Anafora

Documentation and User's Manual

Written by Will Styler and Wei-te Chen

The software, as well as the most recent version of this document, is always available from:

https://github.com/weitechen/anafora

Last updated: June 9, 2013

Contents

1 About Anafora

Anafora (pronounced /ˌænəˈfɔɹə/, "a-nuh-FOUR-uh") is a new annotation tool written at the University of Colorado designed to be a lightweight, flexible annotation solution which is easy to deploy for large and

small projects. Previous tools (Protege/Knowtator, 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, complicating data-use agreements and increasing data fragmentation. Alternatively, these programs can be run remotely via X-windows, but this increases latency and leaves annotation vulnerable to any connectivity interruptions.

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 (not source text) can be saved to local storage for restoration in the event of a connectivity interruption.

In addition, avoiding the complex formats, schemas, and filetypes associated with current solutions, Anafora was built to provide simple, organized representations of the data required for annotation. Annotation schemas and stored annotation data are both saved as human-readable XML, and these are stored alongside plaintext annotated files in a simple, database-free, static filesystem. This allows easy automated assignment of new sets, pre-made data organization, and ease of administration and oversight unmatched by other tools.

Most importantly, though, Anafora has been designed to offer an efficient and learnable means to annotate and adjudicate using even complex schemas, pipelines and workflows. Designed with complex schemas in mind (featuring spanned and spanless annotations, relations, annotation and pointer properties), Anafora has been built to handle any annotation type, and to be equally at home with multiple steps, passes, or annotation types.

Anafora provides annotation projects, simple or complex, with an easy-to-use single, lightweight and open-source solution to all spanned and spanless annotation needs.

1.1 Software License

Anafora is free and open source software. It has been released under the Apache 2.0 License, the text of which is viewable in full at:

http://www.apache.org/licenses/LICENSE-2.0.html

1.2 Frequently Asked Questions (FAQ)

1. *Can Anafora be installed on my machine and used locally for annotation?*
    - No, at this time we've focused on building a secure and lightweight centralized annotation system rather than a distributed model. In addition, Anafora requires a Linux environment to run. That said, although it will require some technical skill, there's no reason that you can't install Anafora on your linux machine (or in a virtual machine) and then work directly with that instance.
2. *Does Anafora support offline annotation?*
    - No. Anafora requires communication with the central server to save and open annotations, so without any internet connection, you will be unable to begin or save annotation.

      However, Anafora does cache annotations locally, allowing you to continue working in the same document/window during an internet outage. If you've made annotations while no internet was available, when you reconnect and reload, you'll be prompted to save your locally cached annotations to the server. This means that Anafora can work well with a slow, intermittent, or tethered internet connection, well beyond what is feasible with X11 tunneling or completely synchronous web-based tools.

3. *Will Anafora read annotation files from Protégé/Knowtator, Brat, eHost, etc?*
    - Anafora is designed to read AnaforaXML only, however, the format is similar to that used by many other tools, and we have built scripts to convert files from other annotation software (e.g. Protégé/Knowtator) into AnaforaXML in the past.
4. *Does Anafora support overlapping spans?*
    - Yes! Anafora has been designed to show overlapping spans with a double-bordered gray box. Clicking this box will bring up a menu allowing you to choose your desired annotation.
5. *Does Anafora read and display syntax trees?*
    - Anafora reads from plaintext files, and does no parsing or processing of the data within. So, although the plaintext tree files can be read into the program, they will be displayed as they are, and annotators will not be able to interact with nodes or labels beyond selecting spans.

Part I

Anafora for Annotators

2 How Anafora's web-based annotation works

Anafora is a secure, web-based tool. Once properly installed on a remote server, it can be accessed from anywhere with a web browser, and no part of the document being annotated is saved to the local machine.

When you open Anafora and select your document, the relevant text is opened in a browser along with the schema and a pane for properties, for you to annotate.

As you annotate, your annotations (only the numerical spans, annotation IDs, and associated properties) are cached in memory and saved to the server automatically every 2 minutes. In the event that your connection is interrupted, reloading the page will reload your annotations from that cache and save them to the server, meaning that although a consistent internet connection is desirable, an occasional interruption shouldn't result in data loss.

When you finish a session, you'll want to use the "save" command and exit the browser. By doing that, you'll know that your work is safe.

2.1 User-level requirements

In order to run Anafora, you'll need:

- A consistent internet connection (although 100% consistency isn't necessary)
- A modern browser supporting HTML5, Javascript and CSS (Google Chrome is our recommended choice and is the browser we're primarily testing with, but Safari, Chromium, Firefox, and other Webkit-based browsers have been shown to work well as well.)
- An account set up on the server hosting Anafora (which is a member of the requisite groups)
    - For adjudication and access to other annotators' data, you must be a member of the "anaforaadmin" group on the server on which Anafora is installed.

Caution! *No version of Microsoft Internet Explorer is supported by Anafora, and using IE has been demonstrated to break key functionalities in the interface. IE users are encouraged to download Google Chrome or Firefox for annotation (and, frankly, for general browsing use).*

3 Logging in and choosing a document

Logging in to Anafora is as simple as directing your browser to the URL of your specific Anafora installation. You'll log in to the tool and will then be immediately greeted with the file-choosing interface which allows you to pick which corpus, annotation schema and annotation type (e.g. entities, relations, adjudication) you'd like to work on for the session.

First, you'll want to select the annotation type that you're doing under the "Schema" list. So, if you'd like to do Coreference annotation, choose "Coreference", for Temporal, you might choose the "Temporal" schema. This will tell Anafora what sort of annotations you're doing for this session, and sort the document list accordingly. Then, you'll choose the sub-type of annotation ("Entity" to create spanned entities, "Relation" to create relations, "Both" if both happen at once).

Once Anafora knows exactly what annotations you'd like to do, select the project that you're working on, choosing first the higher-level project (usually a grant name or greater project) and then, where applicable, the specific corpus.

Once you've chosen a corpus, three columns will appear. The leftmost column ("New Tasks") lists all documents available for annotation at the moment. The center column ("In-Progress Tasks") lists any documents which you've begun (and thus, claimed), but have not yet marked as completed. The rightmost column ("Completed Tasks") lists prior work that you've done and marked completed, so you can easily go back to reference a prior note.

To begin annotation, choose a document from "New Tasks" or "In-Progress Tasks".

3.1 About Document Assignment

Anafora assigns available documents to annotators automatically. If no documents show up under "New Tasks", this means that no new documents are currently available for annotation. Anafora prevents more than two annotators from working on the same document, and will only assign a document to two annotators. Once two annotators have begun a document (making and saving annotations), that document will no longer show up for other annotators. If you see no new tasks, try a different annotation mode (assuming you've been trained on more than one mode), because it's quite possible that new documents are available for entity annotation, but not relations, for instance. If no new tasks at all are available, contact your supervisor to see if more documents can be added to the system.

Caution! *If you've started a document, it won't be assigned to other annotators, and until you've completed a document and marked it as complete, it won't be able to be reassigned nor adjudicated. Make sure to always remember to mark your documents as complete once you've finished, and try to finish your in-progress documents before starting new ones!*

4 The Anafora Window

Once a document and schema have been opened and selected, the annotator is presented with Anafora's 3-pane view: on the left, the annotation schema, in the center, the source text to be annotated, and on the right, annotation details.

4.1 The Schema Pane

The annotation schema is always present in the left-most pane of the Anafora window, and is organized hierarchically. Clicking the small white triangle next to a high-level category will reveal its child annotation types.

Each annotation type line has four elements. First, a checkbox which hides or reveals that annotation type. Then, a small square indicating the color of these annotations in annotated documents (set in the schema). Then, the name of the annotation type (e.g. "EVENT" or "Markable" or "Disorder". Finally, in parentheses, the number of annotations of that type currently in the document. Clicking the name of an annotation type selects that type for fast annotation (using the spacebar).

The checkboxes next to each annotation type and category, as mentioned, can be used to hide or reveal annotation types individually in the document. To hide or display all types, use *Schema* $\rightarrow$ *Hide/Display All*.

In the annotation schema set up for your use, most annotations have been given a hotkey for easy annotation. This hotkey can be discovered by looking at the schema using the *Schema* menu, the hotkey for each annotation type is in (parentheses) next to the name. Annotations without hotkeys (which are occasionally necessary for large schemas) can be created using the *Schema* menu, or by selecting the annotation type in the left pane and using the Spacebar as a hotkey.

4.2 The Document Pane

The center pane shows the document being annotated, as well as any spanned annotations visible in the document. The document itself cannot be edited in Anafora, only annotated. To increase the font size of the document, simply increase your browser's font size.

The majority of your work will be done here, selecting portions of the text or clicking to edit or link existing annotations.

If, in the document, there are two annotations whose spans overlap, this span overlap area is indicated by a gray box with a double black line. Clicking on this gray box will give a pop-up window allowing you to select which of the two annotations you would like to select.

4.3 The Annotation Details Pane

The rightmost pane in Anafora is the annotation details pane. Blank when no annotations are selected, this pane shows details about the selected (or recently created) annotation.

When an annotation is selected, you'll see the type of the annotation in bold at the top, along with a button to delete the selected annotation. Below that, the annotation's unique ID (containing, among other things, the name of the annotator who originally created it). Below that, for span-based annotations, you'll see the complete text associated with this annotation, disjoint spans separated with '...'.

Below the text is the span editor. This allows you to add, subtract, and modify the spans associated with an annotation. The character offset of each span's start and end is listed and modifiable, along with small arrowbars which allow you to add or subtract characters from the start or end one-by-one. In addition, the

text of each span is included under ”span”. Below the last span, you’ll see a “+” button. This button adds the selected text in the document pane to the selected annotation as a disjoint span. Once this is clicked, the new span will show up in the span editor as well, along with the text which makes it up.

Below that is the Property Editor. The default values, if specified in the schema, for each property will automatically be filled in when the annotation is created. however, by clicking on the ”Value” column for each property, you can edit the properties as needed. Single choice properties will be edited using a drop-down menu, multiple-choice properties using a series of checkboxes. Any properties requiring free text entry will have an editable box in the value column, and a relational property (one which points to another, existing annotation) can simply be clicked, followed by a click on the annotation to which it points.

Some properties may be required, and if you attempt to move to another annotation, the program will remind you to go back and pecify the required property.

When working in adjudication mode, the annotation details pane will show up twice, once for each annotation being compared. in addition, two new options will be available at the top of the annotation details pane, “Mark as gold” and “Delete”. An annotation which has already been marked as gold will have the “Cancel Gold” option in place of “Mark as Gold”.

4.4 The Relation Grid

When working in relations annotation mode, the relation grid will be available to you. To enable it, go to *“View”* → *“Relation Grid”*.

When enabled, the relation grid will show up at the bottom of the document pane, and displays all of the relations present the document, in order of their appearance in the text. Clicking on entries in this relation grid works identically to clicking on entries in the document pane, in that you can select a relation by clicking on it, or even point to a relation from within another annotation by clicking it in the grid.

The relation grid will show the type of the relation, text of the span-ful annotations which make up each relation, as well as any properties of the relation, in the same order that they are specified in the annotation schema itself.

5 Annotating in Anafora

5.1 Creating a new span-based annotation

Once your document is open, and you have identified a word, phrase, or character in need of annotation, the very first step is to select the desired span in the document pane.

When you double-click on a word, most browsers will expand the selection to include the entire word. This can often make annotation quicker, and require less precision mousing to make the proper selection. For partial word annotations, or when one would like to explicitly capture punctuation or numbers, one can select by hand, holding down “Control” (on a PC) or “Option” (on a Mac) to disable the automatic expansion of the selection to the whole word.

Once the proper span is selected, simply hit the hotkey designated for the annotation you need to make. These hotkeys can be found in the *“Schema”* menu. for indications which do not have a hotkey, you can select it from the *“Schema”* menu itself, or, you can select the desired annotation type in the schema pane and simply tap the Spacebar to create an annotation of the selected type..

At this point, the properties of the annotation have been automatically filled in with the default values specified in the schema files (where specified), and are all visible in the annotation details pane. You can then modify these properties (by drop-down menu, radio buttons, relation or free-text entry) to match the desired output.

In addition, you can use the span editing tools to either modify the span character-by-character, or to add a second, disjoint span by selecting more text in the Document pane and using the "+" button.

Assuming that all required properties have been filled in, you can then select more text in the document pane and proceed to creating a new annotation. This newly created annotation can also be deleted using the "Delete Annotation" button.

If, in the document, there are two annotations whose spans overlap, this span overlap area is indicated by a gray box with a double black line. Clicking on this gray box will give a pop-up window allowing you to select which of the two annotations you would like to select.

5.2 Creating a new relation annotation

For relations annotation, you must first enable the Relations grid (*"View"* → *"Relation Grid"*), which will display a list of the relations in the text, in order of occurrence.

To create a new relation, use the applicable hotkey for the type of annotation you'd like to create. Once the hotkey is struck, Anafora hides all span-ful annotations which are not allowed to fill slots in this relation. To fill the first slot in the relation, press "1" then click on an annotation. To fill the second slot, press "2" and click an annotation. To fill in the third property, press 3 to highlight that slot.

For annotations with MaxLink > 1, you'll hit the relevant key for the slot ("1" or "2") and then click as many entities as you'd like to fill that slot (up to the MaxLink value). When finished filling a slot, click another or hit "Escape".

Adding annotations to the relation can also be accomplished by clicking each slot in the properties area and then clicking the annotation in the document pane. As with spanned annotations, properties are filled in according to default values in the schema (if any) and can be edited as needed.

Another way to discover existing relations (useful when annotating) is to click on a span annotation within the document pane. If this annotation is involved in a relation, the relation will be listed in the pop-up window, and can be selected from there as well.

To move onto the next relation, select another one in the relation grid, or hit the proper hotkey to begin the next relation.

5.3 Pausing and Connection Interruptions

While annotating, Anafora is automatically saving your annotations every two minutes, assuming that changes have been made, such that even a complete system crash will only result in a small loss of annotation. in the event that you accidentally navigate away from the webpage, or that your internet is temporarily disconnected, simply reload the Anafora page that you were working on and the program should ask you to restore your annotations from your local storage, assuming that annotations were made that were not saved on the server. this will bring you back to where you were before the unexpected pause.

5.4 Marking a document as completed

Once you've fully annotated the document according to the task, use *"File"* → *"Mark as Completed"*, which marks the file as completed and queues it up for adjudication.

Remember, a file that you've not marked as completed cannot be adjudicated, and will not be reassigned without human intervention. Even if you finished annotating a document completely, until you actually mark the document as completed using this process, no further progress can be made.

6 Adjudicating in Anafora

6.1 About Adjudication Mode

Anafora has a full-featured adjudication mode built in, which uses the same interface and process as annotation. When a person assigned the proper permissions logs into Anafora, they're presented with the "Adjudication" annotation type option in the initial document selection screen. *Only people specified as adjudicators will be able to do adjudication.*

When adjucation mode is selected, only documents with two completed annotator-copies of the required annotation type (relations, spanned annotations) are displayed as available for adjudication. the presence of a document in the "new tasks" list in adjudicator mode means that both annotators have already completed work on it, and it can be freely selected and adjudicated.

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 automatically mark as gold any annotation with identical span and properties across both annotators. Then, the 4-pane adjudication window is opened, displaying the schema and document panes as before, as well as two annotation detail panes, one for each annotator. A progress bar at the bottom of the schema window displays the number of gold annotations out of the total number in the document. As with other types of annotation, progress is automatically saved, and manual saving works as well.

6.2 Adjudicating the Document

Once the document is open, you can then navigate through the document by keyboard, using the "<" and "." keys to move forward and backwards through the unadjudicated (non-Gold) annotations. You can move through all annotations using Control + "<" and Control + "."

Alternately, you can find unadjudicated annotations by hand. Gold-standard spanned annotations will show up differently in the Document pane, with gold entities showing up in a more pastel, muted color relative to the unadjudicated annotations of the same type. Similarly, gold standard relations in the relations grid will be grayed out relative to unadjudicated relations.

Regardless of how you find it, when an annotation marked by only one annotator is encountered, details will show up in one of the detail areas, and you will have the choice to either remove the annotation, modify it, or mark it as gold.

When two similar annotations with a conflict (either span or properties) are found, details about both annotations will show up in the Annotation Details pane, any areas which differ (span, a property, etc) highlighted with a red box. You can then use the Left and Right arrow keys to select either the left or right annotation as Gold, which will automatically delete the other.

Relations our adjudicated in the same way, using the relations grid to find conflicts and marking relations is either gold or deleted.

In addition to adjudicating existing annotations, you can add annotations of his or her own which will be considered gold annotations automatically. annotating and adjudicator mode works identically to annotating in annotation mode.

6.3 Completing Adjudication

Once the adjudicator has marked as gold (or deleted) all unadjudicated annotations in the document (as shown using a progress bar at the bottom of the Schema pane), the adjudicator can mark the document as completed (again *"File"* $\rightarrow$ *"Mark as Completed"*), which changes all annotations' status to "Gold" and, where required, makes the document available to the next round of annotation.

7 Logging Out

To end your annotation or adjudication session, use *“File”* → *“Save”*. The save button will be grayed out if no changes have been made since the last save (automatic or manual). If you attempt to leave the page without saving, the program will prompt you to do so.

Once your annotations are saved, you can use *“File”* → *“Open”* to return to the file chooser, or simply close the browser window to exit Anafora altogether. You will be logged out automatically.

Part II

Anafora for Project Administrators

8 Administering an Anafora Annotation Project

Because Anafora is primarily file-based, administration is primarily a question of arranging and renaming files, allowing adminstration to easily occur over a command line or using SFTP.

8.1 Starting a new Anafora-based Annotation project

To start a new Anafora project, a few steps must be taken:

1. You must install Anafora on a local server, and set up access to the annotation page by password
2. Any users doing administration or adjudication must be made members of the ”AnaforaAdmin” server group
3. You must set up the Anafora Project file (see Section 9) and copy the source documents into their proper locations within the project files
4. You must create Anafora XML versions of any annotation schemas to be used (See Section 10) and copy them into the project file’s .schema folder
5. You must determine the type of pipeline to be used for each annotation schema (See 8.2)

   Once these tasks are completed, it’s a matter of training your annotators and watching the data flow in.

8.2 Different pipelines in Anafora

Anafora supports two different pipelines for a any given annotation schema.

The first is a simple pipeline, combining spanned and relation annotations, with a single adjudication. In this pipeline:

1. Two annotators are assigned a document
2. Both annotators mark spanned annotations as well as relations
3. When both annotators mark the document as complete, the document is assigned to an adjudicator
4. The adjudicator adjudicates the document
5. When marked as complete by the adjudicator, the document is Gold Standard

The second pipeline is more complex for projects where spanned annotations and relations happen at different stages:

1. Two annotators are assigned a document
2. Both annotators mark spanned annotations only
3. When both annotators mark the document as complete, the document is assigned to an adjudicator
4. The adjudicator adjudicates the spanned annotations
5. When marked as complete by the adjudicator, the document is reassigned to annotators for relations annotation, the gold-standard spanned annotations automatically imported.
6. Both annotators mark relations only (unable to modify gold standard annotations)
7. When both annotators mark the document as complete for relations, the document is assigned to an adjudicator
8. The adjudicator adjudicates relations
9. When marked as complete a second time by the adjudicator, the document is Gold Standard

    How is this designated per schema in the program??!!?

9 The Anafora Project File

One of the many benefits of Anafora is that data is kept organized in an intuitive way, without any human intervention. Knowing this Project File's structure (and how to find the data within) is a critical part of adminstrating a project using Anafora.

By default, the project file is organized into three folder levels: Project $\rightarrow$ Corpus $\rightarrow$ Document. The highest level, "project", is designed to separate different large-scale projects from one another and contains the. The "Corpus" level contains one or more corpora (subdivisions of the source data) to be annotated in a given project, and beneath the corpus level, we have individual documents. At the highest level (alongside "Project" folders), Anafora also has the .schema folder, containing the master copies of all the schemas available for use on all annotations.

In order for a document to be added to Anafora, it must be a plain-text file, and must be contained within a folder sharing the same name as the document itself. So, the path to a single document, a plaintext file named "doc0750", would be:

`AnaforaProjectFile/BigProject/ProjectCorpus2/doc0750/doc0750`

9.1 Anafora Filenames

Within the individual document folder, individual annotations will show up as separate XML files alongside the plaintext source text, with the filename format below:

`[textname].[schema].[annotationtype].[annotator].[status].xml`

So, our doc0750, if annotated by stylerw according to the "Temporal" schema for Temporal Entities would have the filename:

`doc0750.temporal.temporal-entity.stylerw.inprogress.xml`

Once the annotator marks it as completed, the name would change to:

`doc0750.temporal.temporal-entity.stylerw.completed.xml`

(Incidentally, if an annotator has forgotten to mark a document as completed, simply changing the filename's "inprogress" to "complete" will do so without logging into the tool.)

The Gold annotation (post adjudication), in the same folder as the others, would have the name:

`doc0750.temporal.temporal-adjudication.gold.completed.xml`

And so forth.

9.2 Final Data Format

A fully annotated document in Anafora will have a different .completed. file for every annotator for each annotation pass/type, as well as .gold. xml files for each annotation type. For example, in a workflow with a ”temporal” annotation task and a ”UMLS” task (each with entity and relations steps) as well as a ”CoRef” task (with simultaneous entity and relation annotation, a folder full of completed annotations for a document called ”doc0750” might look like:

- `doc0750/`
  - `doc0750`
  - `doc0750.coreference.coreference.wtchen.completed.xml`
  - `doc0750.coreference.coreference.jshart.completed.xml`
  - `doc0750.coreference.coreference-adjudication.gold.completed.xml`
  - `doc0750.temporal.temporal-entities.jsmith.completed.xml`
  - `doc0750.temporal.temporal-entities.wstyler.completed.xml`
  - `doc0750.temporal.temporal-entities-adjudication.gold.completed.xml`
  - `doc0750.temporal.temporal-rels.jsmith.completed.xml`
  - `doc0750.temporal.temporal-rels.wstyler.completed.xml`
  - `doc0750.temporal.temporal-rels-adjudication.gold.completed.xml`
  - `doc0750.umls.umls-entities.mpalmer.completed.xml`
  - `doc0750.umls.umls-entities.gsavova.completed.xml`
  - `doc0750.umls.umls-entities-adjudication.gold.completed.xml`
  - `doc0750.umls.umls-rels.mpalmer.completed.xml`
  - `doc0750.umls.umls-rels.gsavova.completed.xml`
  - `doc0750.umls.umls-rels-adjudication.gold.completed.xml`

In this organization, all files are kept from all stages of annotation, file names and contents are never changed (after the initial marking of a document as complete), and identifying (or accessing) data from prior stages of annotation is straightforward. Annotator names are preserved for future reference.

9.3 Adding New Files

To add a new file, corpus, or annotation to Anafora, take the below steps:

1. Make sure that all text files are plaintext, without extension (e.g. a full filename of ”doc001”)
2. Place all text files in folders named after the document (e.g. ”doc001/doc011”)
3. Place all of those folders into a subcorpus folder, and copy that into the existing project folder
4. Move to the top-level folder that you’ve added and run the commands necessary to ensure the files have the correct permissions:

```
chgrp -R anaforaadmin *
setfacl -R  -m u:nobody:rwx *
chmod -R 770 *
find -type d -exec chmod g+s {} \;
```

Your files will now appear in Anafora as expected.

9.4 Adminstrative Files

Anafora looks for (and recognizes) two special filenames in each document folder which can be used by adminstrators to manage annotations.

- `.noassign` - If a `.noassign` file is present in a document's folder, the document will not be assigned to annotators even if some annotations are not yet complete. This is useful for a document for which you've converted Gold-standard data and want to keep in the project file, but don't want duplicated effort.
- `.nolimit` - Adding a `.nolimit` file disables Anafora's limit of two annotators who can work on each pass of each document. This is useful for training or demonstration files, where each new annotator might want to work on the file, not just two of them.

The contents of the files are irrelevant, Anafora only checks that no file by these names are present in the document folder.

9.5 Manipulating annotations in Anafora

Although there is seldom need for administrator intervention, manipulating annotations in Anafora is quite easy, and simply involves creating or moving files around in the Project file.

9.5.1 Manually assigning a document to a specific annotator

To manually assign a document to a given annotator, simply create (using `touch` or an equivalent a file in the folder of the document you'd like to assign, with the following format:

`[sourcetextname].[schema].[annotationtype].[annotatorname].inprogress.xml`

The next time the annotator logs in, that file will show up in their "in progress" section.

9.5.2 Manually reassigning a document

The easiest way to manually reassign a document (after the departure of an annotator, for instance), delete (or move) the existing, in-progress annotation file, making the file again eligible for automatic assignment.

If you'd rather reassign the existing annotations, simply change the annotator name in the filename to match the new annotator. The next time the annotator logs in, that file will show up in their "in progress" section. New annotations will use the new user's name in the annotation ID.

10 Schema design and creation

10.1 The basic format of an Anafora schema

Before annotation can begin, one must specify the annotation schema to be used by the annotators. This schema will be composed of two principal types of annotation: entity annotations (which have spans in the text), and relation annotations (which have no spans, and refer only to existing entity annotations). (Note that in this software and in these guidelines, ”entity” is a generic term, and of course, need not refer specifically to ”Named Entities”, but instead to any type annotation applied to a span of text.)

At the very least, every schema must have at least one entity annotation and at least one (top level) entity-type group. In addition to that, you can specify different properties for these annotations and groups, and of course, if your schema requires relations, one can add relations, relation groups, and properties of those relations as well.

10.1.1 Designing the XML schema

Rather than using the GUI to create the annotation schema needed (as is the case in Protégé/Knowtator or eHost/Chartreader), Anafora schemas are created as separate, easily-editable XML files. As such, the schemas follow XML convention, and can be parsed using conventional XML parsing tools, if needed.

There are just four cardinal rules of schema design using Anafora:

1. All entity and relation annotation types and groups should have unique names.
2. Names, Property types, and other within schema labels should not contain whitespace. Use underscores instead.
3. All entity and relation annotations should be placed within at least one <entities> or <relations> group.
4. The entire schema must be within <schema> tags

If these four requirements are met, and you follow the instructions outlined below, you’re only minutes away from a working annotation schema.

Caution! *Don’t use spaces in any part of your schema, use underscores instead. If you include whitespace in your schema, the schema will not load!*

10.1.2 A minimal schema

A very basic schema, containing entities, relations, and two groups, is reproduced below. The below schema, if copied into a plaintext file and saved with a .xml file, would open in Anafora as a complete schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema>
<entities type="Animals" color="a60000">
       <entity type="Carnivore" color="ff0000">
              <properties>
              </properties>
       </entity>

       <entity type="Herbivore" color="00cc00">
              <properties>
              </properties>
```

```
	</entity>
</entities>

<relations type="AnimalRelations" color="006363">
	<relation type="eats" color="33cccc">
		<properties>
			<property type="Predator" input="link" maxlink="1" instanceOf="Animals"/>
			<property type="Prey" input="link" maxlink="1" instanceOf="Animals"/>
		</properties>
	</relation>
</relations>
</schema>
```

Here, in addition to the XML header, and the schema tags, there are two entity types (”Carnivore” and ”Herbivore”), both in the group ”Animals”, and one relation, ”eats”, which is in the relation group ”AnimalRelations”. No properties are specified for either of our animal types, and the relation has only the properties necessary to link the entities together.

Caution! *A more complete demo schema, featuring all of the property types and permutations currently possible in Anafora, is available on our website, https://github.com/weitechen/anafora*

10.1.3 Assigning Hotkeys to Annotations

The Anafora tool is designed to speed up annotation in a variety of ways, one of which being the use of hotkeys in the annotation flow. You may assign a hotkey to any entity (or relation) annotation by adding `hotkey="[your desired key]"` to their `entity type` statement. Hotkeys may be any single key on the keyboard, including letters, numbers and symbols. Uppercase keys (”S”, ”Shift+s”) are not supported.

With hotkeys assigned, our initial schema might look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema>
<entities type="Animals" color="a60000">
	<entity type="Carnivore" color="ff0000" hotkey="c">
		<properties>
		</properties>
	</entity>

	<entity type="Herbivore" color="00cc00" hotkey="h">
		<properties>
		</properties>
	</entity>
</entities>

<relations type="AnimalRelations" color="006363">
	<relation type="eats" color="33cccc" hotkey="e">
		<properties>
			<property type="Predator" input="link" maxlink="1" instanceOf="Animals"/>
			<property type="Prey" input="link" maxlink="1" instanceOf="Animals"/>
		</properties>
	</relation>
```

```
</relations>
</schema>
```

Now, if an annotator presses ”c” during annotation with a given span selected, it’ll automatically create a ”Carnivore” annotation. Similarly, if the annotator presses ”h”, an ”Herbivore” annotation will be created.

If the annotator presses ”e”, a blank ”eats” relation will be created. Clicking on an entity annotation before pressing ”e” will automatically fill the first slot (here, ”predator”) with the selected entity.

10.2 Creating spanned (”entity”) annotations in the schema

One of the most basic types of annotation is the classification of a given span in the text as being of a given named entity type or being classified as X. These spanned annotations are referred to here as “entity” annotations. In their most basic form, they look like the below:

```
<entity type="Animal" color="1cf9e7">
       <properties>
       </properties>
</entity>
```

In this case, this represents an NE type called ”Animal” (to be applied, presumably, to all animals in the document), with the hexadecimal color code (for display) 1cf9e7. If your entity has no additional properties, your work is done.

Every entity (and relation) must have a unique name in the schema, and names may not include the space character. Use the underscore ”_” instead.

10.2.1 Assigning Hotkeys to Annotations

The Anafora tool is designed to speed up annotation in a variety of ways, one of which being the use of hotkeys in the annotation flow. You may assign a hotkey to any entity (or relation) annotation by adding `hotkey="[your desired key]"` to their `entity type` statement. Hotkeys may be any single key on the keyboard, including letters, numbers and symbols.

With hotkeys assigned, our initial schema might look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<schema>
<entities type="Animals" color="a60000">
       <entity type="Carnivore" color="ff0000" hotkey="c">
              <properties>
              </properties>
       </entity>

       <entity type="Herbivore" color="00cc00" hotkey="h">
              <properties>
              </properties>
       </entity>
</entities>
```

Now, if an annotator presses ”c” during annotation with a given span selected, it’ll automatically create a ”Carnivore” annotation. Similarly, if the annotator presses ”h”, an ”Herbivore” annotation will be created.

10.3 Specifying properties for entity annotations

Sometimes, you need annotators to specify additional properties of an existing entity annotation. These can be additional attributes which must be specified by annotators, or pointers to existing entities relations within the text. These will be specified using the <property> tag.

10.3.1 Specifying attribute properties of entities

Let's once again work from an example:

```
<entity type="Animal" color="1cf9e7">
	<properties>
		<property type="Fuzzy" input="choice">Yes,No</property>
		<property type="Modality" input="choice">Actual, Hypothetical, Extinct</property>
		<property type="Diet" input="choice">,Carnivore,Herbivore,Omnivore</property>
	</properties>
</entity>
```

Here, we have three possible properties for each "Animal" annotation. All three are "choices", where the annotator is forced to choose one of the provided options to fill the properties, so we specify `input="choice"`. The first allows us to choose whether or not the animal is fuzzy ("Yes" or "No"). The second allows us to choose whether the animal is "Actual", "Hypothetical", or "Extinct". The third allows us to choose whether the animal is a carnivore, herbivore, or omnivore.

10.3.2 Default values

It's worth noting that in this schema design system, *the first item in the list of attribute properties is the default annotation*. Without any additional action by the annotator, all animals will be classified as Fuzzy(Yes) and Modality(Actual). *To specify that there is to be no default annotation, one must place a comma (,) before the first of the possible items*. In effect, this is negating the first position

10.3.3 Multiple Choice

What happens if you have a property for which more than one choice is a possibility:

```
<entity type="Animal" color="1cf9e7">
	<properties>
		<property type="LivesIn" input="multichoice">Water, Land, Air</property>
	</properties>
</entity>
```

Here, we have a new property for the "Animal" annotation. Some animals live in both water and air, so we need to be able to specify that, we do that using the `input="multichoice"`. Now, annotators can select both "water" and "land", say, for a turtle.

10.3.4 Using other entities to fill property slots

Sometimes, you'll want annotators to be able to specify another entity which fills a property slot. This may be for linguistic cues like negation or modality indicators, to show relations (like a disorder's location in the body), or for any other reason you'd like. Working again from our example:

```
<entity type="Animal" color="1cf9e7">
	<properties>
		<property type="Fuzzy" input="choice">Yes,No</property>
		<property type="Modality" input="choice">Actual, Hypothetical, Extinct</property>
		<property type="Diet" input="choice">,Carnivore,Herbivore,Omnivore</property>
		<property type="ScaredOf" input="list" maxlink="1" instanceOf="Animal,Human,Plant"/>
	</properties>
</entity>
```

Here, we're able to specify a property ("ScaredOf") which is filled in by pointing to other annotations in the text, which specifies which other entities frighten the entity in question. Let's break that new line down a bit further.

The `input="list"` portion designates that this will be a list of entities (even if only a "list" of one). This is the case for any property which is filled by linking to another entity.

The `maxlink="1"` section tells us that only one entity can be chosen to fill that particular slot, in this case, stating that you can only mark a given animal as being scared of one other entity tagged "animal". To specify that any number of entities can go into that slot, specify an absurdly high number (`maxlink="3000"`).

The next section, `instanceOf="Animal,Human,Plant"` specifies that the only allowable annotation types for this slot are "Animal", "Plant" and "Human" (which have been, presumably, described elsewhere in the schema). The specified annotation types can be either entities or relations. This allows you to limit the possible types for a given slot, for instance, limiting the NegationIndicator property so that annotators can only fill that slot with an entity of type NegationIndicatorClass.

When specifying a slot property, to specify that any entity can be fill that slot, specify `instanceOf="Entity"`. To allow any relation, specify `instanceOf="Relation"`. To allow any annotation, specify `instanceOf="Entity, Relation"`.

Finally, it's worth pointing out that writing the property tag using a final `/>` (as below):

```
<property type="ScaredOf" input="list" maxlink="1" instanceOf="Animal,Human,Plant"/>
```

... is functionally identical to closing it using a full `</property>` tag:

```
<property type="ScaredOf" input="list" maxlink="1" instanceOf="Animal,Human,Plant"></property>
```

In the case of slot properties or relations (where nothing goes between `<property...>` and `</property>`), it makes sense to save some typing and use the final `/>` notation.

10.3.5 Allowing freeform notes or comments as properties

Although the program allows a comment field for any annotation by default, sometimes, annotators need to be able to add a freeform note, identifier code or other text information to annotations. To do this, we introduce our final property class:

```
<entity type="Animal" color="1cf9e7">
	<properties>
		<property type="Fuzzy" input="choice">Yes,No</property>
		<property type="Modality" input="choice">Actual, Hypothetical, Extinct</property>
		<property type="Diet" input="choice">,Carnivore,Herbivore,Omnivore</property>
		<property type="ScaredOf" input="list" maxlink="1" instanceOf="Animal,Human,Plant"/>
		<property type="Name" input="text"></property>
	</properties>
</entity>
```

The new property, ”Name”, is meant to allow annotators to specify the name of the Lion (”That’s not just any Lion, that’s Herbie!”), and because of `input="text"`, the input is allowed to be any symbol or character which is answered

10.4 Creating Relation annotations in the schema

As with named entities, each relation must have a unique name in the schema, and will likely want to have a unique color. Below is a very basic relation, once again in our example safari-themed annotation schema:

```
<relation type="eats" color="f61cf9">
	<properties>
		<property type="Predator" input="list" maxlink="1" instanceOf="Animal,Human"/>
		<property type="Prey" input="list" maxlink="3000" instanceOf="Animal,Human,Plant"/>
	</properties>
</relation>
```

This will produce a relation called ”eats”, with the hex code color #f61cf9. This relation has two slots, specified in much the same ways as slots for entity properties are specified.

The first slot, labeled ”Predator” (`type="Predator"`), used to mark the entity doing the eating, can be filled by only one (specified by `maxlink="1"`) entity annotation of the types ”Animal” or ”Human” (specified by `instanceOf="Animal,Human"`). Once again, `input="list"` specifies that this will be a list of entities. So, a plant cannot be a predator in the above relation.

Once again, when specifying a slot for a relation, to specify that any entity can fill that slot, specify `instanceOf="Entity"`.

The second slot, ”Prey”, is used to mark the entities which get eaten. In this case, any number of entities can be chosen (because there is no number in the `maxlink=""` statement), but those entities must be of the types ”Animal”, ”Human”, or ”Plant” (specified by `instanceOf="Animal,Human,Plant"`)

To actually make the relation in annotation, the annotator would create a relation, fill the ”Predator” slot with some predatory entity (a lion, perhaps), then fill the ”Prey” slot with any entities that the lion might eat (gazelles, wildebeest, linguists, and the like).

10.4.1 Single-slot relations

Not all relations will have more than one slot. In coreference annotation, you might have something like the below:

```
<relation type="Identical" color="f61cf9">
	<properties>
		<property type="Identical" input="list" maxlink="3000" instanceOf="markable"/>
	</properties>
</relation>
```

This specifies a relation, named ”Identical”, which can be filled by a great many (`maxlink="3000"`) entities of type ”markable”. Here, the relation is more of a list of coreferent entities.

10.4.2 Multiple-slot relations

Similarly, there’s no reason that a relation couldn’t exist with more than two slots. For instance:

```
<relation type="eats" color="f61cf9">
	<properties>
		<property type="Predator" input="list" maxlink="1" instanceOf="Animal,Human"/>
		<property type="PreferredPrey" input="list" maxlink="3000" instanceOf="Animal"/>
		<property type="Prey" input="list" maxlink="3000" instanceOf="Animal,Human,Plant"/>
	</properties>
</relation>
```

This is an expansion of our above ”eats” relation, which would allow annotators to specify, for instance, that lions prefer to eat gazelles and wildebeest, but that they will eat linguists and birds in a pinch. Max link is 3000 because that’s a very large number, unlikely to be reached by annotators (practically infinite), but it could as easily have been 14000 or 1000000.

10.4.3 Properties of relations

Relation properties are defined identically to entity properties, as you might suspect. So, there’s nothing stopping us from modifying our initial ”eats” relation to include a property or two:

```
<relation type="eats" color="f61cf9">
	<properties>
		<property type="Predator" input="list" maxlink="1" instanceOf="Animal,Human"/>
		<property type="Prey" input="list" maxlink="3000" instanceOf="Animal,Human,Plant"/>
		<property type="Polarity" input="choice">Positive,Negative</property>
		<property type="Season" input="choice">,Summer,Winter,Autumn,Fall</property>
	</properties>
</relation>
```

Now, for each relation, we’re able to choose whether its polarity is positive (”Yes, this entity eats that one”) or negative (”This entity does NOT eat that one”). Once again, here, ”Positive” is the default. We can also specify whether X eats Y in the summer, winter autumn or fall, and because of the initial comma, no default value is assumed.

10.5 Grouping Entities and Relatons

The <entities> and <relations> tags are used to create hierarchies and group annotations in your schema. These are absolutely critical, as *every <entity> or <relation> must be included in at least one <entities> or <relations> group*, but they can also be used for further subdivision of the schema, as discussed below.

For example, we may want to divide our different animal entities into pleasant and unpleasant animals.

```
<entities type="Animals" color="f61ff8">
	<entity type="PleasantAnimal" color="f9ec1c">
		[...]
	</entity>
	<entity type="UnpleasantAnimal" color="e8ec1c">
		[...]
	</entity>
</entities>
```

Or we could go further still with subdivision, creating a three-tiered hierarchy:

```
<entities type="Animals" color="f61ff8">
	<entity type="GenericAnimal" color="f9ec1c">
		[...]
	</entity>
	<entities type="PleasantAnimals" color="f9ec1c">
		<entity type="FuzzyAnimal" color="f61cf9">
			[...]
		</entity>
		<entity type="CuteAnimal" color="f61cf9">
			[...]
		</entity>
		<entity type="TastyAnimal" color="f61cf9">
			[...]
		</entity>
	</entities>
	<entities type="UnpleasantAnimals" color="e8ec1c">
		<entity type="SmellyAnimal" color="1cf9e7">
			[...]
		</entity>
		<entity type="ViciousAnimal" color="1cf9e7">
			[...]
		</entity>
		<entity type="PoisonousAnimal" color="1cf9e7">
			[...]
		</entity>
	</entities>
</entities>
```

By using the <entities> and <relations> tags, any hierarchy you'd like is possible within your schema design. Note, however, that one cannot annotate using group labels or specify their properties, and as such, generic entity types must be generated for superset annotation.

Part III

The Anafora Data Format

Anafora uses a human-readable XML file format to store all completed annotations. For each document, the work of every annotator or adjudicator is saved in separate .xml files, which contain all of that individual's annotations and metadata. These XML files (when coupled with the original annotated text), provide all of the information needed for information extraction and annotation review.

11 About AnaforaXML

The Anafora data format is XML-based, and has the below basic structure, which we'll go through item-by-item in this section.

```
<?xml version="1.0" encoding="UTF-8"?>

<data>
```

```
<info>
  <savetime>21:42:14 12-03-2013</savetime>
  <progress>completed</progress>
</info>

<schema path="./" protocol="file">annotation.schema.xml</schema>

<annotations>

       <entity>
              <id>uniqueid</id>
              <span>start,end</span>
              <type>EntityType</type>
              <parentsType>ParentOfEntityType</parentsType>
              <properties>
                     <property1>value</property1>
                     <property2>value</property2>
              </properties>
       </entity>

       <relation>
              <id>uniqueid</id>
              <type>RelationType</type>
              <parentsType>ParentOfRelationType</parentsType>
              <properties>
                     <Argument>LinkedEntityID1</Argument>
                     <RelatedTo>LinkedEntityID2</RelatedTo>
              </properties>
       </relation>

</annotations>

<adjudication>
</adjudication>

</data>
```

11.1 <info>

In the <info> section of the AnaforaXML file, there are two entries:

- `<savetime>21:42:14 12-03-2013</savetime>`
  - This records the time the file was saved most recently.
- `<progress>completed</progress>`
  - This section tells us if the annotation is *completed* or *inprogress*, determined by the annotator's choice and reflected in the filename.

11.2 <schema path...

The `schema path` line simply dictates the identity and location of the schema used for annotation on the server. So, for:

```
<schema path="./" protocol="file">temporal.schema.xml</schema>
```

We know that the schema is located at the project file root, and the name of it is `temporal.schema.xml`.

11.3 <entity>

For a given entity, the following is specified between the <entity> tags. Let's look at the below sample <entity> as an example:

```
<entity>
	<id>24@e@clinicaldoc12@stylerw</id>
	<span>47,52</span>
	<type>EVENT</type>
	<parentsType>TemporalEntities</parentsType>
	<properties>
		<Tense>PAST</Tense>
		<Polarity>POSITIVE</Polarity>
		<Cause>18@e@clinicaldoc12@stylerw</Cause>
	</properties>
</entity>
```

Annotation ID

```
<id>24@e@clinicaldoc12@stylerw</id>
```

This is a unique, collision-resistant string which is used to specify the particular annotation individually. It is composed of the following, where "e" stands for "entity":

`Counter@e@DocumentName@AnnotatorUsername`

So, reading the ID string from our sample, we know that this particular annotation was the 24th created in the document, is an entity, belongs to a file named *clinicaldoc12*, and was created by an annotator with the server login name stylerw.

Span

```
<span>47,52</span>
```

This is the span, measured by character offset, in the format `start,end`. For disjoint spans, the secondary spans are separated from the original by a semicolon:

```
<span>1377,1386;1407,1409</span>
```

Note that Anafora does not incorporate the span text into the saved XML file. This was a conscious decision to allow AnaforaXML files to contain none of the source text, an important factor when dealing with medical records or other sensitive documents.

Annotation Type

```
<type>EVENT</type>
```

This is the annotation type, and will always match the type values from the associated annotation schema. In this case, we have marked an entity of type "EVENT".

ParentsType

```
<parentsType>TemporalEntities</parentsType>
```

This is the annotation type which is “parent” to the selected annotation type in the schema. In this case, the parent of EVENT in our schema is “TemporalEntities”.

Properties

```
<properties>
	<Tense>PAST</Tense>
	<Polarity>POSITIVE</Polarity>
	<Cause>18@e@clinicaldoc12@stylerw</Cause>
</properties>
```

In the <properties> section, properties are specified individually, in the format:

```
<PropertyName>Value</PropertyName>
```

The Property’s name is read in from the schema, and the value is determined by annotation, and is represented identically whether chosen by single choice, multiple choice, or free entry.

When properties are themselves relations (as with <Cause> above) and point to another annotation, the value of the property is the Unique ID of the associated annotation.

All properties specified in the schema are stored upon creation of the entity (according to default value if no changes are specified), and the contents of the <properties> section will vary depending on the nature of the entity being annotated.

Each additional entity annotation is then added surrounded by <entity> tags.

11.4 <relation>

Relation annotations, as below, are nearly identical to entities in the Anafora XML format:

```
	<relation>
		<id>93@r@clinicaldoc12@stylerw</id>
		<type>BEFORE</type>
		<parentsType>TemporalRelations</parentsType>
		<properties>
			<Argument>25@e@clinicaldoc12@stylerw</Argument>
			<RelatedTo>34@e@clinicaldoc12@stylerw</RelatedTo>
			<Polarity>POSITIVE</Polarity>
		</properties>
	</relation>
```

The only notable differences are the lack of a attribute and the substitution of an “r” (for “relation”) in the Unique ID.

In addition to any conventional properties specified in the schema (like Polarity here), links to entities are also stored as properties, named as designated in the schema.

So, our example relation specifies that the first ID listed (25@e...) is in a BEFORE relation with the second (34@e...).

Relations with more than two linked entities are specified by duplicating the properties field, as in the below coreference chain:

```
<relation>
	<id>53@r@clinicaldoc12@stylerw</id>
	<type>IDENTICAL</type>
	<parentsType>EventCoreference</parentsType>
	<properties>
		<FirstMention>15@e@clinicaldoc12@stylerw</FirstMention>
		<Subsequent>39@e@clinicaldoc12@stylerw</Subsequent>
		<Subsequent>48@e@clinicaldoc12@stylerw</Subsequent>
	</properties>
</relation>
```

Here, 39@e... and 48@e... are both related (IDENTICAL) to 15@e..., and any additional entities in the chain would be added with additional `<Subsequent>` lines.

Each additional relation annotation is then added surrounded by `<relation>` tags.

Part IV

Installing Anafora on your server

12 Host System Requirements

12.1 System Requirements

- Linux or Unix-based Server
- A file system supporting per-user access controls (e.g. Ext2, Ext3)
- **Other requirements exist and will be added to a future version of the manual**

12.2 Software Dependencies

- Apache
- LDAP (for account permissions management)
- Django
- Python 2.7
- **Other dependencies exist and will be added to a future version of the manual**

13 Installation

Until this point, we've spent most of our time coding. Installation instructions for a new machine are pending, check for a newer version of our manual for new information.