Between Platform and APIs: Kachako API for Developers

Yoshinobu Kano
Faculty of Informatics, Shizuoka University, Japan
`kano@inf.shizuoka.ac.jp`

Abstract

Different types of users require different functions in NLP software. It is difficult for a single platform to cover all types of users. When a framework aims to provide more interoperability, users are required to learn more concepts; users' application designs are restricted to be compliant with the framework. While an interoperability framework is useful in certain cases, some types of users will not select the framework due to the learning cost and design restrictions. We suggest a rather simple framework for the interoperability aiming at developers. Reusing an existing NLP platform Kachako, we created an API oriented NLP system. This system loosely couples rich high-end functions, including annotation visualizations, statistical evaluations, annotation searching, etc. This API do not require users much learning cost, providing customization ability for power users while also allowing easy users to employ many GUI functions.

1 Introduction

A platform type of NLP software tends to provide rich GUI functions for easy users to help avoiding burdensome tasks that are not essential for their purposes. However, power users require customization ability in an API oriented way. There has been many efforts to create interoperable NLP systems, including GATE (Cunningham et al., 2002), Taverna (Hull et al., 2006), Galaxy (Blankenberg et al., 2010), Langrid (Ishida, 2006), Heart of Gold (Schäfer, 2006), PANACEA (Bel, 2010), etc.

UIMA (Unstructured Information Management Architecture) is an interoperability framework originally developed by IBM (Ferrucci et al., 2006), currently an open source project in Apache UIMA[2]. Most of UIMA related works are UIMA component implementations, including OpenNLP, JulieLab (Hahn et al., 2008), CCP BioNLP (Baumgartner Jr. et al., 2008), U-Compare (Kano et al., 2009), UIMA-fr (Hernandez et al., 2010), DKPro (Müller et al., 2008), cTAKES (Savova et al., 2010), etc.

Among the UIMA based systems, Kachako (Kano, 2012b) provides many generic features useful for developers to analyze and improve their applications. Unfortunately, these features are parts of a large integrated system which are not easy for the developers to partially reuse. Furthermore, certain number of developers avoid learning UIMA due to UIMA's rich but complex higher interoperability concepts.

We suggest a simplified interface that just requires the so-called stand-off annotation style data structure. In order for developers to more easily reuse Kachako's features, we built an API oriented NLP system based on Kachako, changing to the simplified interface discarding its UIMA compliancy.

We describe points of the UIMA framework (Section 2), the features of the Kachako platform (Section 3), and our suggested simple framework with actual API oriented system (Section 4), finally concluding this paper (Section 5).

2 UIMA

We introduce the basic architecture of UIMA briefly in this section. We refer to the Java implementation of Apache UIMA as UIMA here.

A tool is represented as a *component* in UIMA. A component is a processing unit of UIMA. In UIMA, components are combined in a *workflow*. Processing order of components can be programmable, while most workflows are simple pipelines. UIMA's data structure, *CAS (Common Analysis Structure)*, is in the so-called stand-off format, which consists of a raw (text) data part and an annotation part. The Java version of CAS is called *JCas*. An annotation should be typed by a *type system*, a user defined type

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

hierarchy. A component receives a CAS, may update the CAS and returns the CAS. A component may have its input and output types specified. These types are defined in a UIMA's type system descriptor XML file by the user. Each type is also defined in a corresponding Java class. Basically, CAS should include everything in UIMA. A CAS has one or more *sofa* (Subject OF Analysis, or sometimes called *view*) to hold multi-modal information such as text and audio, original and translated text, etc.

As a whole, in order to use UIMA for combining various NLP tools, researchers should create workflows for combining tools and define types for connecting the input and output of these tools. When running a UIMA workflow, users normally use UIMA's workflow API where CAS creation and disposal are controlled by the UIMA framework side, not by Java VM. A CAS is passed from component to component, then disposed from the Java heap memory after returned by a final component of a workflow. Therefore, developers need to pay attention not to hold any reference to content of a CAS after the CAS finished a workflow, else memory leak occurs even this is a Java API. Its reason is explained as an efficient memory usage by the Apache UIMA documentation. This makes a pitfall to developers.

3 Kachako Platform

The Kachako platform aims to provide automation features on top of the UIMA framework (Kano, 2012a)(Kano, 2012b). If the users can complete their tasks by GUI operations, most UIMA things are obscured in Kachako. However, users are required knowledge of some UIMA concepts.

Roughly speaking, Kachako has four features: automatic workflow creation, automatic workflow execution, Annotation Viewer GUI for result analysis with statistical table for comparison and evaluation, Annotation Searcher to index and search the results. We explain each feature below.

As Kachako aims to provide automation features, installation, update and execution of the platform itself is automatic in a web-based way. Kachako has its tens of own UIMA components that are also ready-to-use by automatic installation feature. Users can also register their own UIMA components to create a UIMA workflow using these UIMA components. Users can create a UIMA workflow automatically or manually by selecting UIMA components. Figure 1 shows the Kachako's workflow creation GUI. Each round-bordered box corresponds to a UIMA component, which input ports are shown in the right, output ports are shown in the right. These I/O ports are specific to Kachako, which ensures connections between components. The automatic workflow creation feature uses these I/O information to calculate (partial) workflow candidates when a start component and an end component are specified by users. These ports may belong to different sofas, shown as thick coloured labels in Figure 1. Each component may have configuration parameters, whose setting panel is available in this GUI.

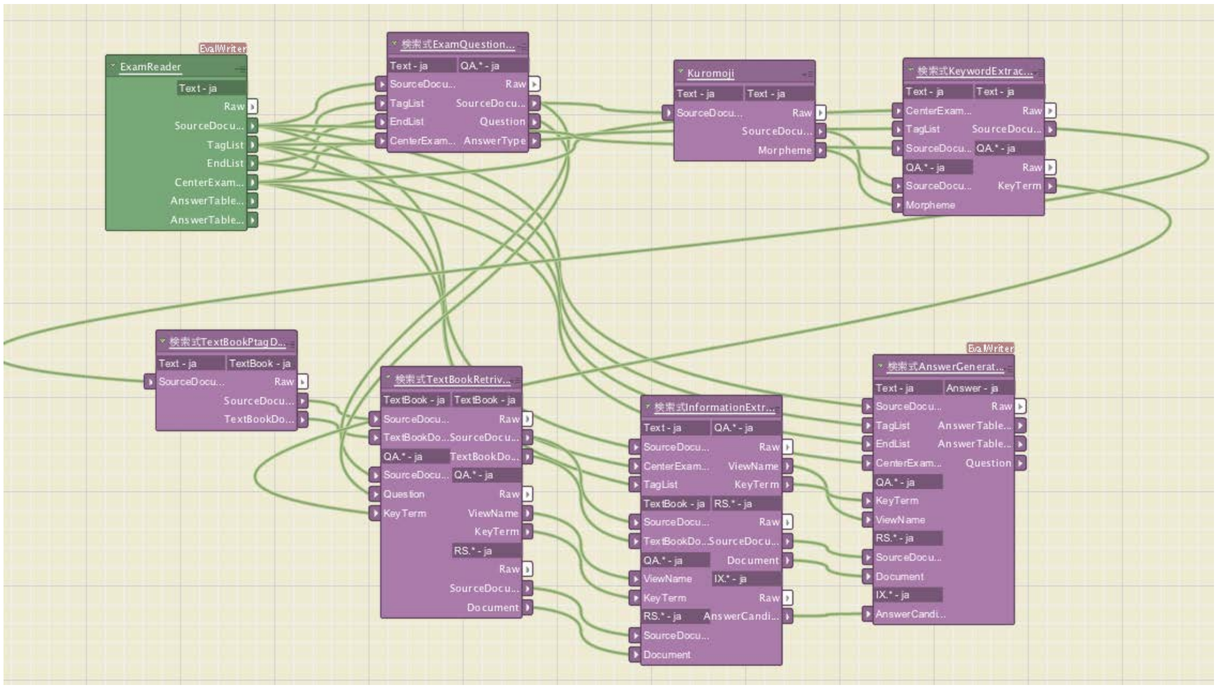
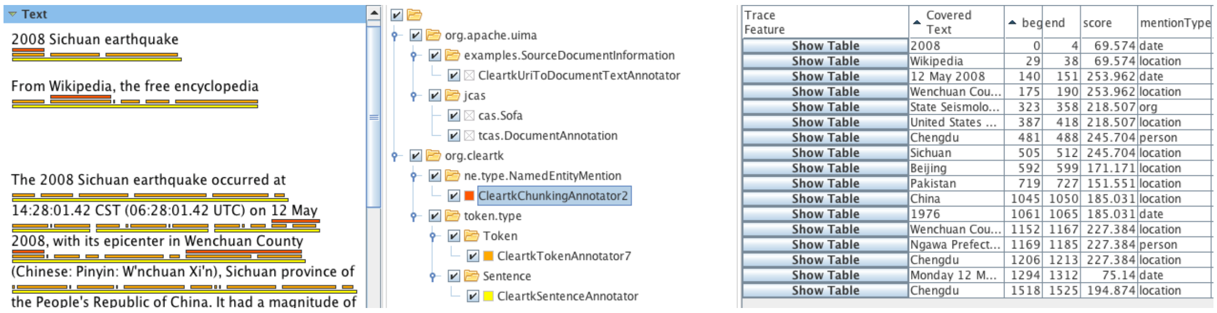


Figure 1. A screenshot of workflow creation GUI. This complex example shows a workflow of the History examination solver for the Todai Robot project (Kano, 2014).

Once a workflow is created, installation and execution of the workflow and its components are available automatically. As this is a UIMA workflow, workflow results are stored as CASes. Figure 2 shows a screenshot of the Annotation Viewer GUI that fully visualizes contents of a CAS. The leftmost panel shows the visualization result where each annotation is displayed as an underline with the raw text part. Users can filter which annotation types to show in the middle panel. The rightmost panel shows all of field values of annotations in a sortable way. Users can select a specific annotation to jump and highlight.



Text

2008 Sichuan earthquake

From Wikipedia, the free encyclopedia

The 2008 Sichuan earthquake occurred at 14:28:01.42 CST (06:28:01.42 UTC) on 12 May 2008, with its epicenter in Wenchuan County (Chinese: Pinyin: W'nchuan Xi'n), Sichuan province of the People's Republic of China. It had a magnitude of

- org.apache.uima
  - examples.SourceDocumentInformation
    - CleartkUriToDocumentTextAnnotator
  - jcas
    - cas.Sofa
    - tcas.DocumentAnnotation
- org.cleartk
  - ne.type.NamedEntityMention
    - CleartkChunkingAnnotator2
  - token.type
    - Token
      - CleartkTokenAnnotator7
    - Sentence
      - CleartkSentenceAnnotator

Trace Feature	Covered Text	beg	end	score	mentionType
Show Table	2008	0	4	69.574	date
Show Table	Wikipedia	29	38	69.574	location
Show Table	12 May 2008	140	151	253.962	date
Show Table	Wenchuan Cou...	175	190	253.962	location
Show Table	State Seismolo...	323	358	218.507	org
Show Table	United States ...	387	418	218.507	location
Show Table	Chengdu	481	488	245.704	person
Show Table	Sichuan	505	512	245.704	location
Show Table	Beijing	592	599	171.171	location
Show Table	Pakistan	719	727	151.551	location
Show Table	China	1045	1050	185.031	location
Show Table	1976	1061	1065	185.031	date
Show Table	Wenchuan Cou...	1152	1167	227.384	location
Show Table	Ngawa Prefect...	1169	1185	227.384	person
Show Table	Chengdu	1206	1213	227.384	location
Show Table	Monday 12 M...	1294	1312	75.14	date
Show Table	Chengdu	1518	1525	194.874	location

Figure 2. A screenshot of Annotation Viewer GUI.

Kachako has a comparison and evaluation feature where users can plug their own evaluation metrics as UIMA components. Figure 3 shows an example of a comparison workflow. Purple boxes in the right are evaluation components that receives a pair of input ports of same type, performs comparisons.

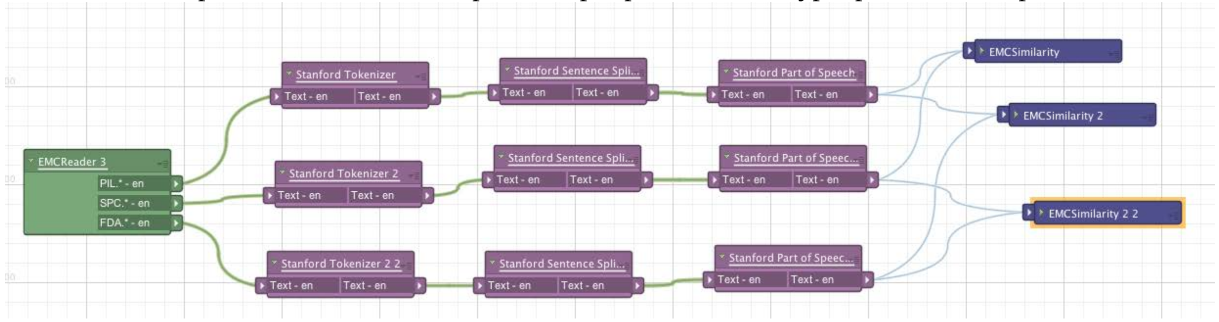
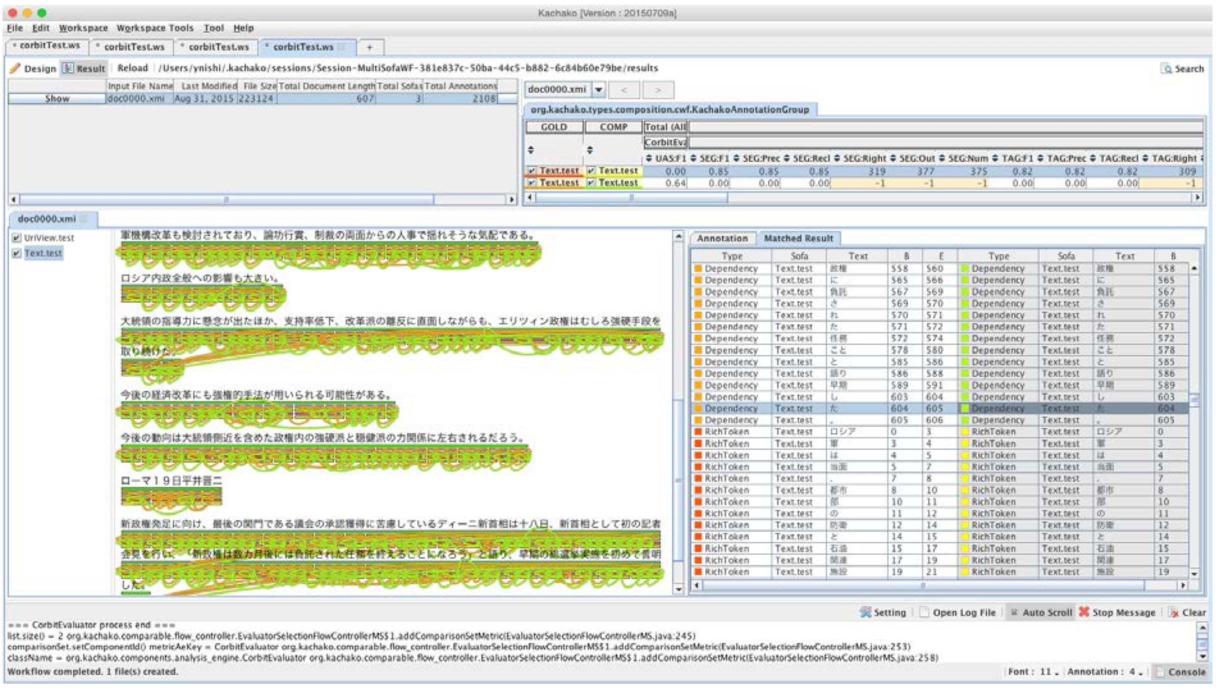


Figure 3. A screenshot of a comparison/evaluation workflow.



Kachako [Version : 20150709a]

File Edit Workspace Workspace Tools Tool Help

* corbitTest.ws | * corbitTest.ws | * corbitTest.ws | * corbitTest.ws | +

Design | Result | Reload | /Users/ynishi/.kachako/sessions/Session-MultiSofaWF-381e837c-50ba-44c5-b882-6c84b60e79be/results | Search

	Input File Name	Last Modified	File Size	Total Document Length	Total Sofas	Total Annotations
Show	doc0000.xmi	Aug 31, 2015	223124	607	3	2108

doc0000.xmi

org.kachako.types.composition.cwf.KachakoAnnotationGroup

GOLD	COMP	UAS:F1	SEG:F1	SEG:Prec	SEG:Recl	SEG:Right	SEG:Out	SEG:Num	TAG:F1	TAG:Prec	TAG:Recl	TAG:Right
✓ Text.test	✓ Text.test	0.00	0.85	0.85	0.85	319	377	375	0.82	0.82	0.82	309
✓ Text.test	✓ Text.test	0.64	0.00	0.00	0.00	-1	-1	-1	0.00	0.00	0.00	-1

doc0000.xmi

✓ UriView.test
✓ Text.test

軍機構改革も検討されており、論功行賞、制裁の両面からの人事で揺れそうな気配である。

ロシア内政全般への影響も大きい。

大統領の指導力に懸念が出たほか、支持率低下、改革派の離反に直面しながらも、エリツィン政権はむしろ強硬手段を取り続けた。

今後の経済改革にも強権的手法が用いられる可能性がある。

今後の動向は大統領側近を含めた政権内の強硬派と穏健派の力関係に左右されるだろう。

ローマ１９日平井晋二

新政権発足に向け、最後の関門である議会の承認獲得に苦慮しているディーニ新首相は十八日、新首相として初の記者会見を行い、「新政権は数カ月後には負託された任務を終えることになろう」と語り、早期の総選挙実施を初めて言明した。

Annotation | Matched Result

Type	Sofa	Text	B	E	Type	Sofa	Text	B
Dependency	Text.test	政権	558	560	Dependency	Text.test	政権	558
Dependency	Text.test	に	565	566	Dependency	Text.test	に	565
Dependency	Text.test	負託	567	569	Dependency	Text.test	負託	567
Dependency	Text.test	さ	569	570	Dependency	Text.test	さ	569
Dependency	Text.test	れ	570	571	Dependency	Text.test	れ	570
Dependency	Text.test	た	571	572	Dependency	Text.test	た	571
Dependency	Text.test	任務	572	574	Dependency	Text.test	任務	572
Dependency	Text.test	こと	578	580	Dependency	Text.test	こと	578
Dependency	Text.test	と	585	586	Dependency	Text.test	と	585
Dependency	Text.test	語り	586	588	Dependency	Text.test	語り	586
Dependency	Text.test	早期	589	591	Dependency	Text.test	早期	589
Dependency	Text.test	し	603	604	Dependency	Text.test	し	603
Dependency	Text.test	た	604	605	Dependency	Text.test	た	604
Dependency	Text.test	。	605	606	Dependency	Text.test	。	605
RichToken	Text.test	ロシア	0	3	RichToken	Text.test	ロシア	0
RichToken	Text.test	軍	3	4	RichToken	Text.test	軍	3
RichToken	Text.test	は	4	5	RichToken	Text.test	は	4
RichToken	Text.test	当面	5	7	RichToken	Text.test	当面	5
RichToken	Text.test	、	7	8	RichToken	Text.test	、	7
RichToken	Text.test	都市	8	10	RichToken	Text.test	都市	8
RichToken	Text.test	部	10	11	RichToken	Text.test	部	10
RichToken	Text.test	の	11	12	RichToken	Text.test	の	11
RichToken	Text.test	防衛	12	14	RichToken	Text.test	防衛	12
RichToken	Text.test	と	14	15	RichToken	Text.test	と	14
RichToken	Text.test	石油	15	17	RichToken	Text.test	石油	15
RichToken	Text.test	関連	17	19	RichToken	Text.test	関連	17
RichToken	Text.test	施設	19	21	RichToken	Text.test	施設	19

Setting | Open Log File | Auto Scroll | Stop Message | Clear

```
=== CorbitEvaluator process end ===
list.size() = 2 org.kachako.comparable.flow_controller.EvaluatorSelectionFlowControllerMS$1.addComparisonSetMetric(EvaluatorSelectionFlowControllerMS.java:245)
comparisonSet.setComponentId() metricAeKey = CorbitEvaluator org.kachako.comparable.flow_controller.EvaluatorSelectionFlowControllerMS$1.addComparisonSetMetric(EvaluatorSelectionFlowControllerMS.java:253)
className = org.kachako.components.analysis_engine.CorbitEvaluator org.kachako.comparable.flow_controller.EvaluatorSelectionFlowControllerMS$1.addComparisonSetMetric(EvaluatorSelectionFlowControllerMS.java:258)
Workflow completed. 1 file(s) created.
```

Font : 11 | Annotation : 4 | Console

Figure 4. A screenshot of the Annotation Viewer for a comparison/evaluation workflow.

Figure 4 shows a result window of such an evaluation workflow. The upper-left panel shows statistics for each CAS, e.g. numbers of annotations. The upper-right panel shows statistics of evaluation metrics, e.g. precision, recall, and F-score while these metrics could be customized as shown in the figure. Comparison/evaluation components should be implemented by developers as there could be variety of different metrics depending on the individual developers' purposes. The lower-left panel is the Annotation Viewer, while sofa selector is shown in the left as this is a multi-sofa case. The Annotation Viewer shows rather a complex example, where many inter-annotation links are shown as arcs. The lower-right panel shows the feature values of annotations. Users can also highlight which specific annotation was matched with respect to a given metric.

Annotation Searcher is an optional function in Kachako to enable indexing and searching of CAS contents. As the example screenshots illustrate, annotations could be very complex, whose number can be huge. It is not realistic to manually check interested patterns of annotations. However, existing search engines assume to index textual information and simple values in their default. Annotation Searcher provides a special function to search by region algebra (Clarke et al., 1995)(Jaakkola et al., 1999)(Masuda et al., 2009) which can cover most of required complex queries, including AND/OR operators, inter-annotation relationships of overlap, follow, include, etc., and inter-annotation links while mixing with the normal textual search conditions. The Annotation Searcher is built on top of Apache Solr/Lucece[3] search engine, adding this region algebra feature in an efficient way, allowing automatic parallel indexing/searching. The Annotation Searcher can be called by just checking an option, receiving any CAS content to index its text and annotations. Search result can be shown in the Annotation Viewer to highlight corresponding annotations.

4 Simplified Interoperability and Kachako API

We suggest a simplified interface for interoperability in order for the developers to more easily employ the functions of Kachako platform. We built an API version of Kachako based on this interface.

Firstly, we discard functions related to the workflow construction. Discarded functions include the automatic workflow creation and its GUI.

Secondly, we designed a pseudo JCas interface which is similar to the original UIMA JCas but just a normal Java class. This pseudo JCas has a text part and an annotations part as same as the original JCas. The annotations part assumes to hold instances of a pseudo *TOP* type or descendants we define. That is, the original type system is defined as a Java class hierarchy. Developers are simply required to produce this pseudo JCas to use the Kachako API functions.

By these designs, we could reuse the Kachako platform functions without much changes. Most of the functions are now available as API by these changes. For example, if developers generate any pseudo JCas, they can simply call the Annotation Searcher API to index and search the content of the pseudo JCas. The Annotation Viewer GUI is available just reading the pseudo JCas. The comparison and evaluation features of the Annotation Viewer GUI is also available by storing groups of annotations.

Modifying existing components into this simplified API is straightforward, because we used similar names for corresponding classes and methods. We discarded the original JCas's index feature, because the pseudo JCas could be extended to store any field, while the annotations part could be used as the original indexed annotations. We have already modified a couple of Kachako UIMA components into this new interface style, which simply required to change several lines of codes per component.

5 Conclusion and Future Work

Interoperability framework is useful but we need different level of interfaces depending on the types of users. UIMA provides a good framework for NLP users but it also restricts users' design due to its workflow management system. Kachako, a UIMA based NLP platform, also suffers the same problem just because it is compliant with UIMA. We build a new API oriented system reusing the Kachako platform, where the interoperability interface is simplified to be only stand-off annotations. This new system allows developers to use its functions without learning costs. Future work includes a new feature to employ machine learning features.

[3] https://lucene.apache.org/

Acknowledgement

This work was partially supported by MEXT Kakenhi.

Reference

Baumgartner Jr., W. A., 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), 1. Journal Article, . doi:1747-5333-3-1 [pii]10.1186/1747-5333-3-1

Bel, N. (2010). Platform for automatic, normalized annotation and cost-effective acquisition of language resources for human language technologies. panacea. *Procesamiento del Lenguaje Natural*, *45*, 327–328. article, .

Blankenberg, D., Von Kuster, G., Coraor, N., Ananda, G., Lazarus, R., Mangan, M., Nekrutenko, A. and Taylor, J. (2010). Galaxy: a web-based genome analysis tool for experimentalists. *Curr Protoc Mol Biol*, *Chapter 19*, Unit 19 10 1-21. Journal Article, . doi:10.1002/0471142727.mb1910s89

Clarke, C. L. A., Cormack, G. V. and Burkowski, F. J. (1995). An Algebra for Structured Text Search and a Framework for its Implementation. *The Computer Journal*, *38*(1), 43–56. doi:10.1093/comjnl/38.1.43

Cunningham, H., Maynard, D., Bontcheva, K. and Tablan, V. (2002). GATE: A framework and graphical development environment for robust NLP tools and applications. *40th Anniversary Meeting of the Association for Computational Linguistics* (pp. 168–175). Conference Proceedings, Philadelphia, USA.

Ferrucci, D., Lally, A., Gruhl, D., Epstein, E., Schor, M., Murdock, J. W., Frenkiel, A., Brown, E. W., Hampp, T., et al. (2006). Towards an Interoperability Standard for Text and Multi-Modal Analytics. Report, IBM Research Report.

Hahn, U., Buyko, E., Landefeld, R., Mühlhausen, M., Poprat, M., Tomanek, K. and Wermter, J. (2008). An Overview of JCoRe, the JULIE Lab UIMA Component Repository. *LREC'08 Workshop, Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP* (pp. 1–8). Conference Proceedings, Marrakech, Morocco.

Hernandez, N., Poulard, F., Vernier, M. and Rocheteau, J. (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*. Conference Proceedings, Valletta, Malta.

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P. and Oinn, T. (2006). Taverna: a tool for building and running workflows of services. *Nucleic Acids Res*, *34*(Web Server issue), W729-32. Journal Article, . doi:34/suppl_2/W729 [pii]10.1093/nar/gkl320

Ishida, T. (2006). Language Grid: An Infrastructure for Intercultural Collaboration. *Proceedings of the International Symposium on Applications on Internet*. Conference Paper, IEEE Computer Society. doi:10.1109/saint.2006.40

Jaakkola, J. and Kilpeläinen, P. (1999). *Nested Text-Region Algebra* (techreport).

Kano, Y. (2012b). Kachako: a Hybrid-Cloud Unstructured Information Platform for Full Automation of Service Composition, Scalable Deployment and Evaluation. *the 1st International Workshop on Analytics Services on the Cloud (ASC), the 10th International Conference on Services Oriented Computing (ICSOC 2012).* Shanghai, China.

Kano, Y. (2012a). Towards automation in using multi-modal language resources: compatibility and interoperability for multi-modal features in Kachako. *The Eighth edition of the International Conference on Language Resources and Evaluation (LREC 2012)*. Istanbul, Turkey.

Kano, Y. (2014). Solving History Exam by Keyword Distribution: KJP System at NTCIR-11 QALab Task. *the 11th NTCIR (NII Testbeds and Community for information access Research) workshop* (pp. 530–531).

Kano, Y., Baumgartner, W. A., McCrohon, L., Ananiadou, S., Cohen, K. B., Hunter, L. and Tsujii, J. (2009). U-Compare: share and compare text mining tools with UIMA. *Bioinformatics*, *25*(15), 1997–1998. Journal Article, . doi:10.1093/bioinformatics/btp289

Masuda, K. and Tsujii, J. (2009). Tag-Annotated Text Search Using Extended Region Algebra. *IEICE Transactions on Information and Systems*, *E92.D*(12), 2369–2377. article, . doi:10.1587/transinf.E92.D.2369

Müller, C., Zesch, T., Müller, M.-C., Bernhard, D., Ignatova, K., Gurevych, I. and Mühlhäuser, M. (2008). Flexible UIMA Components for Information Retrieval Research. *LREC 2008 Workshop "Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP"* (pp. 24–27). Conference Proceedings, Marrakech, Morocco.

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 : JAMIA*, *17*(5), 507–13. BMJ Publishing Group Ltd. doi:10.1136/jamia.2009.001560

Schäfer, U. (2006). Middleware for creating and combining multi-dimensional NLP markup. *Proceedings of the 5th Workshop on NLP and XML: Multi-Dimensional Markup in Natural Language Processing*, NLPXML '06 (pp. 81–84). inproceedings, Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=1621034.1621050