Interoperability $= f($community$,$ division of labour$)$

Richard Eckart de Castilho
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science
Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de/

Abstract

This paper aims to motivate the hypothesis that practical interoperability can be seen as a function of whether and how stakeholder communities duplicate or divide work in a given area or market. We focus on the area of language processing which traditionally produces many diverse tools that are not immediately interoperable. However, there is also a strong desire to combine these tools into processing pipelines and to apply these to a wide range of different corpora. The space opened between generic, inherently "empty" interoperability frameworks that offer no NLP capabilities themselves and dedicated NLP tools gave rise to a new class of NLP-related projects that focus specifically on interoperability: *component collections*. This new class of projects drives interoperability in a very pragmatic way that could well be more successful than, e.g., past efforts towards standardised formats which ultimately saw little adoption or support by software tools.

Keywords: interoperability, community

1. Introduction

The fragmentation of corpus formats, annotation schemes, and NLP tools that we see in the area of natural language processing (NLP) is an obstacle to the effective use of NLP technology. However, it is not unusual to see such fragmentation. Given that building language resources and NLP tools requires very specific expertise and that such expertise is sparsely distributed across the globe, it is even quite natural. Another strong factor is that the researchers developing such tools and resources often need to focus on their qualification work and find it easier to build from scratch technology which they fully understand and which does exactly what they need; they consider this preferable to learning technologies which are potentially complex yet more interoperable, which they may never perfectly understand, and which may not exactly fit their needs. As a consequence, we see many NLP-related tools being implemented as stand-alone software for a single NLP task, or as more or less comprehensive NLP stacks covering multiple NLP tasks, each of these tools using their own formats and annotation schemes.

Much work has been and is presently being undertaken to address this fragmentation and to promote interoperability – some more successfully than others:

- Standardization of formats and schemata: XCES (Ide et al., 2000), LAF (Ide and Romary, 2004), GrAF (Ide and Suderman, 2007), TEI (Consortium, 2007), NIF (Hellmann et al., 2013), XMI (OMG, 2002), Folia (van Gompel and Reynaert, 2013), etc.
- Interoperability frameworks abstracting over individual tools and focus on a common data exchange model and workflow modelling process: GATE (Cunningham et al., 2011), UIMA (Ferrucci and Lally, 2004), to some extent also NLTK (Bird et al., 2009) or CoreNLP (Manning et al., 2014), etc.
- NLP platforms allowing to build workflows from a set of integrated components: U-Compare (Kano et al., 2011), WebLicht (Hinrichs et al., 2010), etc.

However, in most of these cases, the efforts are primarily directed at the NLP community at large, trying to convince stakeholders to adopt specific standards or formats themselves and to make their own tools compatible with these. This creates an unhealthy competition between standards, formats, and interoperability platforms for the attention and commitment of the stakeholders and does not help in addressing the common goal of all these efforts – namely, improving interoperability and reducing fragmentation.

In other areas – for example, in the space of Linux distributions – we face a similar situation as in NLP, although at a much larger scale: there are many thousands of tools and libraries, each developed and maintained mostly by small groups of people. However, the task of packaging up these tools into a Linux distribution and giving such a distribution a uniform feeling (e.g. in terms of installation and configuration) is handled by separate dedicated communities focussing on this specific task.

A similar community structure and division of work may be a suitable strategy also for the area of NLP. Specifically, the task of wrapping NLP tools for interoperability frameworks should fall neither to the developers of the NLP tools nor to the developers of the interoperability frameworks; rather, it should rather be handled by a dedicated community or communities focussing exclusively on *component collections*. The data format and schema at the heart of the collection is driven by the needs of the integrated components. While it may be less generic than formats and schemata developed independently, a broad-supporting component collection may give a much better incentive for users to stick to such a format than a generic, independently developed format without broad tool support could.

2. Differentiating the Stack

2.1. Interoperability Frameworks

GATE was one of the first frameworks to provide an abstraction over individual NLP tools via a common data exchange model and workflow process. It still offers one of the most comprehensive NLP ecosystems, covering the full

stack from analysis tools to graphical user interfaces for all kinds of NLP-related tasks. GATE is maintained mostly by a group of developers at the University of Sheffield who steadily improve and expand the GATE ecosystem. However, few third parties provide GATE components, and there are presently rather few community contributions to the GATE core.

Another popular interoperability framework is Apache UIMA. The focus of UIMA is more specific than that of GATE, mainly targeting a common data model and the building of scalable workflows. Apache UIMA provides hardly any actual NLP components, nor does it define a schema (i.e. type system) for components to communicate with each other. This makes the framework unattractive to many "end user" researchers who wish to build NLP systems, but it allows communities to form that fill the gap between the plain interoperability framework, the tool providers, and the end users. UIMA was initially developed at IBM and later transformed into a community project the Apache Software Foundation. Still, many of the core UIMA developers have day jobs at IBM. Contributions from third parties are also rather few.

2.2. Component Collections

There are several examples of communities maintaining component collections, although with slightly different goals. This paper will focus here on collections based on UIMA, but similar considerations likely apply to the GATE ecosystem. For example, ClearTK (Ogren et al., 2009) integrates a small set of NLP tools, but its main strength is actually statistical NLP, i.e. building machine learning approaches for NLP, training reusable models, etc. Another example is Apache cTAKES (Savova et al., 2010) which provides UIMA-based components to process medical records, integrating some third-party NLP tools and also providing some original components and in particular domain-specific NLP models. U-Compare integrates a wide range of third-party NLP tools with UIMA, with a focus on comparing results generated from different NLP pipeline setups. DKPro Core (Eckart de Castilho and Gurevych, 2014) aims for a high-quality and easily usable integration of a broad range of NLP tools with UIMA, and does not have any other mission beyond that.

It should also be noticed that some tool providers have started integrating their tools with UIMA, for example Apache OpenNLP[1], but this integration is barely being maintained and further developed. The OpenNLP components are intended to be adaptable to different type systems and thus be usable by a wider range of users. However, this does appear to work out well for various reasons (e.g. the extra configuration overhead and the approach's limitation to specific type system designs). Instead, different component collections wrap OpenNLP over and over again. This appears to be a typical example supporting the view that tool providers should not bother with integrating their tools with interoperability frameworks, but rather leave this task to the component collections.

There are various criteria by which component collections can be compared. The underlying interoperability framework is of course the first obvious criterion. In particular for UIMA component collections, the type system is presently a very central element: every component collection uses its own type system with specific strengths and awkwardnesses. But these are not the only differentiation criteria. Other criteria include the variety and number of integrated tools, the flexibility in configuring these tools, the ease of configuration, the easy of deployment, the quality of the documentation, the licence, the activity of the developer community, and the project governance model.

Variations in these criteria make some collections more attractive to specific user communities than others. Some of these factors and their effects are hard to measure (e.g. ease of use, governance model). Also, if the developer communities themselves conduct such measurement, they would have to divert valuable resources from actually working on the project. As the communities driving these projects typically do so as a volunteer side-product of their actual work in research or industry, such effort is typically not taken. As the aim of our present paper is to incite reflection and generate discussion on the current state of interoperability, rather than to perform a detailed analysis of component collections, we do not engage in such a detailed comparison at this point. Instead, the following section briefly presents a subjective view on the strategies taken in DKPro Core with respect to these criteria.

2.3. A Closer Look at a Component Collection

DKPro Core is a collection of components for the UIMA framework. It integrates a broad range of third-party NLP tools using the DKPro Core type system. The type system mainly covers the basic layers of linguistic analysis including tokenization, part-of-speech tagging, chunking, parsing, named entities, coreference, semantic role labelling, and more. DKPro Core is implemented in Java which makes it portable across major system platforms.

Tools DKPro Core tries to integrate as many third-party components for the different analysis levels as possible, but this is naturally limited by developer resources. DKPro Core 1.8.0 will consist of $\approx$ 100 analytics components and will support $\approx$ 50 data formats. Well-engineered tools with few transitive dependencies are easier to integrate than complex tools. In particular, tools that address the higher levels of linguistic analysis are often more difficult to integrate if these tools themselves already include multiple pre-processing steps. For example, the BART coreference resolution tool (Versley et al., 2008) includes many preprocessing components, which makes it time-consuming engineering task to isolate the actual coreference resolution aspect and to integrate that as a UIMA component in DKPro Core. Simply including the whole BART system, including all the third-party libraries it depends on, as a single component could be done but may easily lead to runtime problems (e.g. conflicting library versions).[2]

[1] http://opennlp.apache.org

[2] It should be noted that UIMA allows components to be isolated from each other to avoid these kinds of conflicts. However, this requires components to be packaged as UIMA PEAR archives which in our view makes them less easy to use programmatically – thus DKPro Core does not presently offer PEARs. Another altern-

Configuration The configuration of components in DKPro Core aims to provide maximum flexibility, exposing as many parameters of the integrated tools as feasible, while at the same time aiming for maximum ease of use. To achieve the latter, two main approaches are taken: 1) parameters with the same or very similar meaning have the same names across all components, irrespective of whether the names are the same in the underlying tools; 2) the majority of parameters use sensible default values and do not have to be set explicitly by the user. This also entails that DKPro Core defines default models to be used. The concrete models are selected taking the language of the documents being processed into account. Furthermore, DKPro Core builds on the uimaFIT library (Roeder et al., 2009) which greatly facilitates the programmatic use of UIMA components as compared to the plain UIMA API.

Deployment DKPro Core goes to great lengths to avoid placing the burden of manually obtaining and installing NLP tools and models on the user. To this end, it integrates with the software repository ecosystem around Apache Maven, through which software and data packages can be automatically discovered and downloaded, including any transitive dependencies. Various NLP tools are distributed via Maven directly by their authors. In other cases, the DKPro Core team has packaged and uploaded tools and libraries to the Maven ecosystem, typically in coordination with the original authors, e.g. mstparser[3] or mate-tools.[4] In the case of LanugageTool,[5] the original authors even decided to adopt Maven themselves for future releases. As a general principle, only those DKPro Core components which have all their dependencies available via Maven are part of the official releases. Additionally, the models needed for the respective tools are packaged and distributed via Maven by the DKPro Core team.

Documentation The documentation of DKPro Core has been greatly improved just recently through an largely automatically generated reference documentation that aggregates snippets of documentation and metadata from multiple sources (JavaDoc, Maven, UIMA descriptors, model metadata, etc.) and compiles these into five comprehensive reference documents on the type system, components, models, I/O formats, and tagset mappings. This approach allows the project to deliver comprehensive documentation without investing unreasonable amounts of time into maintaining the same information redundantly in multiple documentation files.

Licensing Most of DKPro Core is licensed under the Apache Software License 2.0 (ASL). However, it also integrates important NLP tools licensed under the GNU General Public License (GPL) and due to the reciprocal licensing model, the corresponding DKPro Core components are also licensed under the GPL. This could in principle lead to the undesired effect that original DKPro Core code initially implemented in a GPLed module could not be moved to an ASL module as part of a refactoring – in particular, if such code had been contributed to DKPro Core by a third party. For this reason, the project has adopted a contributor licence agreement which ensures that all contributions made to the project, irrespective of whether they are made to an ASL or GPL component, are received under terms compatible with the ASL licence. Thus, the project retains full flexibility to refactor its original code even across its internal GPL/ASL licence boundaries.

Developer Community DKPro Core started out as an internal project of the UKP Lab in 2007 and took a long way from there to its present form as an open source project. With the adoption of the contributor licence agreement, DKPro Core is now able to grow into a truly community-sustained and community-driven project. With the recent closing down of Google Code, the project has moved to the GitHub social coding platform, which has led to more contributions and an increased level of interaction with users. For further community growth involving contributors from different backgrounds, research institutions, or companies, it might prove beneficial in the future to adopt a more formalised project governance model.
The effort that could be invested in DKPro Core into growing the collection to support many tools, into optimising deployment, and into making configuration easy was supported by the fact that the project focusses only on the collection and was also able to take aggregate smaller and bigger improvements from many contributors with a particular interest in interoperable components.

3. NLP Platforms Revisited

As mentioned previously, there have already been various projects building NLP platforms. However, these typically had a strong focus only enabling integration while leaving the actual integration of tools to the tool providers. This appears to be changing now as some upcoming NLP platforms seem to collaborate more closely with the providers of component collections for the integration of tools. OpenMinTeD and LAPPS are two examples of platform projects with this new strategy. They aim at integrating different component collections, even ones based on different underlying interoperability frameworks like GATE and UIMA, and make them interoperable. Instead of insisting on a single format and schema, they leave some room to support a select set of formats (e.g. UIMA XMI, GATE XML, and JSON-LD) and schemata within their platforms and already offer or plan to offer conversions between these. In this way, the platforms will be able to profit from existing, comprehensive component collections in multiple NLP ecosystems and at the same time strengthen these, direct more attention at existing collections, and help growing their communities.

4. Conclusion

Dedicated communities that focus specifically on building component collections are able to pay more attention at driving and optimising interoperability, ease of use, and ease of deployment of these components. A well-designed and comprehensive component collection should help to reduce the format and schema fragmentation, as users should

ative would be a web service–based integration, but this conflicts with DKPro Core's quest for portability.

[3] http://sourceforge.net/projects/mstparser/

[4] http://code.google.com/p/mate-tools/

[5] http://www.languagetool.org

be more likely to build on the type system offered by the collection instead of inventing a new one.

Other communities building on such collections can then focus on their actual goals, such as visually building NLP workflows, comparing results of different NLP pipeline setups, building flexible machine learning frameworks making use of NLP features, scaling out and distributed processing, building component registries, etc.

Likewise, NLP tool providers can continue to focus on their original interest of building high-quality tools and can trustfully leave the integration with interoperability frameworks to the component collection maintainers.

So to summarise, the decoupling of component collections and the associated interoperability considerations from underlying interoperability frameworks and from tools related to workflows, editing, or evaluation should be a beneficial step towards a more healthy division of labour between the communities, with less competition for the tool providers' attention and a stronger ability to reduce the fragmentation in terms of formats and schemata in our field.

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. It was further supported by the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01UG1416B (CEDIFOR). Thanks to Angus Roberts and Tristan Miller for their valuable comments.

6. Bibliographical References

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

Consortium, T. (2007). TEI P5: Guidelines for Electronic Text Encoding and Interchange. Guidelines, TEI Consortium, November. URL: http://www.tei-c.org/Guidelines/P5/.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., Gorrell, G., Funk, A., Roberts, A., Damljanovic, D., Heitz, T., Greenwood, M. A., Saggion, H., Petrak, J., Li, Y., and Peters, W. (2011). *Text Processing with GATE (Version 6)*.

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. ACL and Dublin City University.

Ferrucci, D. and Lally, A. (2004). UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3–4):327–348, September.

Hellmann, S., Lehmann, J., Auer, S., and Brümmer, M. (2013). Integrating NLP using linked data. In *The Semantic Web–ISWC 2013*, pages 98–113. Springer.

Hinrichs, M., Zastrow, T., and Hinrichs, E. (2010). WebLicht: Web-based LRT services in a distributed eScience infrastructure. In Nicoletta Calzolari, et al., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 489–493, Valletta, Malta, May. European Language Resources Association (ELRA).

Ide, N. and Romary, L. (2004). International standard for a linguistic annotation framework. *Nat. Lang. Eng.*, 10(3-4):211–225, September.

Ide, N. and Suderman, K. (2007). GrAF: A graph-based format for linguistic annotations. In *Proceedings of the Linguistic Annotation Workshop*, pages 1–8, Prague, Czech Republic, June. ACL.

Ide, N., Bonhomme, P., and Romary, L. (2000). XCES: An XML-based encoding standard for linguistic corpora. In Nicoletta Calzolari, et al., editors, *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC'00)*, pages 825–830, Athens, Greece, May. European Language Resources Association (ELRA).

Kano, Y., Miwa, M., Cohen, K. B., Hunter, L. E., Ananiadou, S., and Tsujii, J. (2011). U-Compare: A modular NLP workflow construction and evaluation system. *IBM Journal of Research and Development*, 55(3):11:1–11:10, May.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Ogren, P. V., Wetzler, P. G., and Bethard, S. J. (2009). ClearTK: A framework for statistical natural language processing. In Christian Chiarcos, et al., editors, *Proceedings of the Biennial GSCL Conference 2009, 2nd UIMA@GSCL Workshop*, pages 241–248, Potsdam, Germany, September. Gunter Narr Verlag.

OMG. (2002). OMG XML metadata interchange (XMI) specification. Technical report, Object Management Group, Inc., January.

Roeder, C., Ogren, P. V., Baumgartner Jr., W. A., and Hunter, L. (2009). Simplifying UIMA component development and testing with Java annotations and dependency injection. In Christian Chiarcos, et al., editors, *Proceedings of the Biennial GSCL Conference 2009, 2nd UIMA@GSCL Workshop*, pages 257–260. Gunter Narr Verlag.

Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., and Chute, C. G. (2010). Mayo clinical text analysis and knowledge extraction system (cTAKES): Architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513.

van Gompel, M. and Reynaert, M. (2013). FoLiA: A practical XML format for linguistic annotation – A descriptive and comparative study. *Computational Linguistics in the Netherlands Journal*, 3:63–81, December.

Versley, Y., Ponzetto, S. P., Poesio, M., Eidelman, V., Jern, A., Smith, J., Yang, X., and Moschitti, A. (2008).

BART: A modular toolkit for coreference resolution. In *Proceedings of the ACL-08: HLT Demo Session*, pages 9–12, Columbus, Ohio, June. ACL.