Callisto: A Configurable Annotation Workbench

David Day, Chad McHenry, Robyn Kozierok, Laurel Riek

The MITRE Corporation
202 Burlington Road
Bedford, Massachusetts, 01730, USA
{day,red,robyn,laurel}@mitre.org

Abstract

In order to support a range of textual annotation tasks, we have developed a new annotation tool called Callisto. To promote task-specific specialization of the interface and associated constraint checking, Callisto provides a facility for the independent development, compilation and installation of task module plug-ins (in the form of Java Archive jar files). The common Callisto backend provides a set of "annotation services" to which all separate GUI components can subscribe, enabling a common framework through which annotation updates are propagated to all components. A number of annotation task models have already been defined, and those that are of very general applicability have been made easily re-configurable for small changes in task definition. Callisto is implemented in Java to make use of Java's considerable support for Unicode-encoded multilingual data. Callisto is freely available for downloading and use.

Introduction

This paper and demonstration introduce Callisto, a configurable linguistic annotation workbench. This new multilingual annotation tool can be used for the development of annotated language resources in all Unicode-supported languages. The emphasis in its development has been to maximize the ability to develop new task-specific enhancements, while providing a powerful and generally useful set of "annotation services." This tool is implemented in Java and is freely available for downloading and use.

Task Configurability

The central design focus in this new annotation tool is modularity and configurability. To maximize the ease with which new task-specific modules can be developed, Callisto provides a set of core "annotation services" that are available to all annotation task modules. For example, Callisto provides a Model-View-Controller design that ensures that user interface components are independent, yet always display consistent information. All changes to the annotations, both structural and informational, are made through the annotation models control interface. Updates and changes are then propagated back to graphical components via an event handling infrastructure, to update the view presented to the annotator. Standard components provided within Callisto include a textual annotation view, and a configurable table display providing capabilities inappropriate in the textual view. Each has default creation, deletion and modification capabilities.

Callisto adopts the view that sufficiently complex annotation tasks call for task-specific models that are implemented directly in source code. This enables each task to incorporate arbitrarily complex task-specific logic (such as constraints between and among parts of the annotation structure) and task-specific GUIs (that promote the ability to view efficiently the annotations and their relationships), by giving task developers the full power of Java and access to the annotation representations. Such specialized interfaces serve to enforce (or graphically highlight) task guideline requirements. A growing number of GUI components for editing and viewing annotations are available to expedite this development process.

Custom tasks developed to date for Callisto include:

- Automatic Content Extraction (ACE) entity and relation detection, characterization and co-reference;
- TIMEX2 temporal phrase normalization;
- MUC-style named entity tagging;
- Entity Extraction & Link Discovery (EELD) event annotation;
- TimeML event and temporal expression tagging;
- MUC-style entity co-reference chains.

Tasks that are currently under development include:

- Bio-informatics annotation with indices into gene and protein ontologies;
- Cross-document coreference annotation of entities.

Because of the annotation model adopted in Callisto, all of these annotation tasks are implemented in Java code that is compiled, packaged and distributed as independent Task Archives that "plug-in" to the Callisto framework. Tasks are packaged as java archive (jar) files which are installed in a designated task directory. Tasks made available in this way are automatically incorporated into the application at runtime, promoting independent development and distribution.

Not all tasks require new code development. Those tasks that are sufficiently isomorphic to existing tasks can be established through simple modifications to existing task models. For these variants, the identical graphical user interface modalities are immediately available.

Sample Annotation Tasks Supported

The ACE Entity and Relation Annotation Task

To illustrate some of the facilities built into Callisto, we present some of the features supported in the "ACE RDC" task (Doddington, 2003). The current ACE annotation

task involves what might be viewed as two different levels of analysis – mentions and their denotation – for two classes of content – entities and relations. Entity Detection and Tracking (ACE-EDT, 2003) requires identifying all of the "mentions" (references) to an entity within a document, establishing their coreference relationships, and identifying a range of properties at both the mention and the entity levels. For example, at the entity level one indicates the semantic class of the entity (person, organization, location, facility or Geo-Political-Entity/GPE), and whether or not this is a specific or generic entity. At the entity mention level the annotator is expected to indicate the syntactic form of the reference (name, nominal or pronominal), the full textual extent and its syntactic "head," and some indication of the literal and surface references (if the reference involves metonymy). Relation Detection and Characterization (ACE-RDC, 2003) requires identifying the mentions of entities that indicate the presence of a relation holding between these entities (this is the "relation mention"), as well as the entity-level arguments to a relation that this supports. A particular relationship, such as **`located-at ( United Nations , New York )`** might be "mentioned" in two or more places within a document, establishing the counterpart of a coreference among these relation mentions.

The Callisto RDC task provides two types of "views" of the RDC annotation data. In the main default text pane (see Figure 1, upper left corner), the textual "signal" file is displayed (the document being annotated), and the mentions for each of the entities are highlighted with color coding distinguishing the syntactic "head" of the mention from the rest of the entity mention phrase. The other panes present tabular views of various types of distinct analyses, which, at the user's discretion, can also be separated out as distinct windows. These various tables present entity mentions, entities, relation mentions, and special types of temporal expressions that are used by the relation task.

Both the main text pane and the table widgets are frequently re-used GUI components. Intrinsic behaviors supported by table widgets include re-ordering of columns and sorting rows by selected columns. The field in any individual column can support a variety of value editing types (check boxes, pull-down menus, string entry, etc.). User actions to create or modify annotations can be

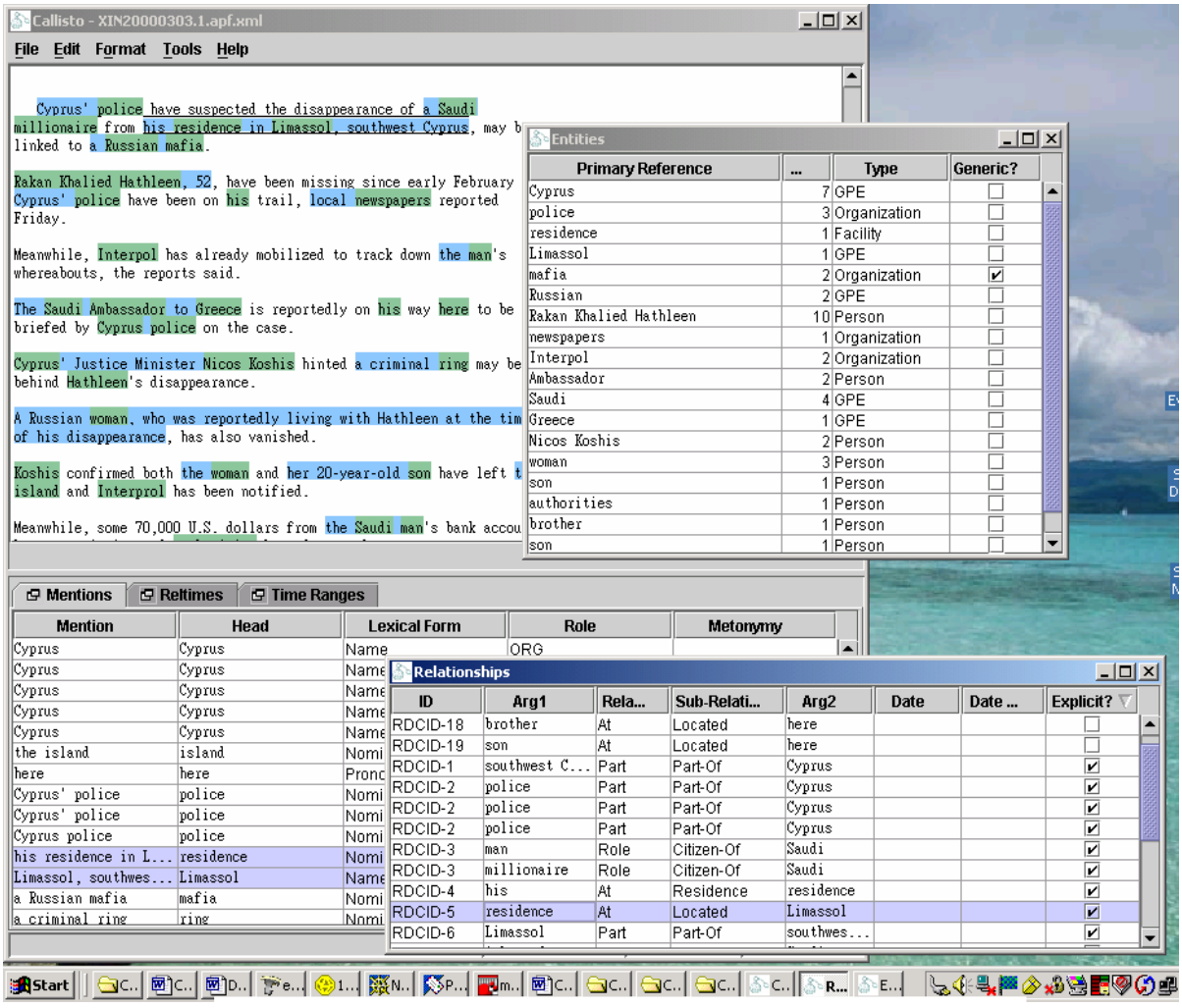

Figure 1. The ACE EDT+RDC task being annotated in Callisto.

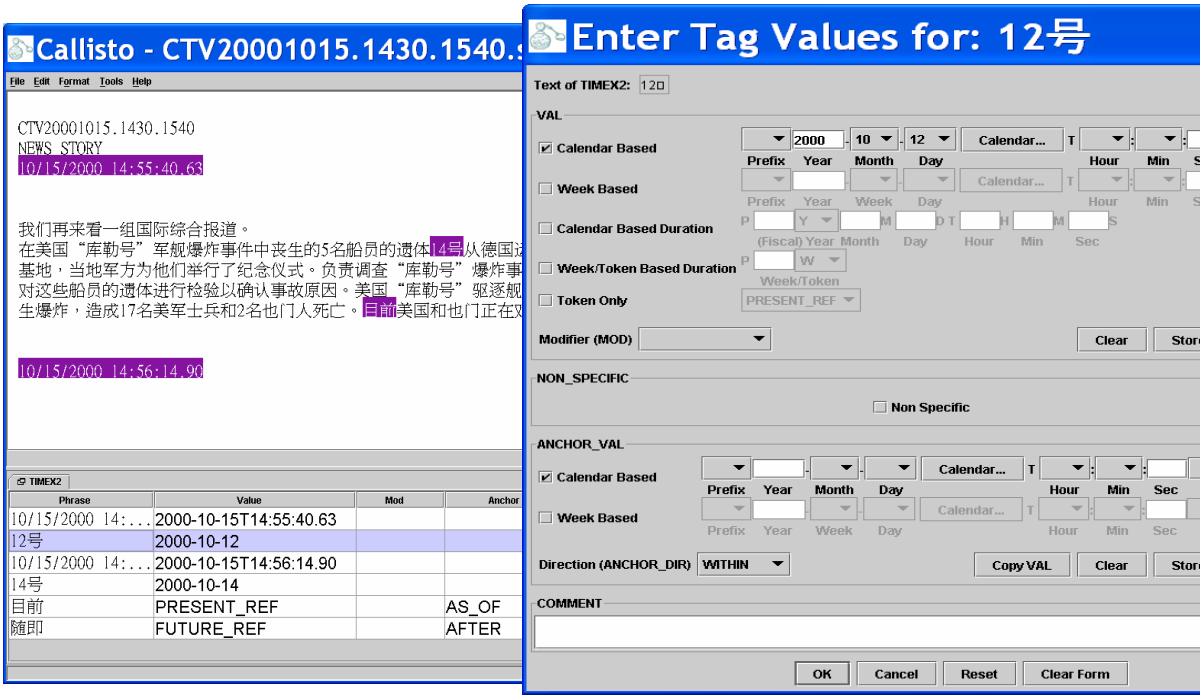

Figure 1. The Timex2 task in Callisto. (a) On the left is the underlying text pane containing the temporal expressions. (b) On the right is the pop-up time expression normalization widget.

initiated both from the tabular presentation panes as well as directly from the document text window on which many of the annotations are anchored.

Timex2 Temporal Expressions Annotation

The Timex2 task guidelines (Ferro, et. al., 2003) define an annotation scheme for associating temporal expressions with a normalized representation of the times they denote. This work has been carried out in support of a number of research activities, including the Translingual Information Detection, Extraction, and Summarization research program (TIDES) and the ACE program (Doddington, 2003).

A Callisto task specification, including a specialized attribute entry/modification widget, has been developed to facilitate the annotation of temporal data according to these standards. Figure 2(a) shows a Chinese text being annotated for temporal expressions. Figure 2(b) shows the Timex2 attributes widget. In the main window, the temporal annotations are listed in table format along with their attributes. The simple true/false attributes can be modified by using the checkboxes in the table. To enter or modify other attributes, the specialized pop-up widget (on the right) is used. The Timex2 widget allows date and time values to be entered and later reviewed in a straightforward human-readable format, or chosen from a calendar, for automatic conversion into the ISO standard format. The widget also supports all other aspects of the Timex2 standard, including the ability to enter a modification to the value, the ability to indicate if the value is non-specific or part of a set, the ability to specify an anchor value and direction, and the ability to include an arbitrary text comment.

Multilingual Texts

Callisto inherits many desirable multilingual capabilities from its use of Java, such as Java's excellent handling of Unicode character encodings and fonts, bi-directional text rendering, etc. Callisto also supports word- or character-based tokenization when selecting text. Callisto incorporates a heuristic method to automatically determine the best font to use (such as for multi-lingual texts); this can be manually overridden by the user. Callisto has already been used to view and annotate data for various tasks in English, Chinese, Spanish, Arabic, and Hindi.

Annotating Multimodal Signals

Callisto is built on the abstract annotation model "Architecture and Tools for Linguistic Analysis Systems," or "ATLAS" (Bird, et al, LREC, 2000), to provide an abstraction over the diversity of possible linguistic signals and their annotations. Callisto uses the jATLAS package (developed by Christophe LaPrun of NIST) that implements much of this specification. The ATLAS formalism enables extensions not only within the sphere of "annotation content" models (that is, what is being asserted about a particular portion of a linguistic signal), but also in the types of signals with which these annotations can be associated, and an extensible mechanism for defining anchors and regions into those signals. This provides a foundation for extending the Callisto services into broader multi-media applicability in the future. To be compatible with arbitrary signal types, Callisto (via jATLAS) makes use of a native XML-based stand-off annotation format, called ATLAS Interchange

Format (AIF). Callisto also supports the import and export of a number of other XML and SGML interchange formats, and provides hooks for the development of additional importers and exporters as needed. Some of these other supported interchange formats currently include: MUC-style in-line SGML (Grishman, 1995), ACE Pilot Format (APF, an XML stand-off annotation that was a precursor to AIF) (NIST 2001), and TimeML in-line XML markup (Pustejovsky, et al, 2003).

Managing Character Offsets and Signals

While stand-off annotation models, such as TIPSTER, GATE (Cunningham, 2002), and the AIF/Atlas model used by Callisto, provide generality for dealing with a range of multi-modal signals, the separation of annotations from signals introduces the potential for version control problems. In addition to having to maintain the mapping (and availability) of two or more files for each annotation task, text files will often have some of their data (newlines, quote characters, etc.) converted inadvertently as they are passed from one application to another. To prevent this headache, and to enable an easier method for transferring standoff annotations along with their signals, Callisto has enriched the AIF standoff annotations with an embedded copy of the text signal itself, encoded in the relatively impervious base64 encoding scheme. In this way a single copy of the Atlas Interchange Format file is sufficient as a fully-encapsulated transfer medium for both the annotations and the original text being annotated.

Summary

Callisto leverages the features of Java to provide a multilingual, multi-platform tool, while its modular design allows a comprehensive suite of customization possibilities. It is built on the ATLAS architecture to promote extensibility and abstraction across the diversity of linguistic signals and their associated annotations. We are actively developing and extending the core capabilities of Callisto, as well as specialized task modules. Plans for Callisto development for the near future include:

- Support for the declarative development of phrase-tagging tasks (without requiring Java programming or parameterization);
- Increased flexibility and reusability of standard GUI library components for text annotation;
- Provision of basic GUI library components for annotation of other modalities of linguistic data; and
- Support for various forms of automated tagging by external software modules.

The plug-and-play task architecture is meant to encourage the independent development and refinement of task-specific annotation configurations on top of a stable and useful set of annotation services. Callisto and its task modules are freely available for downloading at the Callisto web site: http://callisto.mitre.org/. (The current web site requires a registration procedure to obtain an account and password, but we expect to remove this requirement soon.) Documentation of the task interface for developers is expected to be available in the next few months.

References

ACE-EDT (2003). "Annotation Guidelines for Entity Detection and Tracking (EDT), Version 2.5," http://www.ldc.upenn.edu/Projects/ACE/docs/EDT-Guidelines-V2-5.pdf.

ACE-RDC (2003). "Annotation Guidelines for Relation Detection and Characterization (RDC), Version 3.6," http://www.ldc.upenn.edu/Projects/ACE/docs/RDC-Guidelines-V3-6.pdf.

Bird, S., D. Day, J. Garofolo, J. Henderson, C. Laprun, and Mark Liberman, (2000). "ATLAS: A flexible and extensible architecture for linguistic annotation." Proceedings of the Second International Conference on Language Resources and Evaluation (LREC), pp. 1699-1706, Paris: European Language Resources Association.

Cunningham, H., D. Maynard, K. Bontcheva, and V. Tablan (2002). GATE: A framework and graphical development environment for robust nlp tools and applications. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics.

Doddington, George (2003). ACE: Automatic Content Extraction. http://www.nist.gov/speech/tests/ace/ace-tides00/index.htm.

Ferro, L., Gerber, L., Mani, I., Sundheim, B., and Wilson, G. (2003). TIDES 2003 Standard for the Annotation of Temporal Expressions. Technical Report MTR 01W0000041, The MITRE Corporation. (Also available at http://time2.mitre.org/)

Grishman (1995) TIPSTER phase II architecture design. http://cs.nyu.edu/cs/faculty/grishman/tipster.html

Pustejovsky, J., I. Mani, L. Bélanger, B. Boguraev, B. Knippen, J. Littman, Anna Rumshisky, A. See, S. Symonen, J. Van Guilder, L. Van Guilder, M. Verhagen (2003) Final Report of the ARDA Summer Workshop on Graphical Annotation Toolkit for TimeML. http://nrrc.mitre.org/NRRC/TangoFinalReport.pdf.

TIDES (2003). The TIDES Research Program. http://www.darpa.mil/iao/TIDES.htm.

Acknowledgements: This work was performed in support of the Northeast Regional Research Center (NRRC) which is sponsored by the Advanced Research and Development Activity in Information Technology (ARDA), a U.S. Government entity which sponsors and promotes research of import to the Intelligence Community which includes but is not limited to the CIA, DIA, NSA, NIMA, and NRO.