

Component Services for Knowledge Integration in UIMA (a.k.a. SUKI)

From Text Analysis to Knowledge

SUKI is a set of component services designed to enable transforming the results of UIMA analysis into knowledge representations (e.g., [OWL](#)) suitable for formal reasoning and processing by widely available knowledge base and semantic web tools like [Protégé](#) and [Jena](#)

[UIMA](#) is a framework for composing and deploying text and multi-modal analytics to discover the latent meaning in these original unstructured sources. See for details on the UIMA architecture, and for the freely available software framework see [IBM's alphaWorks](#) site.

The results of a set of cooperating UIMA analysis components are captured in a CAS or Common Analysis Structure. The UIMA CAS manages a collection of interrelated objects that contain the results of analysis of unstructured sources. For example, *annotations* which label regions of a document are represented in a CAS. An annotation can, for example, classify a region as a type of entity, relationship or property. The specific nature the annotations are user-defined. Classic examples include mentions of names, organizations, times and events in textual documents. Technically the CAS can be thought of as a Java object model with some additional APIs for manipulating annotations, indices and views over documents generated by workflows of UIMA analysis engines.

UIMA applications take the results of analyzing large collections of documents, in the form of CASes, and build resources that provide precise, application-specific access to the most relevant content discovered therein. So for example, having discovered mentions of financial organizations and relationships that link them to each other, the application builds a search index incorporating these types so that documents containing this information can be precisely targeted by end-user queries.

Another important application of this technology is to build more formal representations of discovered content. Rather than, for example a search engine index that includes specific types in addition to just keywords, certain applications need to build a formal OWL knowledge base (KB) from the results captured in a stream of CASes.

Once a KB, in a standard representation like OWL, is built, off-the-shelf tools such as OWL reasoners and KB processing tools can be applied to these results to find, extend or further refine the knowledge extracted from a collection of raw documents by the UIMA analysis engines. For an overview of the challenges to generating formal knowledge bases from text analysis, see [Welty, Murdock, et al, 2005].

SUKI is built on top of UIMA and provides components and services for generating and providing access to knowledge extracted from the results of a UIMA text analysis pipeline.

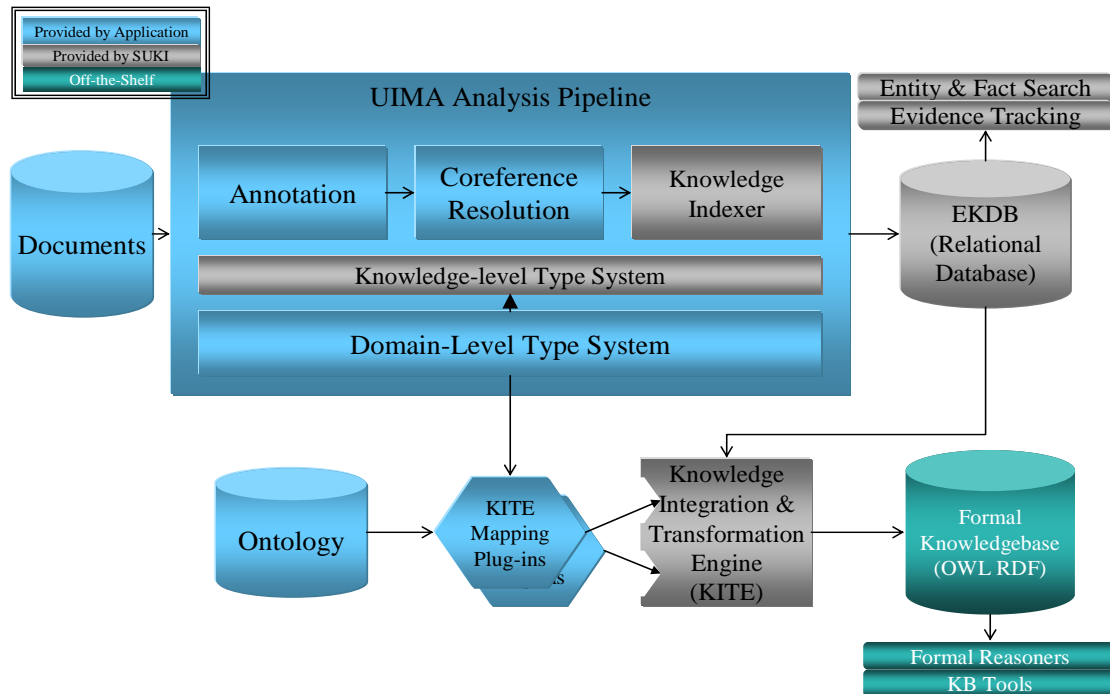


Figure 1: UIMA Knowledge Integration Component Architecture

SUKI includes four major components illustrated in the SUKI architecture in figure 1:

- 1) **Knowledge Level Type System (KLT):** A minimal UIMA upper-level type system for categorizing analysis results to enable their mapping to formal knowledge representations. Developers who want to transform their analysis results to a formal OWL KB, for example, would define their type system as an extension of the KLT.
- 2) **Knowledge Indexer:** A UIMA CAS Consumer that takes the results of per-document analysis and the results of cross-document coreference resolution and indexes this information a relational database (i.e., the EKDB – see below for details) capturing discovered annotations, entities, relationships and links back to the engines that discovered them and to the documents in which they were discovered.
- 3) **Extracted Knowledge Database (EKDB):** A relational database and a set of JDBC services for storing, querying and accessing the raw knowledge extraction results produced by the Knowledge Indexer.
- 4) **Knowledge Integration and Transformation Engine (KITE):** A component framework for building and applying mappings between formal structured

representations. In particular, KITE has been instantiated to map from a UIMA Type System (that extends the KLT) to an OWL Ontology. This KITE instance can then apply these mappings to produce the RDF for an OWL KB from extracted knowledge in the EKDB.

The Knowledge Level Type System

In UIMA, a Type System is a schema for the CAS. It defines the types of things that maybe instantiated in a CAS by UIMA analysis engines. For example, it might include a taxonomy of legal, medical or military classes and their properties with the intent that analysis engines would detect specific mentions of these classes in text (or other types, e.g., image, video) of documents.

Analysis engines annotate mentions of entities, that is, they associate a label, like “person” to a span of text that is the name of a person, or associating the label “vehicle” to a region of an image that depicts a vehicle.

The transformation of analysis results into formal knowledge requires the explicit distinction between annotation of entities like Persons and the annotations of relationships like being the CEO of a particular company.

Relationship detectors are analysis engines that annotate mentions of relationships. For example the span of text “Fred Center is the CEO of Center Micros,” might be annotated by an analysis engine with the label “CeoOf” to indicate that it is a mention of that relationship.

The KLT extends the built-in Annotation type with the high-level UIMA types:

- EntityAnnotation and
- RelationAnnotation

so that this distinction may be explicitly captured in a standard way.

The transformation of analysis results into formal knowledge requires the explicit distinction between mentions of individuals and the individuals themselves.

Analysis engines called coreference resolvers associate annotations with individuals, and determine if two annotations refer to the same individual. For example the mentions “Mr. Center”, “Fred Center” and “CEO of Center Micros” may all be different EntityAnnotations. A coreference resolver would record in the CAS that these three annotations are coreferential – they refer to a single unique individual representing Fred Center. The same can be done for relationship annotations. For example, the CeoOf annotation on the text “Fred Center is the CEO of Center Micros” and another on the text “Mr. Center, the top executive at Center Micros,” would be associated with the single, unique relationship representing that Fred Center is the CEO of Center Micros.

A cross-document coreference resolver is a special type of coreference resolver that would record the same type of result for annotations occurring in separate documents of a collection.

The KLT defines the high-level UIMA type tree:

- Referent
 - Entity
 - Relation

so that the distinction between annotations of mentions and the unique individuals or relationships they refer to (in general “referents”) may be represented in a standard way.

The transformation of analysis results into formal knowledge requires the explicit linkage of relationships to their arguments or participating entities.

Knowledge bases store entities and relationships between entities. In order to reason about this knowledge the linkage between relationships and their arguments has to be explicitly present in the KB. For example, the individuals representing Fred Center and Center Micros must be explicitly linked to the CeoOf relationship to capture the fact that *Fred Center is the CEO of Center Micros*. In this example, we consider the individuals to be arguments to the relation. In general, a *relationship* is the association of specific referents as arguments to a specific relation.

The KLT defines the high-level UIMA type tree:

- RelationArgs (links relation annotations to their argument annotations)
- Link
 - Argument (links relation referents to their argument referents)
 - HasOccurrence (links referents to annotations that represent occurrences)

so that annotations and referents can be explicitly linked to the relationship annotations and the relationships, respectively, in which they participate.

To use UIMA knowledge integration services, an analysis engine developer must link their domain Type System to the KLT through extension. At the risk of oversimplifying this process, consider for example the Person type in a domain Type System with “Annotation” as its super type. The Person type’s super type must be changed to “EntityAnnotation” to conform to the KLT. Similar modifications would have to be made for relationships.

More detailed descriptions of the KLT and how to use it are under development. Please see the technical contacts below to get more information.

Knowledge Indexer

The Knowledge Indexer requires that:

1. a set of UIMA analysis engines have produced entity and relationship annotations in a stream of CASes and
2. a set of coreference resolvers have produced entities and relationships linking the entity and relationship annotations in those CASes to common referents.

The Knowledge Indexer selects from the results of annotation and coreference resolution and populates an instance of the Extracted Knowledge Database (EKDB). It stores discovered annotations, entities, relationships and links back to the engines that discovered them and to the documents in which they were discovered.

Extracted Knowledge Database

The Extracted Knowledge Database is a relational database that primarily stores the entities and relationships produced by the coreference resolvers. But in addition it stores all the provenance information that links this knowledge back to:

- annotations associated with the entities,
- mentions associated with the annotations,
- documents in which the mentions occur
- analysis engines that produced each result.

This information is organized in relational tables and the EKDB includes a set of JDBC services that allow the user to search for facts about entities (i.e. relationships for which an entity is an argument) and to trace these back through the engines and sources that produced them.

This information is essential for generating a variety of formal knowledge representations, tracing the knowledge extraction process and providing justifying evidence for the extracted knowledge.

Knowledge Integration and Transformation Engine (KITE)

The Knowledge Integration and Transformation Engine (KITE) is a SUKI component with a plug-in architecture that supports the transformation of data from one structured system into another. The KITE approach to data integration should be contrasted with the *federated* approach, in which data in heterogeneous systems remain in their original form and are wrapped to provide integration with other systems. In KITE, the data itself is transformed into a new system that has a single description (schema, ontology, type system).

Of particular interest is the configuration of KITE plug-ins that provide transformation from referents stored in an EKDB to an RDF graph that instantiates an OWL ontology. In other words, given an OWL ontology and an EKDB that has stored the knowledge

extracted from a set of annotators and coreference resolvers that extend the KLT, existing KITE plug-ins allow for the generation of RDF data from the EKDB.

The transformation of analysis results into formal knowledge requires the explicit mapping of types in a UIMA type system to classes and properties in a formal ontology.

The mappings in KITE are specified through a set of mapping plug-ins. Existing mapping plug-ins provide for declarative specification of common mapping patterns. For example, a simple type of mapping could be that the Person type in a UIMA type system maps to the Person class in an OWL ontology, or that the Company type maps to the Business class, etc. In other words, for each entity in the EKDB that is an instance of the Person type, there should be a corresponding node (or resource) in the RDF graph that is an instance of the Person class. There is a KITE plug-in for which these simple direct mappings are specified in a table as metadata for the plugin.

A slightly more complex mapping could be that the type Agent in a type system corresponds to either the class Person or Company in an ontology. In other words, for each entity in the EKDB that is an instance of the Agent type, there should be a corresponding instance in the RDF graph that is an instance of the class that is the union of the Person class and the Company class (OWL provides the expressiveness to state this). There is a KITE plug-in for which mappings that require the expressiveness of OWL are specified as metadata for the plug-in.

Mapping plug-ins have triggers that identify the instances in the EKDB that they apply to. When mappings require more expressiveness than the existing plug-ins provide, new ones can be added. One such plug-in exists for mapping relationships that hold at a time to OWL using a technique called “time slicing”, conforming to the draft standard for representing fluents in OWL proposed by the W3C.

KITE also provides for provenance of the transformations, i.e. a record of what mappings took place in generating the RDF graph. For example, it is often the case that a user, when presented with an RDF triple (a binary relationship), will want to know where it came from. KITE provides for the information in the RDF graph to be connected back to the referents and annotations from which they were generated.

The KITE framework is very general and can be extended with plug-ins to read/write data from different sources (e.g. EKDB, RDF store, XCAS), read/write data in different representation languages (e.g. UIMA type system, OWL), as well as different ways of recording provenance for the data transformations (e.g. EKDB, InferenceWeb).

Contact Information

For general information about UIMA Knowledge Integration Services please contact:

- David Ferrucci, ferrucci@us.ibm.com

For specific technical information about KLT and the EDKB please contact:

- J. William Murdock, murdockj@us.ibm.com

For specific technical information about KITE please contact:

- Christopher Welty, welty@us.ibm.com

Related Papers

- Christopher Welty and J. William Murdock. [The Semantics of Multiple Annotations. IBM Research Technical Report RC22979](#). 2003.
- David Ferrucci. [Text Analysis as Formal Inference for the Purposes of Uniform Tracing and Explanation Generation. IBM Research Technical Report RC23372](#). 2004.
- Christopher Welty, J. William Murdock, Paulo Pinheiro da Silva, Deborah L. McGuinness, David Ferrucci, Richard Fikes. Tracking Information Extraction from Intelligence Documents. In Proceedings of the 2005 International Conference on Intelligence Analysis (IA 2005), McLean, VA, USA, 2-6 May, 2005. http://www.ksl.stanford.edu/people/pp/papers/Welty_IA_2005.pdf
- J. William Murdock, Paulo Pinheiro da Silva, David Ferrucci, Christopher Welty and Deborah L. McGuinness. [Encoding Extraction as Inferences. In Proceedings of AAAI Spring Symposium on Metacognition on Computation](#), AAAI Press, Stanford University, USA, pages 92-97, 2005.
- Deborah L. McGuinness, Paulo Pinheiro da Silva, J. William Murdock and David Ferrucci. [Exposing Extracted Knowledge Supporting Answers. Technical Report KSL-05-03](#), Knowledge Systems Laboratory, Stanford University, USA, 2005.
- Anthony Levas, Eric Brown, J. William Murdock, and David Ferrucci. [The Semantic Analysis Workbench \(SAW\)](#): Towards a Framework for Knowledge Gathering and Synthesis. Proceedings of the International Conference on Intelligence Analysis. McLean, VA, May 2-6, 2005.