious project that aims to offer Text and Data Mining services to a wide range of users. One of its critical milestones is the interoperability of several corpus processing workflow engines, including AlvisNLP/ML. We have established that despite differences in data models, component specifications, and implementation, there are enough common grounds between AlvisNLP/ML and our partner's systems. We showed that the integration of AlvisNLP/ML in the OpenMinTeD platform is a reasonable objective.

Most of the effort must be done in concertation with our partners in order to specify mappings between the shared data structure and different type-systems. Also we demonstrated the necessity to define collaboratively a common vocabulary to describe components and resources in order enable easy workflow composition.

Beyond automatic text processing adressed in this paper, the OpenMinTeD project also comprises curation, human validation and vizualisation aspects. These activities are supported by tools that assist users to explore and review data, and to build resources for further processing. Such tools include annotation editors –like Brat (Stenetorp et al., 2012) or AlvisAE (Papazian et al., 2012), and terminology or ontology editors –like OBO-Edit (Day-Richter et al., 2007) or TyDI (Nédellec et al., 2010). We believe that the interoperability effort should extend to user interfaces because it allows for a wider and more realistic range of applications, especially participative resource building and applications that require continuous update and processing. However they raise new challenges since these components operate in a different pace than automatic processing tools.

5. Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme (H2020-EINFRA-2014-2) under grant agreement No. 654021. It reflects only the author's views and the EU is not liable for any use that may be made of the information contained therein. The development of AlvisNLP/ML was funded by the Quaero project (OSEO). The remainder of the work and perspectives described in this paper are funded by OpenMinTeD.

6. Bibliographical References

Bossy, R., Kotoujansky, A., Aubin, S., and Nedellec, C. (2008). Close integration of ML and NLP tools in BioAlvis for semantic search in bacteriology. *Proceedings of the Workshop on Semantic Web Applications and Tools for Life Sciences, UK*.

Bossy, R., Jourde, J., Manine, A., Veber, P., Alphonse, É., van de Guchte, M., Bessières, P., and Nedellec, C. (2012). Bionlp shared task - the bacteria track. *BMC Bioinformatics*, 13(S-11):S3.

Bossy, R., Golik, W., Ratkovic, Z., Valsamou, D., Bessières, P., and Nédellec, C. (2015). Overview of the gene regulation network and the bacteria biotope tasks in BioNLP'13 shared task. *BMC Bioinformatics*, 16(10):1–16.

Comeau, D. C., Islamaj Dogan, R., Ciccarese, P., Cohen, K. B., Krallinger, M., Leitner, F., Lu, Z., Peng, Y., Rinaldi, F., Torii, M., Valencia, A., Verspoor, K., Wiegers, T. C., Wu, C. H., and Wilbur, W. J. (2013). BioC: a minimalist approach to interoperability for biomedical text processing. *Database*, 2013.

Cunningham, H., Tablan, V., Roberts, A., and Bontcheva, K. (2013). Getting more out of biomedical documents with GATE's full lifecycle open source text analytics. *PLoS Comput Biol*, 9:1–16, 02.

Day-Richter, J., Harris, M. A., Haendel, M., Gene Ontology OBO-Edit Working Group, and Lewis, S. (2007). OBO-Edit–an ontology editor for biologists. *Bioinformatics (Oxford, England)*, 23(16):2198–2200, August.

Eckart de Castilho, R. and Gurevych, I. (2014). A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Ferrucci, D. and Lally, A. (2004). UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348.

Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al. (2005). Galaxy: a platform for interactive large-scale genome analysis. *Genome research*, 15(10):1451–1455.

Golik, W., Dameron, O., Bugeon, J., Fatet, A., Hue, I., Hurtaud, C., Reichstadt, M., Salaün, M.-C., Vernet, J., Joret, L., et al. (2012). ATOL: the multi-species livestock trait ontology. In *Metadata and Semantics Research*, pages 289–300. Springer Berlin Heidelberg.

Ide, N., Pustejovsky, J., Cieri, C., Nyberg, E., Wang, D., Suderman, K., Verhagen, M., and Wright, J. (2014). The language application grid. In Nicoletta Calzolari, et al., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 22–30. European Language Resources Association (ELRA).

Kano, Y., Baumgartner, W., McCrohon, L., Ananiadou, S., Cohen, K., Hunter, L., and Tsujii, J. (2009). U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*, 25(15):1997–1998. in press.

Nédellec, C., Nazarenko, A., and Bossy, R., (2009). *Handbook on Ontologies*, chapter Information Extraction, pages 663–685. International Handbooks on Information Systems. Springer.

Nédellec, C., Golik, W., Aubin, S., and Bossy, R., (2010). *Knowledge Engineering and Management by the Masses: 17th International Conference, EKAW 2010. Proceedings*, chapter Building Large Lexicalized Ontologies from Text: A Use Case in Automatic Indexing of Biotechnology Patents, pages 514–523. Springer Berlin Heidelberg, Berlin, Heidelberg.

Nedellec, C., Bossy, R., Valsamou, D., Ranoux, M., Golik, W., and Sourdille, P. (2014). Information extraction from bibliography for marker-assisted selection in wheat. In *Metadata and Semantics Research - 8th Research Conference, MTSR. Proceedings*, pages 301–313.

OpenMinTeD Consortium. (2016). Overview - openminted. `http://openminted.eu/about/overview/`. Accessed: 2016-02-29.

Papazian, F., Bossy, R., and Nédellec, C. (2012). AlvisAE: a collaborative web text annotation editor for knowledge acquisition. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 149–152. Association for Computational Linguistics.

Rak, R., Rowley, A., Black, W. J., and Ananiadou, S. (2012). Argo: an integrative, interactive, text mining-based workbench supporting curation. *Database*, 2012:bas010.

Stenetorp, P., Pyysalo, S., Topić, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). BRAT: A web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 102–107, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wolstencroft, K., Haines, R., Fellows, D., Williams, A. R., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., de la Hidalga, A. N., Vargas, M. P. B., Sufi, S., and Goble, C. A. (2013). The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(Webserver-Issue):557–561.