



CIVET  
Contentious Incident Variable Entry Template  
- - - DRAFT - - - \*

Philip A. Schrodt  
Parus Analytics  
Charlottesville, VA  
schrodt735@gmail.com

Version 0.6 : July 11, 2015

---

\*Acknowledgements: The development of CIVET is funded by the U.S. National Science Foundation Office of Multidisciplinary Activities in the Directorate for Social, Behavioral & Economic Sciences, Award 1338470 and the Odum Institute at the University of North Carolina at Chapel Hill with additional assistance from Parus Analytics. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License; the software is open source and under the MIT license.

# 1 Introduction

This is the documentation for a prototype of the CIVET<sup>1</sup>—Contentious Incident Variable Entry Template—customizable data entry system. CIVET is being developed by the NSF-sponsored project titled “A Method for Leveraging Public Information Sources for Social Science Research” which is creating tools to improve the efficiency of data generation in the social sciences. The project has an initial focus on coding event data in the domain of contentious politics, but we expect that these tools will be relevant in a number of data-generation domains.

The core objective of CIVET is to provide a reasonably simple—yes, simple—set of commands that will allow a user to set up a web-based coding environment without the need to master the likes of HTML, CSS and Javascript. As currently implemented, the system is a rather ugly prototype; it will also be evolving as we add additional elements. Nonetheless, the system should now be useable for coding.

CIVET is implemented in the widely-used and well documented Python-based Django system<sup>2</sup> which is widely available on various cloud platforms: a rather extended list of “Django-friendly” hosting services can be found at

<https://code.djangoproject.com/wiki/DjangoFriendlyWebHosts>

The complete CIVET code is licensed as open source—we are currently using the MIT license—and provided on GitHub.

CIVET is currently implemented in two modes:

**Coding form template:** This is a template-based for setting up a web-based coding form which implements several of the common HTML data entry formats and exports the resulting data as a tab-delimited text file. This is fully functional and should be useable for small projects.

**Text annotation/extraction:** This is the more advanced system that provides for manual and automated text annotation, then the ability to extract various types of information into the fields of a coding form. At present the coding form is fixed and consequently just provides a demonstration, but we expect to make this customizable in the near future.

We are very interested in feedback on this system, including any bugs you encounter (please let us know what operating system (e.g. Windows, OS-X) and browser (e.g. FireFox, Explorer, Chrome) you were using), aspects of the manual that are unclear (and features that appear too complex), and additional features that would be useful. Please send any suggestions to [schrodt735@gmail.com](mailto:schrodt735@gmail.com).

---

<sup>1</sup><http://en.wikipedia.org/wiki/Civet>

<sup>2</sup>An earlier prototype was implemented in the Flask framework: see Appendix 3

## 2 Installing CIVET

To date we've only installed the system on Macintosh computers, though the only difference between a Macintosh installation and other installations should be the installation of the Django system.

On Macintoshes running OS-X 9 and 10, the required Python 2.7 comes pre-installed. The `pip` installation program may also be pre-installed—I'm having trouble determining this from the Web, and forget whether I had to install it when I last upgraded—but if not, install that.

1. In the Terminal, run `sudo pip install Django`: you will need administrative access to do this.
2. Download the CIVET system from <https://github.com/philip-schrodt/CIVET-Django>, unzip the folder and put it wherever you would like
3. In the Terminal, change the directory so that you are in the folder *Django-CIVET/djcivet\_site*
4. In the Terminal, enter `python manage.py runserver`
5. In a browser, enter the URL [http://127.0.0.1:8000/djciv\\_data/](http://127.0.0.1:8000/djciv_data/)

At this point you should see the CIVET home screen shown in Figure 1

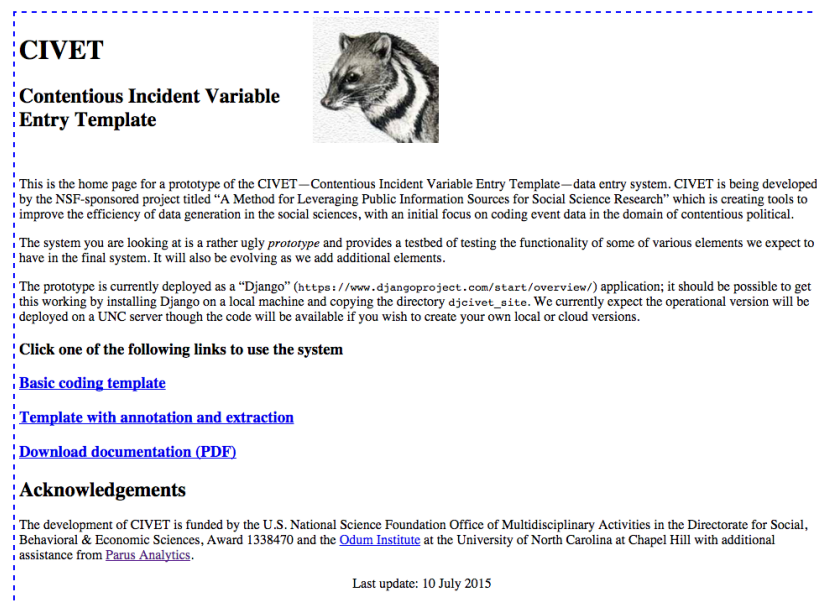


Figure 1: CIVET home screen

### 3 Template-Based Data Entry Form Operating Instructions

From the home page, click the link in the line *Basic coding template* link

1. Choose a template file then click the **Read file** button or just click the **Use demo template** button without selecting a file. You can also enter a coder ID if you want to use the `_coder_` variable. The **Download demo template** will download a copy of the template.
2. Enter data using the usual HTML data entry widgets
3. To save a set of coded fields, click one of the buttons which follow the title **Options after saving**:

**Code another case:** Save, then return to the same form

**Download data:** Save, then download data as a tab-delimited text file

4. The “Download CIVET data” page provides a text box for a file name, and the **Download file** button downloads the coded data. Use the *Start new data file* link to re-start the coding and the *Continue coding with this file* link to continue adding to the existing records.
  - The .txt file is tab-delimited and contains the variable names in the first line.
  - If the file name does not end in “.txt,” this suffix will be added.
5. To quit the program, just close the window.<sup>3</sup>

### 4 Introduction to Civet Coding Form Templates

A CIVET template file specifies the individual components of the form: these are the familiar components from web forms but the syntax used to specify them is simpler than what you will find in HTML.

CIVET is simply adding these controls to an HTML `<form>` and, as with all things HTML, most of the placement of the fields is handled by the browser.<sup>4</sup> CIVET provides some limited formatting through the insertion of text and line breaks, and with some experimenting you should be able to keep the form from being too ugly.

---

<sup>3</sup>This, it turns out, is a HTML/Javascript security feature which prevents rogue websites from closing windows unless they have created the window.

<sup>4</sup>Writing in HTML5 and CSS, one can actually exercise a very fine degree of control over the placement, but if you are comfortable with that sort of code, you presumably aren’t using CIVET in the first place. That said, you can see the HTML generated by CIVET by using the *View source* option in your browser, then save it as a file using *Save Page As...* and that could provide a starting point for creating prettier code.

The template file should be a simple text file—most systems are happier if this ends in the suffix `.txt`—similar to that used in an *R* or *Stata* script (that is, not a formatted file such as that produced by Word). Appendix 1 gives an example of a template file, and the code for this can also be downloaded from a link in the program.

At present the program does only a very limited amount of error checking; more of this will be added in the future. If the template does contain one or more errors, the system will display this on a web page.

## 4.1 Command formats

Commands generally have the following format

```
command: entry-title [var-name] options
comma-delimited list
```

Commands vary in how many of these components they have, but all follow this general pattern.

Each command ends with a blank line (or, if you prefer, the commands are separated by blank lines.)

Commands can also be cancelled by adding a “-” in front of the command: this will cancel the entire command, that is, all of the lines associated with the command, not just the first line. For visual symmetry, a “+” in front of the command “activates” it, though the command will also be active without the plus.

“#” denotes a comment: anything following a “#” is ignored, so lines beginning with “#” are completely ignored.

## 4.2 Items in template specification

The commands involve one or more of the following items:

**entry-title** : This is the title of data entry field. If this ends with / a line-break (`<br>`) is inserted after the text. The titles are escaped: at present the characters `<`, `>` and the single and double quotes are replaced with the equivalent HTML entities `&lt;`, `&gt;`, `&quot;`, and `&rsquo;`.<sup>5</sup> The **entry-title** field cannot contain the characters “[” or “]”—if these are present they will be interpreted as bounding the **var-name** field—but the escaped versions “\[” and “\]” are allowed.

---

<sup>5</sup>In the current implementation, named HTML entities such as `&copy;` and `&euro;` can be included and should produce the correct character. At present numbered entities such as `&#91;`—the HTML equivalent of “]”—do not work since the `#` is interpreted as a comment delimiter: depending on whether there is demand for this feature, the system could provide a way around this.

**var-name** : The text of the variable name for this field; this will be used in the first line of the .csv output file

**comma-delimited-option-list** : a list of the items that can be selected, separated by commas. A ‘\*’ at the beginning of the item means that it will be initially selected.

**comma-delimited-var-name-list** : a list of items which appear in **var-name** fields, separated by commas.

**page-text** : Any text

**number** : An integer

## 5 Templates: Specifying variables

### 5.1 Specifying variables to save

This command gives the variables that will be saved; these can be in any order but each of these must correspond to a **var-name** somewhere in the form, or are one of the special variables discussed below. A tab-delimited version of this list will be the first line of the output file. The command can occur anywhere in the file.

**save:** comma-delimited-var-name-list

**Example:**

```
save: worldregion, eyewitness, groupname, comments
```

### 5.2 constant

Sets the value of a variable to a constant; this can be used in a **save:**

**constant:** page-text [varname]

**Example:**

```
constant: Data set 0.2 [data_id]
```

### 5.3 filename

Sets the default file name for the downloads: this can be changed before downloading

**filename:** page-text

**Example:**

filename: our\_wonderful\_data.csv

## 5.4 Special variables

`_coder_` : Text entered in the *CIVET template selection* page

`_date_` : Current date. this is currently in the form DD-mmm-YYYY but later versions of the system will allow other formats

`_time_` : Current time in hh:mm:ss format

# 6 Templates: Data Entry Fields

## 6.1 Checkbox

A simple binary check-box. The value of the variable will be first item in the list when the box is not checked; the second item when the box is checked. The \* notation on the second item can be used to specify whether or not the box is initially checked.

**select:** entry-title [var-name]  
comma-delimited-option-list

**Example:**

select: Eyewitness report? [eyewit]  
no,\*yes

## 6.2 Select from pull-down menu

Pull-down menus—which are called a “select” in HTML—are specified with the syntax

**select:** entry-title [var-name]  
comma-delimited-option-list

**Example:**

select: Region [worldregion]  
North America, South America, Europe, \*Africa, Middle East, Asia

## 6.3 Radio buttons

A series of radio buttons are specified with the syntax

**radio:** entry-title [var-name]  
comma-delimited-option-list

The entry / in the option list causes a line-break (<br>) to be inserted

**Example:**

```
radio: Region/ [worldregion]
North America, South America, Europe, *Africa, /,Middle East, Asia
```

## 6.4 Enter single line of text

This creates a box for a single line of text (HTML `type="text"`). The `width = number` is optional and specifies the size of the text entry box in characters: the default is `width = 32`

```
textline: entry-title [var-name] width = number
initial-text
```

**Example:**

```
textline: Name of group [groupname]
<enter name>
```

## 6.5 Extract single line from annotated text

This creates a box for a single line of text (HTML `type="text"`) that will interact with annotated text; in addition information can be manually entered or cut-and-pasted into this box. If this command is used in a form that does not have associated annotated text, it behaves the same as `textline` and the `class` information is ignored.

The `class=class-name` is required and specifies the name of the annotation class that the text-entry box is connected with; a class can be associated with multiple text-entry boxes. There are three standard classes:

- **nament:** named-entries, which are determined by capitalization
- **num:** numbers
- **date:** dates

The `width = number` is optional and specifies the size of the text entry box in characters: the default is `width = 32`

```
textclass: entry-title [var-name] class=class-name width=number
initial-text
```

**Example:**

```
textclass: Name of city [cityname] class=nament
<enter city>
```



## 6.6 Enter multiple lines of text

This corresponds to an HTML “TEXTAREA” object. The `rows = number cols = number` is optional and specifies the size of the text entry box in characters: the default is `rows = 4 cols = 80`

```
textarea: entry-title [var-name] rows = number cols = number
initial-text
```

**Example:**

```
textarea: Comments [comments] rows = 2 cols = 64
-- put any additional comments here --
```

## Templates: Additional Web Page Formatting

### 6.7 Set page title

Sets the title of the web page: that is, the HTML `< title > ... < /title >` section of the header.

```
title: page-text
```

**Example:**

```
title: CIVET-based coding form
```

### 6.8 Insert text

Adds text to the form: the various options follow the usual HTML formats. In interests of simplicity, text is “escaped” so that special characters are not interpreted as HTML: note that this means that in-line mark-up such as `< i >`, `< b >` and `< tt >` will not work, so if you need this activate and use the **html:** command. Also keep in mind that these commands need to be separated by a blank line.

```
h1: page-text
h2: page-text
h3: page-text
h4: page-text
p: page-text
```

**Example:**

```
h1: Primary data set coding form
```

`p:`Please enter data in the fields below, and be really, really careful!

The simple command

`p:`

is useful for putting some space between form elements.

## 6.9 Insert HTML

[This command may or may not be included in the operational version of the system, as it provides some opportunities for mischief. Stay tuned. It is in the code but currently deactivated; if you are installing your own version of the system, it can be activated by changing a single character in the source code.]

Adds arbitrary HTML code without escaping.

**html:** page-text

## 6.10 Insert a line break

Adds a new line in the form

**newline:**

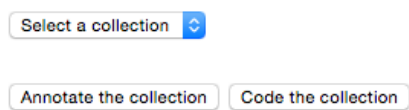
## 7 Text Annotation and Extraction

The text annotation and extraction component of CIVET allows text to be annotated (and edited) either automatically or manually, and then information extracted either using the annotation, with copy-and-paste, or simply entered. In order to keep track of this, CIVET implements a document management system in the YAML format; this provides a systematic means of storing the texts, meta-data such as the source and copyright information, and the coded records. In the current implementation, we only have a proof-of-concept prototype of this system: this is not quite ready for use in actual coding.

To activate the demo, from the home page, click the link *Template with annotation and extraction*. This will take you to a screen (Figure 2) where you can select a collection of texts to annotate and code: this is done with the **Select a collection** pull-down menu. In the current implementation, this set is fixed; in the operational version these can be uploaded from your computer.

### Select a collection to edit:

**Note 10-July-2015:** These collections are currently hard-coded in the system. In the operati



**Operating instructions for the text-extraction demonstration are [here](#)**

Figure 2: CIVET Collection selection screen

Exit this screen by clicking either

#### Annotate the collection:

This goes to the annotation and editing page described in Section 8. The sample texts have not been annotated so for purposes of the demonstration you will probably want to use this option.

#### Code the collection:

This goes to the coding and text extraction page described in Section 9. This option can be used when working with texts that have already been annotated.

## 8 Annotation and Editing

The annotation and editing page implements a minimal version<sup>6</sup> of the Javascript `ckeditor` which allows the texts to be edited and annotated. Editing works as you would expect, including cut/copy/paste options.

Annotation is handled with the **Styles** drop-down menu in the window toolbar (Figure 3): select the text you want to annotate and then select the annotation to apply. At present the system implements only four types of annotation (Figure 4)—named-entity, number, date and action—but in the full version of the system you will be able to specify additional annotation categories.

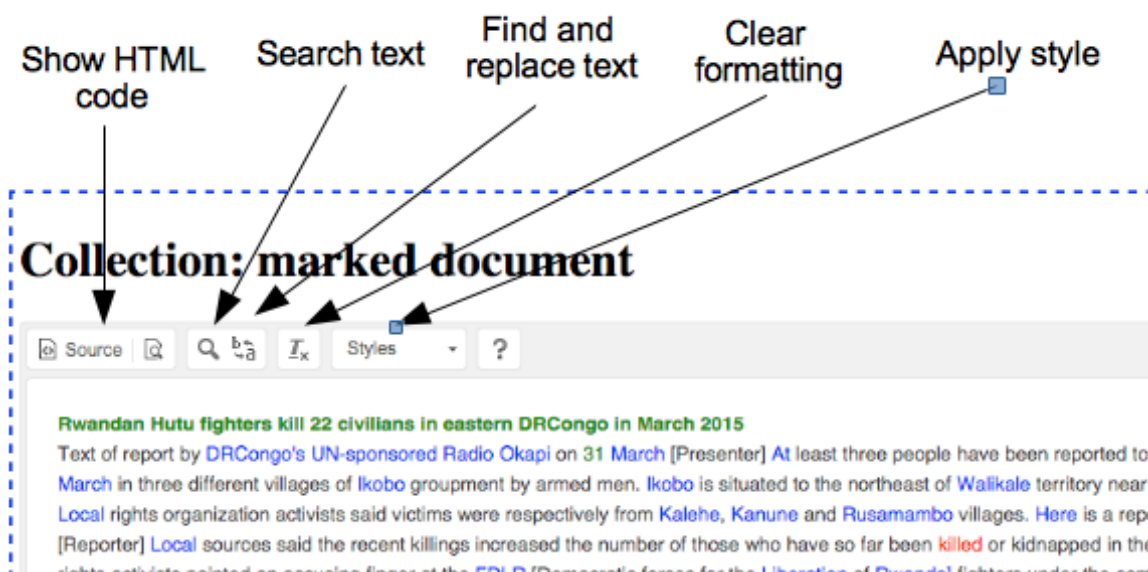


Figure 3: CIVET Editor

The following options are available on this screen

### Save edits and select new collection:

This saves whatever annotation has been done and returns to the collection section (Figure 2) screen: this option would be used if you are only annotating text rather than coding them. Annotations are eventually saved in the `textmkup` field of the YAML file along with the date of the annotating and the coder ID.

### Save edits and code the collection:

This saves whatever annotation has been done and goes to the coding and text extraction page described in Section 9.

### Discard edits and select new collection:

This discards the edits and returns to the collection section (Figure 2) screen

---

<sup>6</sup>that is, the version of `ckeditor` uses only a very small set of the features that are available for the editor: if you want to customize this, additional features can easily be added.

# Collection: TestTexts\_001

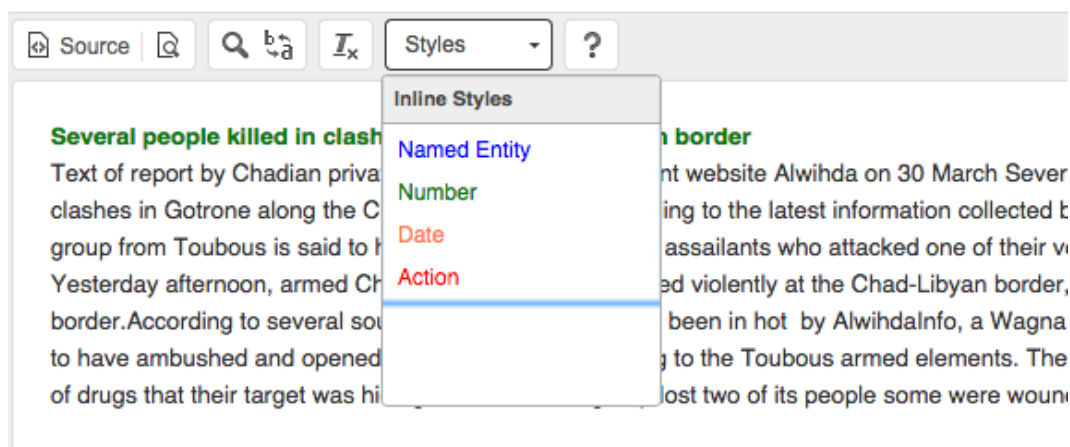


Figure 4: CIVET Editor

## Annotate the collection:

This applied the automated markup system which currently identified the following:

**Named-entities:** This is based on capitalization; consecutive capitalized words are combined.

**Numbers:** Digits: this will be extended to include numerical words and phrases such as “one” and “two-hundred”

**Special categories:** This is current implemented for a set of verbs dealing with bodily injury

Figure 3 shows the result of automated annotation: named-entities are in blue; the special category is in red.

## 9 Coding and Text Extraction

The CIVET coding form screen in the demonstration version is shown in Figure 5. The fields of the form are currently fixed but in the full version will be specified using the coding form template system described in Section 4.<sup>7</sup>

The general operation of the coder/extractor is described below:

---

<sup>7</sup>This feature is actually already operational: the form displayed is specified in the file `djcivet_site/djciv_data/static/djciv_data/CIVET.demo.coder.template.txt`; this can be modified if you want to experiment.

**Document:**

**Rwandan Hutu fighters kill 22 civilians in eastern DR Congo in March 2015**

Text of report by DR Congo's UN-sponsored Radio Okapi on [March [Presenter] At least three people have been reported to have been killed on Sunday [March in three different villages of Ikobo groupment by armed men. Ikobo is situated to the northeast of Walikale territory near the border with Lubero territory. Local rights organization activists said victims were respectively from Kalehe, Kanune and Rusamambo villages. Here is a report filed by Bernadin Nyangi. [Reporter] Local sources said the recent killings increased the number of those who have so far been killed or kidnapped in the entity this month to [The rights activists pointed an accusing finger at the FDLR [Democratic forces for the Liberation of Rwanda] fighters under the command of a certain Mutoka, as perpetrators in the killings and kidnappings. They accused the same FDLR faction of having raped at least [women this month alone as [other individuals were victims of torture and other sorts of human rights abuses.

**Template form for CIV-yaml demo files**

Type of incident: ☒ Demonstration ☐ One-sided Violent ☐ Armed Clash

Nature of incident

If "Other", provide details in the report section

Did incident involve local authorities? ☒

Location

Maximal injuries

Brief description of incident

**Options after saving:**

Figure 5: CIVET Coder

1. Clicking a text entry boxes associated with an annotation category will highlight the relevant words in text: For demonstration purposes these are

**Location:** named-entities

**Maximal injuries:** actions

The ‘tab’ key cycles between the coding fields, or an option can be selected using the mouse.

2. When a annotated category field is active, the first relevant word or phrase in the text is highlighted. The right-arrow key will cycle the highlighted word. To copy a highlighted word into the text box, use the down-arrow key.
3. Text can also be selected using the mouse: To copy the selected text into the text box, use the left-arrow key.
4. Cut-and-paste from the text to the date fields work as you would expect
5. Text can also be entered manually.

6. To save a set of coded fields, click one of the buttons along the bottom. At present, all three buttons save; we will be adding “cancel” and “reset” options. The options are:

**Return to this case:** Save, then return to the same text

**Select new case:** Save, then return to the same text

**Download data:** Save, then download data as a text file

7. The “CIVET Download” page (Figure 6) provides a text box for a file name, and the Download file button downloads the coded data as a tab-delimited text file.

## Download CIVET data:

### Enter name of output file:

civet.output.txt [" .txt" suffix will automatically be added]

Download file

### To continue, use one of the following links:

- [Start new data file](#)
- [Continue coding](#)
- [Return to home page](#)

### To quit, just close the window

Figure 6: CIVET Data download page

- The .txt file contains the variable names in the first line.
- If the file name does not end in “.txt”, this will be added.

The Download file does not exit the page: that can be done with any of these buttons

**Start new data file:** Re-start the coding with a new collection

**Continue coding with this file:** Continue adding to the records in the current data set

**Return to home page:** Return to home page

8. To quit the program, just close the window.<sup>8</sup>

---

<sup>8</sup>This, it turns out, is a HTML/Javascript security feature which prevents rogue websites from closing windows unless they have created the window.

## Appendix: Sample Template File

```
# CIVET template demonstration file

h1:Ministry of Magic Hogwarts Incident Report

radio: House where incident occurred: [house]
Gryffindor, Hufflepuff, Ravenclaw, *Slytherin

p:

select:Nature of incident [natincid]
*Minor mischief, Unauthorized absence, Accident, Major infraction, Unforgivable Curses, Other

p:If "Other", provide details in the report section

checkbox: Was incident reported to school authorities? [authreport]
No,*Yes

checkbox: Did incident involve muggles? [muggles]
No,Yes

p:

textline: Name of student(s) [names] width=80
Enter names here

p:

textarea:Brief description of incident [descrip] cols = 80
Enter brief description here

p:

textline:Reporting official [reporter] width=40
Enter your name here

h3:Thank you for your assistance; we will contact you by owl should we require more information

save: _date_, house, natincid, authreport, muggles, names, descrip, reporter
```



## Appendix 2: Input Format

CIVET works with “collections” of individual stories: these are typically multiple related news stories—“texts”—from which one or more data records—“cases”—are coded. These are stored in a YAML format—<https://en.wikipedia.org/wiki/YAML>—which is a structured human-readable text file containing a number of data fields. Fields marked with **\*\*** are required.

Collections are stored in folders and uploaded to the system in compressed form, then downloaded when a coding session is completed. So long as the YAML formatting is preserved—which should be fairly straightforward—the system is indifferent as to whether editing is done inside or outside of CIVET.

### 9.1 Collection fields

**collid** : Collection ID, which needs to be unique within a set of collections. If this is not provided in the file, collfilename is assigned by the program

**collfilename** : directory and name of the YAML file (without the suffix) where the file was read from; this is assigned by the program

**colldate** : collection date YYYY-MM-DD

**colledit** : **\*\*** datetime of editing of this collection **\*\*** [provided by system]

**colcmt** : collection comments

**texts** : one or more related texts

**cases** : zero or more coded records

### 9.2 Text fields

- **textid** : **\*\***unique text ID for CIVET. This needs to be unique within any set of collections, and given how collections can get mixed across folders, ideally should be unique for the entire project. If a value for the **text** field is not provided it will be assigned by the program.

**textdate** : text date YYYY-MM-DD **\*\***

**textpublisher** : publisher [any string]

**textpubid** : publisher ID [any string]

**textbiblio** : bibliographic citation

**textgeogloc** : geographical locations

**textauthorr** : author [any string]

**textlang** : language

**textlicense** : copyright notification or other license information

**textlede** : \*\* lede/headline/abstract—this is a short summary of the article which will be highlighted and also will appear in the sorting routine.

**textcmt** : comment

**textoriginal** : \*\*original text of the story; this will not be modified by the system

**textmkup** : \*\* marked up text: this is the annotated version of the story with any mark-up that has been added either automatically or manually

**textmkupdate** : datetime time of editing of this block \*\* [provided by system]

**textmkupcoder** : coder ID

### 9.3 Case fields

- **caseid** : \*\* Internal case/event ID. This is assigned by the program and probably should not be changed; external IDs can be entered as variables.

**casedate** : \*\* Date and time this case was coded [provided by system]

**casecmt** : comment for case

**casecoder** : coder ID

**casevalues** : This is a string formatted as a Python dictionary which contains pairs of variable names and values

Dates are ISO-8601 ([http://en.wikipedia.org/wiki/ISO\\_8601](http://en.wikipedia.org/wiki/ISO_8601); <http://www.w3.org/TR/NOTE-datetime>; <https://xkcd.com/1179/>; <http://www.cl.cam.ac.uk/~mgk25/iso-time.html>) so generally either

- YYYY-MM-DD
- YYYY-MM-DDThh:mm:ss
- YYYY-MM-DDThh:mm:ss[+-]hh:mm

Figure 7 shows an example of a simple YAML file.<sup>9</sup>

```
collid: TestTexts_001
colldate: 2015-06-08
colledit: 2015-06-08
collcmt: Text file source: Apr15.OTH.stories.txt

texts:

- textid: TestTexts_002_001
  textdate: 2000-01-01
  textpublisher: BBC Monitoring Africa
  textpubid: BBCAP
  textbiblio: BBCAP00020150401eb4100105
  textgeoloc:
  textlang: English
  textlicense: (c) 2015 The British Broadcasting Corporation. All Rights Reserved. No material may be reproduced except
with the express permission of The British Broadcasting Corporation.
  textlede: Several people killed in clashes along Chadian-Libyan border
  textcmt:
  textoriginal: |
    Text of report by Chadian privately-owned, pro-government website Alwihda on 30 March

    Several people have been killed and others injured in violent clashes in Gotrone along the Chad-Libyan border. According
    to the latest information collected by AlwihdaInfo, since yesterday [29 March], a heavily armed group from Toubous is
    said to have been in hot pursuit of assailants who attacked one of their vehicles, leading to the death of two of
    their people.

  textmkup: |
    <div class="textblock" data-textid=" TestTexts_002_001"><div class="textlede" style="color:green; font-weight: bold;">
    Several people killed in clashes along Chadian-Libyan border</div><div class="textcontent">Text of report by
    <span style="class:nament; color:blue">Chadian</span> privately-owned, pro-government website
    <span style="class:nament; color:blue">Alwihda</span> on <span style="class:num; color:green">30</span>
    <span style="class:nament; color:blue">March</span> Several</span> people have been
    <span style="class:termst; color:red" title="whacked">killed</span> and others
    <span style="class:termst; color:red" title="whacked">injured</span> in violent clashes in
    <span style="class:nament; color:blue">Gotrone</span> along the
    <span style="class:nament; color:blue">Chad-Libyan</span> border. <span style="class:nament; color:blue">According</span>
    to the latest information collected by <span style="class:nament; color:blue">AlwihdaInfo</span>,
    since yesterday [29 <span style="class:nament; color:blue">March</span>], a heavily armed group from
    <span style="class:nament; color:blue">Toubous</span> is said to have been in hot pursuit of assailants who
    <span style="class:termst; color:red" title="whacked">attacked</span> one of their vehicles, leading to the death of
    two of their people.
  textmkupdate: 2015-06-08
```

Figure 7: CIVET Coder

---

<sup>9</sup>This is a screen capture of a file being edited with BBEdit, hence the color mark-up.

## Appendix 3: Prototype on Google Application Engine

An earlier demonstration version of the program, written in the Flask framework, is deployed as an application on the Google App Engine at <http://ace-element-88313.appspot.com/>. The “Coding Form Template” option in this program works as described in Section 3. The code for this version can be downloaded from <https://github.com/philip-schrodt/CIVET-Flask>

The other option in the program is the “Text-Extraction Demonstration Form” which was a prototype of the full annotation/extraction system. To activate the demo, from the home page, click the link in the line *See a demo of the text-highlighting system by clicking here*

1. Select a text file to edit: you can use either the pull-down menu or radio boxes, then click the **Edit the file button**.
2. Click one of the text entry boxes will highlight the relevant words in text: For demonstration purposes these are words beginning with the letters ‘a’, ‘c’, ‘d’, ‘e’ and ‘s’. The ‘tab’ key cycles between these options, or an option can be selected using the mouse.
3. When a text entry box is active, the first relevant word in the text is highlighted. The right-arrow key will cycle the highlighted word. To copy a highlighted word into the text box, use the down-arrow key.
4. Text can also be selected using the mouse: To copy the selected text into the text box, use the left-arrow key.
5. Cut-and-paste from the text to the date fields work as you would expect [TEST THIS]
6. Text can also be entered manually.
7. To save a set of coded fields, click one of the buttons along the bottom. At present, all three buttons save; we will be adding “cancel” and “reset” options. The options are:

**Return to this case:** Save, then return to the same text

**Select new case:** Save, then return to the same text

**Download data:** Save, then download data as a text file

8. The “CIVET Download” page provides a text box for a file name, and the **Download file** button downloads the coded data. Use the *Start new data file* link to re-start the coding and the *Continue coding with this file* link to continue adding to the existing records.
  - The .txt file contains the variable names in the first line.
  - If the file name does not end in “.txt”, this will be added.
9. To quit the program, just close the window.<sup>10</sup>

---

<sup>10</sup>This, it turns out, is a HTML/Javascript security feature which prevents rogue websites from closing windows unless they have created the window.