Towards automation in using multi-modal language resources: compatibility and interoperability for multi-modal features in Kachako

Yoshinobu Kano
PRESTO, JST (Japan Science and Technology Agency)
4-1-8 Honcho, Kawaguchi, Saitama 332-0012, Japan
E-mail: kano@nii.ac.jp

Abstract

Use of language resources including annotated corpora and tools is not easy for users, as it requires expert knowledge to determine which resources are compatible and interoperable. Sometimes it requires programming skill in addition to the expert knowledge to make the resources compatible and interoperable when the resources are not created so. If a platform system could provide automation features for using language resources, users do not have to waste their time as the above issues are not necessarily essential for the users' goals. While our system, Kachako, provides such automation features for single-modal resources, multi-modal resources are more difficult to combine automatically. In this paper, we discuss designs of multi-modal resource compatibility and interoperability from such an automation point of view in order for the Kachako system to provide automation features of multi-modal resources. Our discussion is based on the UIMA framework, and focuses on resource metadata description optimized for ideal automation features while harmonizing with the UIMA framework using other standards as well.

Keywords: automation, multi-modal, UIMA

1. Introduction

Although there have been many discussions about metadata of language resources, such discussions tend to focus on human readable metadata but not on machine readable metadata. For example, human readable information of authors, license, and organization would be useful when creating a catalogue of language resources. However, machine readable metadata is also important when we need to combine language resources.

A Natural language processing (NLP) task is normally accomplished by combining a couple of language resources, including annotated corpora and NLP tools. Users of language resources are required to understand behaviours of the resources in order to find which resources can be combined. Such behavioural information is, in most cases, only fragmentally described in the resource metadata, or in worse cases, users need to investigate the corpus annotations (or source codes when the resource is an NLP tool) directly.

We claim that combinations of language resources, including comparisons and evaluations, can be automated if language resources are properly designed and their metadata well described in a machine readable way. Such a design can largely reduce human work, allowing the users to concentrate on the essential part of their entire task. This claim requires that several layers of resource representation, such as the data format, data type definitions, and resource metadata descriptions, should be standardized in a compatible and interoperable way.

We adopt the UIMA framework (Ferrucci, et al., 2006) as the base framework for the compatibility and interoperability. UIMA, Unstructured Information Management Architecture, is getting widely used in the community, e.g. the CMU component repository, JCoRe (Hahn, et al., 2008), BioNLP Component Repository (Baumgartner, et al., 2008), ClearTK (Ogren, et al., 2008) and the UIMA-fr project (Hernandez, et al., 2010). IBM's Watson question answer system (Ferrucci, 2011), which is now very famous as winning the Jeopardy! Quiz competing with a human champion, is also based on the UIMA framework.

Since UIMA is a generic framework, it is still not enough to be truly interoperable to make the human work decreased. For example, what data types should be defined, which part of tools should be decomposed or composed into a single resource component, and what to describe in the metadata, are left to resource developers. The previous works which provide language resources as UIMA components do not sufficiently address these issues, so it was not necessarily possible to determine whether arbitrary two components can be combined; comparisons including evaluations were also impossible within the UIMA framework.

The U-Compare (Kano, et al., 2009) system allows combinations and evaluations in a UIMA compliant way. Although U-Compare addressed the above issues to some extent, U-Compare is designed as single-modal, assuming written mono-language. In addition, U-Compare does not provide sufficient automation features so it was not easy for users to exploit the combination features.

We have created a brand new system called Kachako (Kano, 2012), which provides automation features based on the UIMA framework. While Kachako is intended to provide automations for users in an ideal way, it assumes single-modal resources and multi-modal resources are currently not supported. We suggest more generic design of language resource infrastructure in this paper. This design allows combinations and evaluations of multi-modal resources in an automated way, e.g. text, audio, and cross-linguistic resources. These features will be publicly available as they are integrated into the

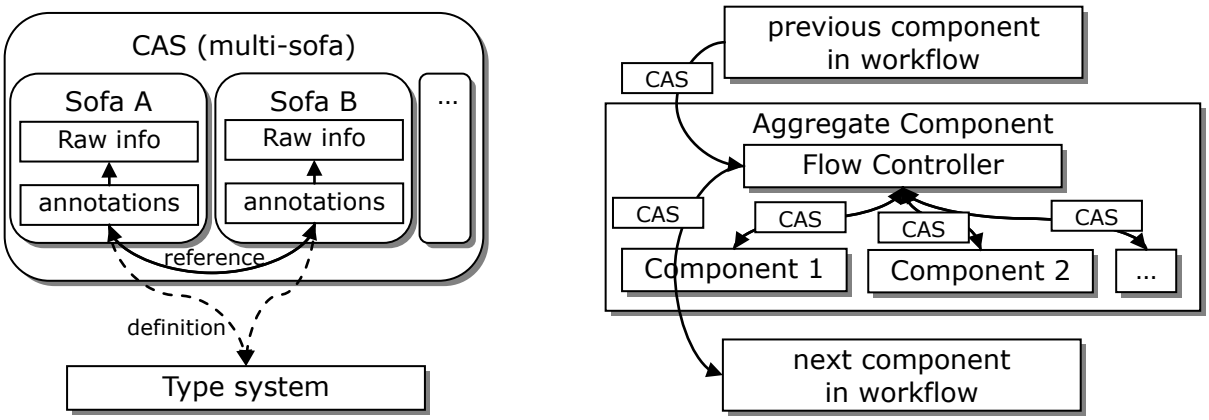

Figure 1. A conceptual figure of the UIMA framework focusing on relevant parts of this paper. The left-hand figure illustrates a CAS structure of multi-sofa. The right-hand figure shows a processing order of an aggregate component together with flow controller and child components.

Kachako system.

In this paper, we describe about UIMA briefly as background information in Section 2. Then we describe our motivation and goal in Section 3, and describe our design and implementation which achieve the goal in Section 5. We conclude this paper in Section 6 describing possible future directions.

2. UIMA

UIMA is an open framework specified by the OASIS open international standard[1]. Apache UIMA[2] provides a reference implementation as an open source project. UIMA itself is intended to be purely a framework, i.e. it does not intend to provide specific tools or type definitions. Users should develop such resources by themselves. In this section we briefly describe about UIMA focusing on architectures related to this paper. Figure 1 illustrates those architectures conceptually.

2.1 CAS: data structure

UIMA uses a Common Analysis Structure (CAS) as its standard data structure. A CAS holds raw information and annotations, e.g. raw text and linguistic annotations in case of NLP. The UIMA framework uses the "stand-off annotation" style (Ferrucci et al., 2006), which associates an annotation with the raw information via offset positions in the raw information. A CAS holds a set of such annotations while an annotation is not necessarily linked with positions directly. An annotation may refer to another annotation, thus an entire set of annotations in a CAS can represent any directed graph structure.

2.2 Type system: data type definitions

Each annotation should have its type defined explicitly. Types should be defined in a hierarchical way by developers in a UIMA's type system XML descriptor file. A type has a single parent type so a type system forms a tree structure.

2.3 Component: processing unit

A processing unit in UIMA is called a component. A primitive UIMA component is a processing unit which actually performs a specific task receiving a CAS and adding new annotations to the CAS. An aggregate UIMA component holds a set of child components deciding which component to process the CAS next. This decision is made by a flow controller specified in the aggregate component. The default flow controller is a serial pipeline while developers can create any flow using the content of input CAS of the aggregate component. This decision is made dynamically every time before processing a child component. Aggregate components can be nested. UIMA standardizes component metadata as a component descriptor XML file, which has fields of I/O capabilities (types of inputs and outputs), a flow controller in case of aggregate component, supported language names, and I/O sofa capabilities, etc.

2.4 Sofa: multi-modal data structure

UIMA provides Sofa (Subject of Analysis [3]) allowing a CAS to hold sub-CASes, which can be used to represent multi-modal information. A sofa aware i.e. multi-sofa component can access all of the sub-CASes, while a sofa unaware i.e. single-sofa component can only access the default sub-CAS. If there is two or more sofas (i.e. sub-CASes), one of the sofas should be specified as a default sofa for a sofa aware component to access a relevant sofa. There is no global information available; any information (raw information and annotations) should belong to one of the sofas. Each sofa should have a unique sofa name as a String value.

3. Motivation and Goal

Our motivation is simple: automation. That is, our goal is to provide automation features for users to achieve their individual tasks without wasting time in handling issues which can be essentially performed by the system. However, such automation is not possible by the system side only, but the resources themselves should be well

[1] http://www.oasis-open.org/committees/uima/

[2] http://incubator.apache.org/uima/

3 In the current Apache UIMA implementation, *sofa* and *view* are almost equal while they are specified differently. We use only *sofa* to avoid confusion.

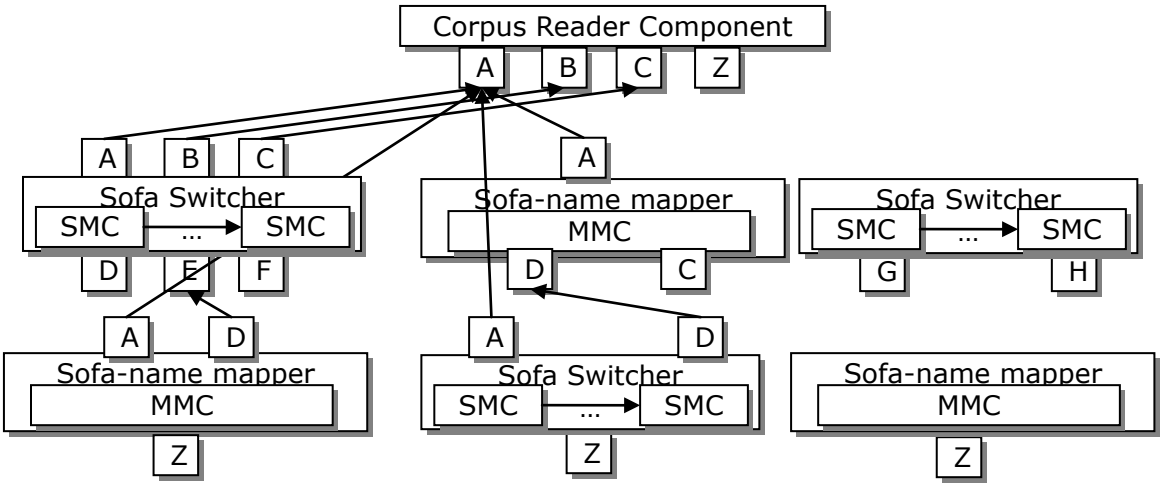

Figure 2. A conceptual figure of the multi-modal virtual workflow architecture.
SMC stands for Single-Modal Component, MMC stands for Multi-Modal Component.

described from the automation point of view.

We also claim that reusability of resources is crucial. Once a resource is created, the resource should be able to be used without modification. For example, a single-modal resource, from the users' point of view, should be able to be combined with multi-modal resources as it is.

Assuming the above conditions, our goal is to define metadata descriptions and automatically calculate possible component combinations from a given set of components. This goal requires another condition; the possible combinations of components should be calculated from metadata in a static way but not dynamically at runtime. This is because a CAS, i.e. input and output data, does not necessarily contains sufficient information to determine which combination of components is possible. For example, a person name recognizer may not detect any person name depending on its input. Thus the next component cannot notice what sort of output type may be passed from the person name detector component. This I/O information should be described in the component metadata, so the possible combinations should be calculated from metadata. By calculating combinations in this static way, all of configurations can be done by lightweight metadata without heavy executable files.

Kachako provides such architecture which calculates possible combinations of components from I/O information and efficiently process the combinations. This architecture is similar to the virtual workflow architecture (Kano et al., 2011; Kano, 2011) that allows UIMA components to be combined and compared, while Kachako's one has different design and implementation created from scratch to provide automatic workflow generation feature. This architecture calculates possible combinations of components from user specified components, assuming that the component I/O capabilities are correctly described. UIMA components implemented without aware of this architecture can be (re)used. All of the processes are performed as a single UIMA workflow while it virtually runs various workflows internally. This architecture is compliant with the UIMA standard.

Another point of this architecture is that its internal data structure is designed in an efficient way. Because combinations of components could share their input annotations passed from previous combinations of components, output annotations of each component is grouped and shared as much as possible while all of outputs are stored in a single CAS. By selecting relevant groups of annotations, each component receives and outputs annotations as if it runs in one of the virtual workflows.

4. Multi-modal Interoperability and Compatibility for Automation

Multi-modal resources can be easily represented in UIMA by exploiting the multi-sofa structure. For example, an audio-sofa and its translated text-sofa can be put together into a single multi-sofa CAS. However, there are a couple of issues when combining multi-sofa components.

Firstly, a single-sofa component and a multi-sofa component cannot be mixed without specifying a relevant default sofa for the single-sofa component to find and process an appropriate sofa. A solution would be not to use multi-sofa but put all of information together into a single sofa. However, this solution requires that all of components should be implemented to adapt this mechanism. This requires reimplementation of existing components. Further, putting differently encoded information together, e.g. audio and text, would not be practically possible because each CAS assumes to hold the same sort of raw information. Therefore, we need multi-sofa.

Secondly, the virtual workflow architecture described in the previous section does not support multi-sofa, while its comparison feature (including evaluation) is a crucial task. For example, a user of machine translation (e.g.

translation from language L1 to L2) would like to compare gold standard data (L2) with results of machine translation tools (L2). If we use multi-sofa, the original text (L1) and texts of the translated language (L2) will be stored in different sofas. Because our goal is to allow comparisons without component re-implementations, the architecture should work outside of these components while everything should be UIMA compliant. Such architecture will allow maximum reuse of potentially available UIMA components with minimum cost of users' and developers' human work.

In the sections below, we assume that the language resources are already implemented as UIMA components, which metadata describes their I/O capabilities correctly.

4.1 Multi-modal Resource Representation

Because we assume that the I/O capabilities already have sufficient information to calculate inter-component dependencies of single-modal components, the main issue here is to support multi-modality that harmonizes with the current single-modal architecture.

A component should have, even implicitly, its input and output requirements not just for annotation data types but also for language and modality. As UIMA does not specify possible values of relevant fields for modality specification, we need such specification based on standardization and theoretical requirements to allow I/O dependency calculations.

Although UIMA has "supportedLanguages" metadata field in the component metadata, supportedLauguages metadata field only assumes that its field value should be compliant with the standardized language code. This is not sufficient because a language specification may include input/output distinctions. As the current standard language code (the IETF language tag[4]) defines "private use" subtag, we can put input/output distinction into this private use subtag without violating the standard.

The sofa name is also unspecified in UIMA; the only requirement is that sofa names should be unique within a single CAS. Because the sofa name is the only metadata which describes modality of a sofa, the sofa name should be sufficiently specified to represent the content type. MIME (Multipurpose Internet Mail Extension) would be the relevant standard as MIME can specify text, image, video, etc. These usages of standards would not be a single solution; we may use more suitable alternative in future.

4.2 Multi-modal Virtual Workflow Architecture

Our multi-modal virtual workflow architecture is conceptually shown in Figure 2. This architecture only assumes that each of the UIMA components has properly described component metadata as discussed above. The sofa names are mapped to be unique while maintaining original MIME as prefix. If a pipeline of single-sofa components is used, it will be wrapped by a sofa switcher component which maps a relevant sofa to/from the default sofa.

Each group of annotations corresponds to a component which created these annotations. Each group holds corresponding I/O sofa name(s) which are specified when creating the annotations. By these extended information, the virtual workflow architecture can support multi-sofa components in a similar way of its single-sofa version.

5. Summary and Future Directions

We suggested a multi-modal resource representation and multi-modal virtual workflow architecture, all compliant with the UIMA standard. We will provide implementation of such resources integrated in our Kachako system. These resources can reduce human works by allowing reuses of UIMA standardized resources, automation of combinations, and comparisons/evaluations, by minimum cost of human work. Increasing the number of multi-modal language resources available in the suggested way would be a future work.

6. Acknowledgments

This work was partially supported by JST PRESTO and Grant-in-Aids for Scientific Research (C) [21500130] (MEXT, Japan).

7. References

Baumgartner, W.A., Jr., Cohen, K.B. and Hunter, L. (2008) An open-source framework for large-scale, flexible evaluation of biomedical text mining systems, *J Biomed Discov Collab*, **3**, 1.

Ferrucci, D.*, et al.* (2006) Towards an Interoperability Standard for Text and Multi-Modal Analytics. IBM Research Report.

Ferrucci, D.A. (2011) IBM's Watson/DeepQA, *SIGARCH Comput. Archit. News*, **39**.

Hahn, U.*, et al.* (2008) An Overview of JCoRe, the JULIE Lab UIMA Component Repository. *LREC'08 Workshop, Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*. Marrakech, Morocco, pp. 1-8.

Hernandez, N.*, et al.* (2010) Building a French-speaking community around UIMA, gathering research, education and industrial partners, mainly in Natural Language Processing and Speech Recognizing domains. *LREC 2010 Workshop of New Challenges for NLP Frameworks*. Valletta, Malta.

Kano, Y. (2012) Kachako: A Data-Centric Platform for Full Automation of Service Selection, Composition, Scalable Deployment and Evaluation. To appear.

Kano, Y.*, et al.* (2009) U-Compare: share and compare text mining tools with UIMA, *Bioinformatics*, **25**, 1997-1998.

Ogren, P.V., Wetzler, P.G. and Bethard, S. (2008) ClearTK: A UIMA Toolkit for Statistical Natural Language Processing. *LREC 2008 workshop 'Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*. Marrakech, Morocco, pp. 32-38.

[4] http://tools.ietf.org/rfc/bcp/bcp47.txt