Anafora: A Web-based General Purpose Annotation Tool

Wei-Te Chen
Department of Computer Science
University of Colorado at Boulder
weite.chen@colorado.edu

Will Styler
Department of Linguistics
University of Colorado at Boulder
william.styler@colorado.edu

Abstract

Anafora is a newly-developed open source web-based text annotation tool built to be lightweight, flexible, easy to use and capable of annotating with a variety of schemas, simple and complex. Anafora allows secure web-based annotation of any plaintext file with both spanned (e.g. named entity or markable) and relation annotations, as well as adjudication for both types of annotation. Anafora offers automatic set assignment and progress-tracking, centralized and human-editable XML annotation schemas, and file-based storage and organization of data in a human-readable single-file XML format.

1 Introduction

Anafora[1] is a new annotation tool designed to be a lightweight, flexible annotation solution which is easy to deploy for large and small projects. Previous tools (such as Protege/Knowtator (Ogren, 2006) or eHost) have been written primarily with local annotation in mind, running as native, local applications and reading complex file or folder structures. This limits cross-platform deployment and requires the annotated data to be stored locally on machines or run in X-Windows, complicating data-use agreements and increasing data fragmentation.

Anafora was designed as a web-based tool to avoid this issue, allowing multiple anntators to access data remotely from a single instance running on a remote server. Designed for WebKit-based browsers, annotators can work from nearly any modern OS, and no installation, local storage, or SSH logins are required. All data is regularly autosaved, and annotations are saved to cache for restoration in the event of a connectivity interruption.

In addition, avoiding the complex schemas and filetypes associated with many current solutions, Anafora was built to maintain simple, organized representations of the data it generates. Annotation schemas and stored data are both saved as human-readable XML, and these are stored alongside plain-text annotated files in a simple, database-free, static filesystem. This allows easy automated assignment and organization of sets and offers ease of administration and oversight unmatched by other tools.

Most importantly, though, Anafora has been designed to offer an efficient and learnable means for annotation and adjudication using even complex schemas and multi-step workflows (such as UMLS (medical named entity tags), Coreference, and THYME Temporal Relations annotation, described in (Albright et al., 2013)). This allows Anafora to a single-source solution for whole-text annotation across all of your projects.

2 Comparisons with existing tools

Anafora has been designed from the ground up with some key advantages over existing whole-text annotation solutions such as eHost/Chartreader (South et al., 2012), Protege/Knowtator (Ogren, 2006), and BRAT (Stenetorp et al., 2012).

Both Protege and eHost are locally-run Java software (although eHost also relies on a remote in-

[1]Anafora is free and open-source, and is available (along with documentation and sample projects) for public use on *https://github.com/weitechen/anafora*

stall of Chartreader). Although they are available for all major platforms, they require annotators to install the applications locally and upgrade the installations as major issues come up. More importantly, both store the texts being annotated locally on the machine used for annotation, which is problematic under many data-use agreements for medical or otherwise sensitive data. Anafora addresses this shortcoming by its web-based design, allowing easy software update and eliminating local data storage, while also enabling automatic and centralized set assignment.

Another of Anafora's strengths over existing tools is flexibility and complex schema support. At last review, eHost/Chartreader offered only rudimentary between-annotation relations (primarily for co-reference), lacking the flexibility needed for more complex relation sets. BRAT does offer an effective relation annotation tool, but doesn't support the more complex schemas and property types that Anafora does (e.g. multi-slot relations, relation properties, pointers as properties of entities, etc). So, although both BRAT and eHost/Chartreader are excellent solutions for simple annotation schemas, for complex schemas and workflows, Anafora is a more flexible and capable choice.

Finally, Anafora's biggest strength is its lightweight implementation. Unlike Protege/Knowator's folder model where each assigned annotation task contains a separate copy of the schema, document, and project, Anafora's folders-containing-XML model of document and schema storage means that each document and schema is stored only once in one easily accessible place, and unlike eHost/Chartreader, administration can be done by changing and moving files from SFTP or a command line, rather than by logging in to a separate program. This central storage means that schema modification is as easy as changing one XML file, which will be used for all subsequent annotations, and the file-based model eliminates the need to back up large databases.

In short, although many annotation tools exist, Anafora's combination of light weight, web-based UI, centralized file storage and complex schema support make Anafora unique and an excellent choice for any annotation project.

```
<definition>
 <entities type="TemporalEntities" color="015367">
  <entity type="EVENT" color="00ccff" hotkey="e">
   <properties>
    <property type="DocTimeRel" input="choice">
     ,BEFORE,OVERLAP,AFTER,BEFORE/OVERLAP</property>
    <property type="Type" input="choice">
     N/A,ASPECTUAL</property>
    ......</properties></entity>
  .......</entities>
 <relations type="TemporalRelations" color="0000ff">
  <relation type="TLINK" color="0000ff" hotkey="l">
   <properties>
    <property type="Source" input="list"
     maxlink="1" instanceOf="EVENT,TIMEX3"/>
    <property type="Target" input="list"
     maxlink="1" instanceOf="EVENT,TIMEX3"/>
   ......</properties></relation>
  ......</relations></definition>
```

Figure 1: Anafora Schema Example

3 Schema and Data Format

In Anafora, annotations are divided into two types: *Entity* and *Relation*. An *Entity* annotation associates a certain span in the text with a type and list of properties. *Relation* annotations specify a relationship between multiple *Entities*. The nature of these *Entity* and *Relation* annotations (as well as their properties) are stored in an XML Schema file, while data files store the *Entities* and *Relations* specified by annotators for each file.

3.1 Schema

The schema file defines the data type and attributes of the annotations. Our schema format is defined in XML form, which is a simple and human-readable markup file. A Temporal schema is shown in Fig. 1.

The first part of the schema file is "defaultattribute" element in which the schema's overall attributes are defined. Another part is the "definition" element which defines the hierarchy of the schema tree, the annotation types, and the associated properties for each type. The schema is in the form of a tree structure. The "entities" and "relations" tags represent subgroupings of annotation types, while the "entity" and "relation" tags each define a different *Entity* or *Relation* annotation. The "type" attribute defines the name of the annotation type, the "color" attribute defines the displayed color of this annotation in the Anafora interface, and the "hotkey" attribute is the key which triggers creation of that an-

```
<entity>
 <id>5@e@ID020_clinic_058@gold</id>
 <span>1328,1342</span>
 <type>EVENT</type>
 <parentsType>TemporalEntities</parentsType>
 <properties>
  <DocTimeRel>BEFORE</DocTimeRel>
  <Type>N/A</Type>
  ......</properties></entity>
......
<relation>
 <id>2@r@ID020_clinic_058@gold</id>
 <type>TLINK</type>
 <parentsType>TemporalRelations</parentsType>
  <properties>
  <Source>5@e@ID020_clinic_058@gold</Source>
  <Target>7@e@ID020_clinic_058@gold</Target>
  <Type>CONTAINS</Type>
 </properties></relation>
```

Figure 2: Anafora Data File Example

notation in Anafora.

For each type, the properties to be annotated are listed under "Property", where the "type" attribute indicates the name of the property, while the "input" attribute specifies the manner of attribute selection or entry. The value of the "Property" is a list of accepted choices. For example, the "Type" property in the "Event" entity limits the value to "N/A" or "ASPECTUAL", where "N/A" is the default. Please refer to the Guidelines for further detail.

One great advantage of this XML-based schema format is greater flexibility than existing tools both in schema and modification. To make any modification to the schema, one simply edits the XML and the revised schema will apply to any new data files. Another advantage is human-readability, allowing schemas to be easily understood and ported from other tools.

3.2 Data File

The Anafora data file (see Fig. 2) stores the annotation instances for each annotated file. It, like the Schema file, uses an XML format.

The "info" section provides the basic information for the file, such as the save time and annotation completion status. The "schema" tag specifies the path to the schema file used for annotation. Following is the "annotation." Each "entity" and "relation" element represents an annotation instance. Every annotation has a unique "id", and the annotation "type" and "parentType". For *Entity* annotations, the text's span is given using character offsets in the source file. For all annotations, the "property" section specifies the values for properties listed in the schema, and, for *Relations*, properties are used ("Source" and "Target" above) to point to the unique IDs of the annotations being linked.

4 System Overview

Anafora is a web-based tool, developed using Django (a Python framework on server side) and jQuery (a JavaScript library on client side). On the server side, our system manages file storage and user access control. By avoiding the use of any database, Anafora is very agile and flexible, and most of the computing work is executed by the user's browser. And, because modern browsers have done an excellent job tuning JavaScript execution, Anafora is lightweight on the user's machine as well. Anafora's browser-based design also allows the tool to run well on any OS with a web browser, alleviating the cross-platform issues common with other tools.

Anafora allows both keyboard- and mouse-based annotation, improving both efficiency and immediate familiarity, rather than relying primarily on mouse-clicks.

Anafora also assists project supervisors in several ways. First, all data management is file-based, and the project hierarchy is reflected in the OS level file system's directory structure. Secondly, Anafora assigns tasks to annotators automatically, saving supervisors the time and hassle of task assignment. Finally, Anafora makes pre-annotation extremely easy. By running the document text through a shallow parser and generating a file which marks all noun phrases (for example), annotators could start their work on a named entity task with this information ready at hand.

Anafora allows users to customize their own user interface by overriding the CSS file on the client side. By changing the CSS file, users can modify the apperance, e.g., color, font, and page layout.

5 Project Creation and Administration

Administering and creating new projects is straightforward, and primarily file based. To create a new project, one first uses our schema markup to write an XML schema designating the entities, relations, and annotation properties needed in the schema (see Section 3). Then, a folder is created for the project, containing folders for any subcorpora, and then finally each document is placed into its own folder as a plaintext file. At this point, annotators with the necessary permissions may select the new schema and documents and begin annotating.

A given document's assignments and completion status is read entirely from the filenames generated by the program. To re-assign a set manually, simply change the annotator's name in the existing annotation file's name, or delete the previous annotation file, allowing Anafora to reassign it automatically. Administrators can view any annotator's work through the tool's interface, and can edit the XML at any time. When a document is fully annotated or adjudicated, preparing for release is as easy as copying the .gold. file and source text into the final destination.

6 Annotation using Anafora

When an annotator opens Anafora in any webkit-based browser and logs in, they are greeted with a file-choosing interface which allows them to pick which corpus, annotation schema and annotation type (*Entity* or *Relation*) they'd like to work on for the session (allowing one annotator to easily work with more than one project or schema). Previously completed and in-progress sets are shown in separate columns for easy access, and only documents which have fewer than the maximum number of annotators are displayed. Annotators are not able to open or view any annotations other than their own.

Once a document is opened, the annotator is presented with Anafora's 3-pane view (in Fig. 3): on the left, the annotation schema, in the center, the source text, and on the right, annotation details. To proceed with an *Entity* annotation, the annotator selects a word or portion of text and hits a pre-defined hotkey, triggering the creation of a new annotation of a specified type, using the selected span.

The properties of the annotation are then automatically filled in with the default values specified in the schema files, and the annotator can then go back in to modify these properties (by drop-down menu, radio buttons, relation or free-text entry) as needed. The annotator can also use the span editing tools to either modify the span character-by-character, or to add a second, disjoint span by selecting more text and using the "+" button.

For *Relation* annotation, the annotator will enable the Relation grid, displaying a list of relations in order of occurrence in the text. To create a new relation, the annotator strikes the proper hotkey, and then Anafora hides all entities which are not allowed to fill slots in this relation. Clicking an entity after pressing "1" fills the first slot, and pressing "2" before a click fills the second slot. As with *Entity* annotations, properties are filled in according to default values in the schema and can be edited as needed.

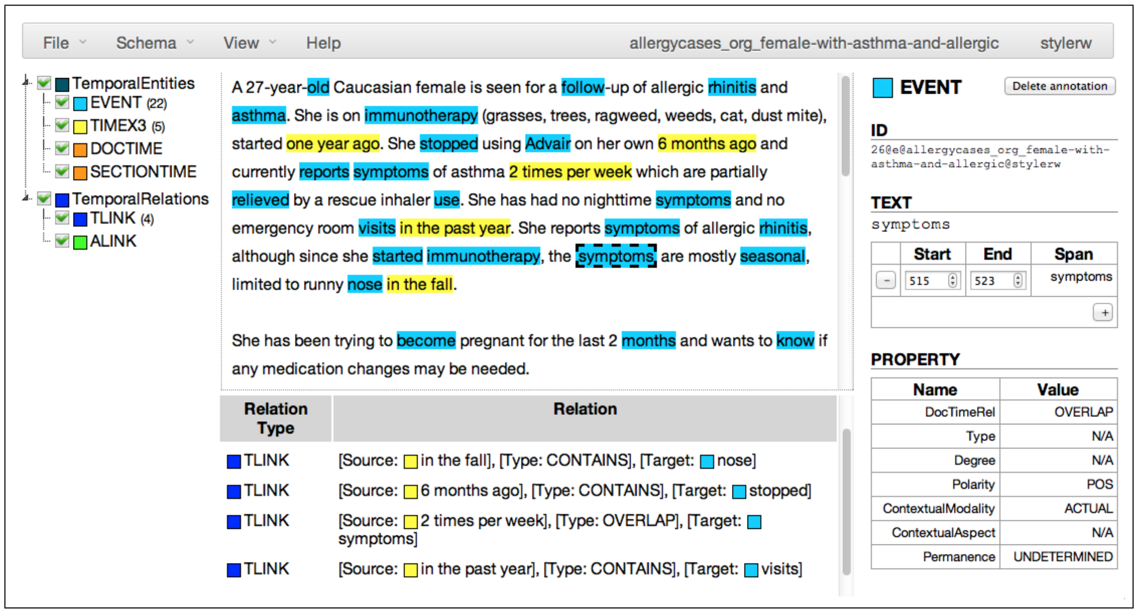
Annotators can choose to manually save and log out at any point, and when an annotator has completed a document, he or she selects "Mark as Completed", which changes the file's status and queues it up for adjudication.

6.1 Adjudication

When a designated adjudicator logs into Anafora, they're presented with the "Adjudication" annotation type option in the initial document selection screen. When this is selected, only documents with two completed annotator-copies are displayed as available for adjudication.

Once an available document is opened, Anafora will automatically merge the two annotators' work into a new, adjudication datafile (preserving the separate annotations), and then mark as gold any annotations matching for both span and properties. In addition, Anafora will mark as conflicting any annotation pairs with either 1) matching properties and overlapping spans or 2) identical spans and different properties. Anafora then displays the schema and source documents as before along with two annotation detail panes, one for each annotator in a conflicting annotation. A progress bar displays the number of gold annotations out of the total number in the document, and again, progress is automatically saved.

The adjudicator can then use the keyboard to move through the unadjudicated (non-Gold) annotations. When an annotation with a conflict is found, details about both annotations will show up on the right, highlighting in red any areas which differ (span, a property, etc). The adjudicator can then use the arrow keys to select either the left



A 27-year-old Caucasian female is seen for a follow-up of allergic rhinitis and asthma. She is on immunotherapy (grasses, trees, ragweed, weeds, cat, dust mite), started one year ago. She stopped using Advair on her own 6 months ago and currently reports symptoms of asthma 2 times per week which are partially relieved by a rescue inhaler use. She has had no nighttime symptoms and no emergency room visits in the past year. She reports symptoms of allergic rhinitis, although since she started immunotherapy, the symptoms are mostly seasonal, limited to runny nose in the fall.

She has been trying to become pregnant for the last 2 months and wants to know if any medication changes may be needed.

Relation Type	Relation
TLINK	[Source: in the fall], [Type: CONTAINS], [Target: nose]
TLINK	[Source: 6 months ago], [Type: CONTAINS], [Target: stopped]
TLINK	[Source: 2 times per week], [Type: OVERLAP], [Target: symptoms]
TLINK	[Source: in the past year], [Type: CONTAINS], [Target: visits]

ID
26@e@allergycases_org_female-with-asthma-and-allergic@stylerw

TEXT
symptoms

	Start	End	Span
-	515	523	symptoms

PROPERTY

Name	Value
DocTimeRel	OVERLAP
Type	N/A
Degree	N/A
Polarity	POS
ContextualModality	ACTUAL
ContextualAspect	N/A
Permanence	UNDETERMINED

Figure 3: Anafora Annotation Window

or right annotation as Gold, which will delete the other. For single-annotator annotations, the adjudicator can choose to simply delete or mark as Gold.

Once no unadjudicated annotations remain in the document and any necessary edits or additions are made, the adjudicator can mark the document as completed, which changes all annotations' status to "Gold" and, where required, makes the document available to the next round of annotation.

7 Conclusion and Future Work

Anafora can be extended readily to offer other classification tasks such as part-of-speech tags or sense tags. However, there are a couple of limitations. First, tree-based annotation, much like constituent-based semantic role labeling, is not currently supported in Anafora. Additional text information (e.g. Frame files and WordNet ontologies) is difficult to display in the same page as the annotations, as the tool was designed for whole-text annotation. Some complicated schema definitions, such as relations (or relation properties) linking to relations, are also not provided.

We are continuing active development (focusing on annotation efficiency and UX design) as more projects with varied needs use Anafora. Performance studies and comparisons are currently in progress. Furthermore, an administrator interface, including annotator management, task status management, and schema editor, will be supplied. In addition, automated pre-annotation is being incorporated into Anafora-based workflows. We will also allow comparison of annotators' work to extracted annotation characteristics from gold data and from each annotator's prior work. We would also like to include active learning and allow annotators to compare their completed annotations to gold standard data. These features should help to improve the learning and annotation efficiency of the annotators.

Anafora is a lightweight and efficient tool for text annotation, easily adaptable to fit even the most complex of annotation tasks and schemas. Source code is available at our GitHub page, *https://github.com/weitechen/anafora*.

Acknowledgments

The development of this annotation tool was supported by award numbers NLM R0110090 (THYME) and 90TR002 (SHARP), as well as DARPA FA8750-09-C-0179 (via BBN) Machine Reading. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NLM/NIH or DARPA. We would also like to especially thank Jinho Choi for his input on the data format, schemas, and UI/UX.

References

Daniel Albright, Arrick Lanfranchi, Anwen Fredriksen, William Styler, Collin Warner, Jena Hwang, Jinho Choi, Dmitriy Dligach, Rodney Nielsen, James Martin, Wayne Ward, Martha Palmer, and Guergana Savova. 2013. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of the American Medical Informatics Association. 2013;0:1-9. doi: 10.1136/amiajnl-2012-001317*.

Philip V. Ogren. 2006. Knowtator: A protégé plug-in for annotated corpus construction. In *Proceedings of the NAACL-HLT, Companion Volume: Demonstrations*, pages 273–275, New York City, USA, June. Association for Computational Linguistics.

Brett R. South, Shuying Shen, Jianwei Leng, Tyler B. Forbush, Scott L. DuVall, and Wendy W. Chapman. 2012. A prototype tool set to support machine-assisted annotation. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, BioNLP '12, pages 130–139, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at EACL-2012*, pages 102–107, Avignon, France, April. Association for Computational Linguistics.