`actor-annot-lite.py` documentation

Philip Schrodt. (schrodt735@gmail.com)
Parus Analytics LLC

Last update: 03 Dec 2021

`actor-annot-lite.py` is a Python program for annotating event files with sentence segmentation, actor, recipient, and location spans. It is designed to use minimal machine resources—in the Macintosh OS-X system, it takes about 6Mb of memory versus the 350Mb required for a single Google Chrome web page—and does not require connection to a server. The program is implemented using the C/Python "ncurses" terminal emulation and uses a small set of keyboard commands rather than a mouse,[1] and is designed to run on a laptop. Like on an airplane.

To run:

python3 `actor-annot-lite.py <filename> <coder>`

> <filename> is the file in the NGEC annotation formats that you want to edit.
>
> <coder> is an optional identification of the person doing the coding, typically your initials. If not present it defaults to PAS

The program uses a file named *actorannot.filerecs.txt* to keep track of where you are in various files; if you haven't edited a file before it will start at the beginning.

The output of the program is in a JSON file with the name of the form

> *actor-<filename>-<coder>-<timestamp>.json*

A new file, with a distinct timestamp, is produced each time you run the program. If the program crashes, all of the saves you've saved using "=" should be preserved. The new file location will not be: records you've already coded by be skipped using "X".

Unless otherwise noted, the program responds to single keystrokes and does not require a <return>. If a *string* is entered—these are the target prompts—then it is ended with <return>. If the program responds with a beep, the keystroke doesn't work in the current context: sometimes an explanation is provided, sometimes not.

A short video is available demonstrating the use of the program; contact schrodt735@gmail.com for a link.

[1] Obligatory xkcd reference: https://xkcd.com/378/

Primary modes and workflow

S: segment sentences

/: segment sentences ending in . automatically, starting from the last selection

digits 1 to 0: select a sentence to work on

A, R, L: actor, recipient, location selection

Workflow: Typically, S will be used initially to segment the sentences—usually only the first sentence needs to be manually selected in order to skip the dateline, then if the text has good segmentation separated by ". ", the remainder are selected automatically using the "/" option; otherwise sentences which can be annotated are selected manually (not infrequently, only the first sentence will qualify). Following segmentation, A, R, and L will be used to annotate sentences which contain information.

Primary mode details

S:

The program responds with the prompt "Enter sentence target or delimiter:", which takes a string. If a delimiter— (period, comma, colon, double-quote)—is entered, the text from the last location +1 to that delimiter will be selected. Otherwise the first word beginning with the [case-sensitive] string will be selected, then further words are selected using <space>. If a word further in the text is where you want to start, use "F" to move forward. <return> is used to accept the sentence, which then appears in the sentence list.

/:

The command "/" (slash) automatically segments the texts into sentences where these end with period-space. It generally works, but the two common sources of failure are:

- there is an abbreviation, typically "U.S. "
- a sentence ends with '."' (period-doublequote)

If the incorrectly-segmented sentences have information that could be annotated, either do the segmentation manually (typically with multiple uses of S-. to skip over abbreviations, and S-" to get the sentences ending in "), or just manually segment the sentences that can be annotated: when writing the output file, only annotated sentences—that is, those with at least one actor, recipient, or location—are included.

If there is no dateline, or the dateline occurs at the end of the story (yes, this happens) "/" can be used on the entire text without selecting the first sentence.

digits:

Sentences in the sentence list are selected using the numerical keys 1 through 0, where 0 corresponds to "10", the maximum. A ">" will appear in front of the selected sentence and subsequent commands will work with this.

A, R, L:

These select the texts for the actor, recipient, and location respectively,

- Responds with the prompt "Enter <type> target:"
- Text is selected using the same navigation as with sentences, except the delimiter option is not available.
- Multiple actors can be selected, though if these occur in a compound it is okay to select the entire compound phrase.

Sentences do not need to have all of these elements filled in, and if a sentence has none of them—this is quite common—just ignore it.

Starting point commands

target <string><return>: Start from the first word that *begins* with <string>: note that this doesn't need to be the entire word

target <return>: Start from the next non-blank word after the last selection (that is, nothing is entered)

F: go to next instance of target: this is used if the target matches more than one word. Wraps to beginning of text if no target past the current one.

; [S mode]: Go to first word after the dateline delimiter in ["- ", "— ","--", ") "]. If there are multiple delimiters, this can be used repeatedly to get to the correct one. (Note that this works differently in A-R-L mode)

A, R, L commands

These are single-character strings in response to the prompt; "[" and "]" also work as keystroke commands after an A-R-L mode has been selected. Python's `istitle()` is defined as "True if all words in a text start with a upper case letter, AND the rest of the word are lower case letters, otherwise False. Symbols and numbers are ignored."

<return> (that is, no character): Select first word in sentence

[: select *second* `istitle()` word in sentence (used in situations like "The French ambassador…)

]: select until a word is not `istitle()` or in the set ["of", "for", "and"]

;: select the first word after `The`

-: Remove last item from A, R, L lists; this can be used as a reset for these selections

Selection commands

space: select next word

B: remove last word from selection

'.',',',':', '"' (period, comma, colon, double-quote): select until a word ends with the delimiter (note that since words are broken on spaces, this means the delimiter must be followed a space so, for example "some text:more text" will not trigger ":"). This can also be used in primary mode to select sentences without first using S

Moving between records

=: save and move to next record

X: skip the record

Q: quit

Miscellaneous commands

!: reset current record, moving all sentences and annotation: which is useful when sentence selection doesn't work or the highlighting is of

@: same as ! except it leaves any existing sentences alone

C: add a coder and time-stamped comment to the `comment` field of the record; this is concatenated at any existing comment if such exists, otherwise a `comment` field is created.

Annotation comments:

1. Getting the first sentence without the dateline—if present; some articles have no dateline—will be useful to future work on identifying these, which have a wide variety of different formats in Factiva and OSC. For example:

- `Gaza, Oct 23, 2012 (Tunis Afrique Presse/All Africa Global Media via COMTEX) --`
- `TOKYO, Aug. 19 --`
- `SINGAPORE -`
- `VIENTIANE. July 7 (Interfax) -`
- `[Interfax AutoSelect] TBILISI. April 7 (Interfax) -`
- `PRETORIA, Oct 15, 2007 (AFP) -`
- `TEXT: Source: Xinhua| 2020-07-16 19:10:26|Editor: huaxia  Video Player Close  JUBA, July 16 (Xinhua) --`
- `"Text of report in English by Polish national independent news agency PAP\nWarsaw, 9 September:`
- `"This product may contain copyrighted material; authorized use is for national security purposes of the United States Government only. Any reproduction, dissemination, or use is subject to the OSE usage policy and the original copyright. [Computer selected and disseminated without OSE editorial intervention]`

2. Actors in compound phrases, e.g. (this example is real…)

```
Russian President Vladimir Putin, Kazakh President Nursultan
Nazarbayev, Chinese President Hu Jintao, Tajik President
Emomali Rakhmonov, Uzbek President Islam Karimov and acting
Kyrgyz President Kurmanbek Bakiyev are taking part in the
summit
```

can be entered as a single entity: this is faster than selecting each individually and we'll need to be experimenting with how to deal with compounds anyway (and in any case, we can parse these to get the individual entities). Similarly, include titles and any other identifying information:

- `Parliament Acting Speaker Ali Iliksoy`

- `Alice Wells, principal deputy assistant secretary for the Bureau of South and Central Asian Affairs of the U.S. State Department`

- `The Head of the EU delegation to Nigeria, Mr. Santiago Ayxela` as we'll be experimenting with how much of this to pick up.

3. If a sentence contains two events with different actor/recipient pairs, include it twice in the original sentences, and code these separately. Similarly, if the information required to code an event spans multiple sentences, include both of these as a segmented "sentence" and code this as a unit. In both these cases, we'll be sorting this out further along in the process.

Output files

This is very much an "upstream" program where the output will be processed for later analysis or, as likely, as input to other programs. The output is primarily the original record, with the addition of two fields:

`annotstatus`: "accept" for "=" and "reject" for "X"
`sentences`: a list of the annotated sentences, where each item has the the structure
{
`actor/recip/locatAnnot`: {
    `span`: list of pairs of begin/end offsets,
    `text`: list of texts
    }
`sentStart`: sentence beginning offset
`sentEnd`: sentence end offset
`sentText`: sentence text
}
Offsets are in characters relative to the record's "`text`" field. Only sentences where at least one field in `actorAnnot, recipAnnot` or `locatAnnot` have information are included.

Output is the unnamed but very human-readable JSON produced by

```
fout.write(json.dumps(record, indent=2, sort_keys=True )
+ "\n")
```

which is distinct from the now-standard JSONL format of one JSON record per line, which at present is also the input format. This is easy enough to read but requires slightly more code than `json.loads()` : contact me if you need assistance here.

Status of program:

This has been developed and tested under the Macintosh OS-X 10.13.6 system. It should work under Windows and Linux but may require modifications to recognize the <return> key.

Error of the form (Mac OS-X)

```
 File "coder-PITF-PROT.py", line 1169, in <module>
    curses.wrapper(main)
  File "/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/curses/__init__.py", line 94, in wrapper
    return func(stdscr, *args, **kwds)
  File "coder-PITF-PROT.py", line 565, in main
    stdscr.addstr(COMM_Y, TERM_WID - 32, "Serial:  " + str(krec) + "   ")
_curses.error: addwstr() returned ERR
```

or [insert Andres Window]
just means you need to enlarge the window