Knowtator: A plug-in for creating training and evaluation data sets for Biomedical Natural Language systems

Philip V. Ogren

Division of Biomedical Informatics
Mayo Clinic College of Medicine
Rochester, MN, USA
Ogren.Philip@mayo.edu

Abstract

A general-purpose text annotation tool called Knowtator is introduced. Knowtator facilitates the manual creation of annotated corpora that can be used for evaluating or training a variety of natural language processing systems. Building on the strengths of the Protégé knowledge representation system, Knowtator has been developed as a Protégé plug-in that leverages Protégé's knowledge representation capabilities to specify annotation schemas. Knowtator's unique advantage over other annotation tools is the ease with which complex annotation schemas (e.g. schemas which have constrained relationships between annotation types) can be defined and incorporated into use. Knowtator is available under the Mozilla Public License 1.1 at http://bionlp.sourceforge.net/Knowtator.

1 Introduction

Knowtator is a general-purpose text annotation tool for creating annotated corpora suitable for evaluating or training Natural Language Processing (NLP) systems. Such corpora consist of texts (e.g. documents, abstracts, or sentences) and *annotations* that associate structured information (e.g. POS tags, named entities, shallow parses) with extents of the texts. The annotations correspond to a gold standard data set that can be used to directly compare against NLP system output. Alternatively, the annotations can be used as input to an NLP system's training algorithms. An *annotation schema* is a specification of the kinds of annotations that can be created. Knowtator provides a very flexible mechanism for defining annotation schemas. This allows it to be employed for a large variety of corpus annotation tasks.

Knowtator has been implemented as a Protégé plug-in and runs in the Protégé environment. In Knowtator, an annotation schema is defined with Protégé class, instance, slot, and facet definitions using the Protégé knowledge-base editing functionality. The defined annotation schema can then be applied to a text annotation task without having to write any task specific software or edit specialized configuration files. Annotation schemas in Knowtator can model both semantic (e.g. protein-protein interactions) and linguistic phenomena (e.g. coreference resolution).

2 Related work

There exists a plethora of manual text annotation tools for creating annotated corpora. While it has been common for individual research groups to build customized annotation tools for their specific annotation tasks, several text annotation tools have emerged in the last few years that can be employed to accomplish

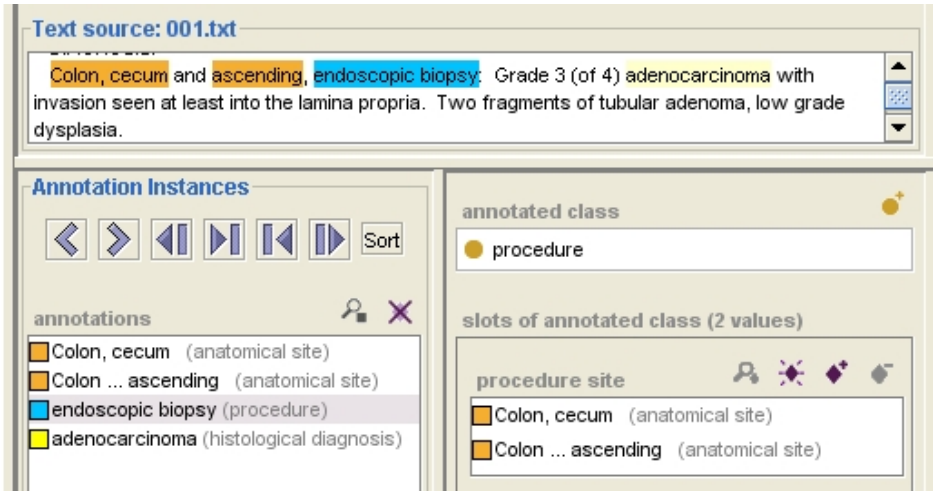

Figure 1 Sample annotations capturing mentions of diagnostic procedure for colorectal cancer.

a wide variety of annotation tasks. Some of the better general-purpose annotation tools include Callisto[1], WordFreak[2], GATE[3], and MMAX2[4]. Each of these tools is distributed with a limited number of annotation tasks that can be used 'out of the box.' Many of the tasks that are provided can be customized to a limited extent to suit the requirements of a user's annotation task via configuration files. In Callisto, for example, a simple annotation schema can be defined with an XML DTD that allows the creation of an annotation schema that is essentially a tag set augmented with simple (e.g. string) attributes for each tag. In addition to configuration files, WordFreak provides a plug-in architecture for creating task specific code modules that can be integrated into the user interface.

A complex annotation schema might include hierarchical relationships between annotation types and constrained relationships between the types. Creating such an annotation schema can be a formidable challenge for the available tools either because configuration options are too limiting or because implementing a new plug-in is too expensive or time consuming.

3 Implementation

3.1 Annotation schema

Knowtator approaches the definition of an annotation schema as a knowledge engineering task by leveraging Protégé's strengths as a knowledge-base editor. Protégé has user interface components for defining class, instance, slot, and facet frames. A Knowtator annotation schema is created by defining frames using these user interface components as a knowledge engineer would when creating a conceptual model of some domain. For Knowtator the frame definitions model the phenomena that the annotation task seeks to capture. An annotation schema consists primarily of a target ontology (see §3.2.1 below) but also contains additional configuration information such as the annotator's names and class color assignments.

As a simple (and hypothetical) example, an annotation schema that captures mentions of diagnostic procedures for the colorectal cancer domain could be modeled with the classes *procedure* and *anatomical site*. A slot called *procedure site* is constrained to be an instance of *anatomical site* and captures the anatomical location where the procedure took place. Annotations in Knowtator using this simple schema are

[1] http://callisto.mitre.org
[2] http://wordfreak.sourceforge.net
[3] http://gate.ac.uk/. GATE is a software architecture for NLP that has, as one of its many components, manual text annotation functionality.
[4] http://mmax.eml-research.de/.

shown in Figure 1. An annotation for the text 'endoscopic biopsy' is associated with the class *procedure* and is the selected annotation in Figure 1.

A key strength of Knowtator is its ability to relate annotations to each other via the slot definitions of the corresponding annotated classes. In the example the *procedure site* slot relates the *procedure* annotation with the annotations corresponding to the class *anatomical site* for the texts 'Colon, cecum' and 'Colon … ascending.' The constraint on the slot ensures that the values of the *procedure site* slot are filled with annotations corresponding to *anatomical sites.*

Protégé is capable of representing much more sophisticated and complex conceptual models which can be used, in turn, by Knowtator for text annotation.

3.2 Knowledge Model

The Knowtator data model consists of three components: 1) an ontology that defines the concepts that are the target of the annotation task 2) a set of instances that capture assertions about relations between and properties of concepts mentioned in text, and 3) a mapping between the target text and members of 2.

3.2.1 Target Ontology

The set of class, instance, slot, and facet frames that define the set of named entities and relations that are the subject of the annotation task is called a *target ontology* and is roughly equivalent to an annotation schema (described in §3.1 above.) A target ontology models the semantic and/or linguistic phenomena that are found in or described by the target texts. The annotation schema has no dependencies on any Knowtator specific class definitions (e.g. classes in the target ontology do not inherit slots from Knowtator specific classes.) The target ontology provides a conceptual model of the target domain and could be used independently of Knowtator.

3.2.2 Concept Mentions

Human language is not constrained to conform to a formal knowledge model regardless of how carefully constructed it[5] might be. Rather than assume that target texts will conform to a pre-defined model of a domain, the Knowtator data model provides a mechanism to describe mentions of concepts with respect to a knowledge model. The term *concept mention* is defined as a description of a concept that has been found in the target text. A *concept mention* provides a means for describing how a concept (e.g. a class or instance frame) is being discussed in text and the assertions that can be drawn from the text about that concept (e.g. relationships between other mentioned concepts or properties of the concept). While Knowtator utilizes the annotation schema to help the human annotator make consistent annotations, it is not constrained by the data model to create annotations that strictly adhere to the target ontology with respect to the constraints that have been defined therein.

3.2.3 Annotations

Annotations provide a mapping between the target texts and concept mentions that have been created. An annotation consists of a concept mention, a span (or spans) of text, and other book keeping information such as the annotator that created the annotation, the date the annotation was created, and a pointer to the target text. The annotation definition is customizable so that additional book keeping information can be captured if desired.

For NLP tasks such as named entity recognition and relationship extraction the Knowtator data model provides a clean separation between the concepts that are being talked about and how they are being talked about. However, for modeling linguistic phenomena (e.g. a deep parse) this clean separation may

[5] Ambiguous anaphora intentional.

not always make sense (i.e. the extent of text in question may actually be an instance of noun phrase rather than a mention of one.) Knowledge engineering issues such as these can be addressed by Knowtator to some extent but such discussion is outside the scope of this extended abstract.

3.3 Features

In addition to its flexible annotation schema definition capabilities, Knowtator has many other features that are useful for executing text annotation projects. A consensus set creation mode allows one to create a gold standard using annotations from multiple annotators. First, annotations from multiple annotators are aggregated into a single Knowtator annotation project. Annotations that represent agreement between the annotators are consolidated such that the focus of further human review is on disagreements between annotators.

Inter-annotator agreement (IAA) metrics provide descriptive reports of consistency between two or more annotators. Several different match criteria (i.e. what counts as agreement between multiple annotations) have been implemented. Each gives a different perspective on how well annotators agree with each other and can be useful for uncovering systematic differences. IAA can also be calculated for selected annotation types giving very fine grained analysis data.

Knowtator provides a pluggable infrastructure for handling different kinds of text source types. By implementing a simple interface, one can annotate any kind of text (e.g. from xml or a relational database) with a modest amount of coding.

Knowtator provides stand-off annotation such that the original text that is being annotated is not modified. Annotation data can be exported/imported to/from a simple XML format.

Annotation filters can be used to view a subset of available annotations. This may be important if, for example, viewing only named entity annotations is desired in an annotation project that also contains many part-of-speech annotations. Filters are also used to focus IAA analysis and the export of annotations to XML.

Knowtator can be run as a stand-alone system (e.g. on a laptop) without a network connection. For increased scalability, Knowtator can be used with a relational database backend (via JDBC).

Knowtator and Protégé are provided under the Mozilla Public License 1.1 and are freely available with source code at http://bionlp.sourceforge.net/ Knowtator and http://protege.stanford.edu, respectively. Both applications are implemented in the Java programming language and have been successfully deployed and used in the Windows, MacOS, and Linux environments.

4 Conclusion

Knowtator has been developed to leverage the knowledge representation and editing capabilities of the Protégé system. By modeling syntactic and/or semantic phenomena using Protégé frames, a wide variety of annotation schemas can be defined and used for annotating text. New annotation tasks can be created without writing new software or creating specialized configuration files. Knowtator also provides additional features that make it useful for real-world multi-person annotation tasks.