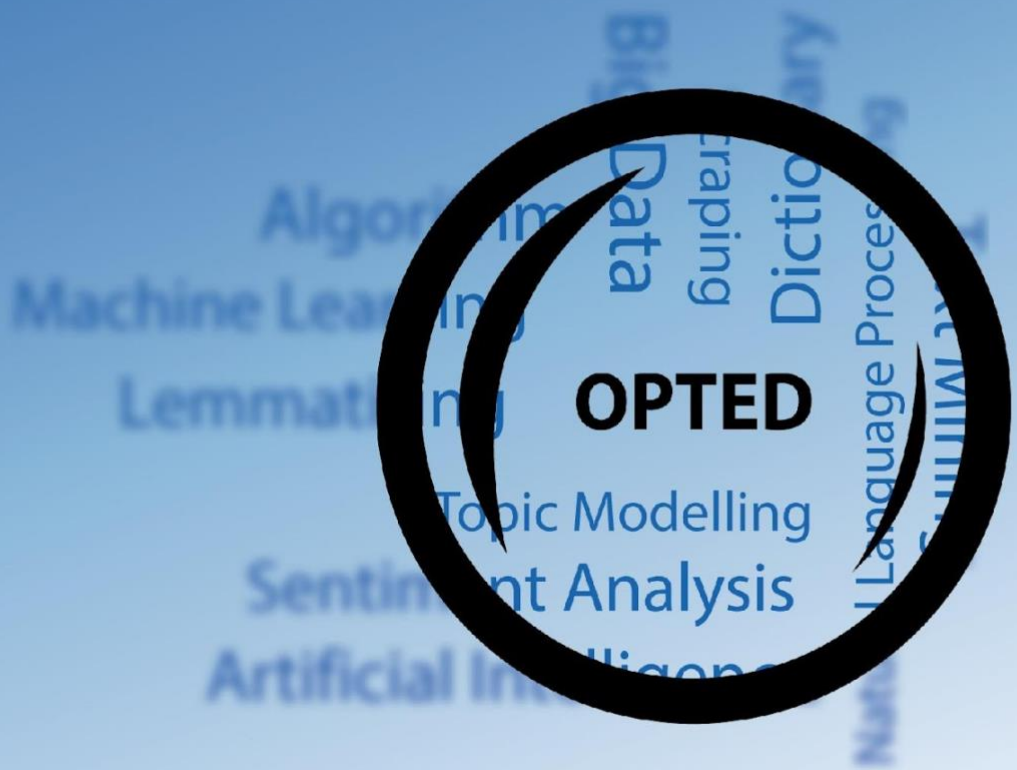
OPTED

Manual Annotation Interface

Kasper Welbers, Wouter van Atteveldt, & Farzam Fanitabasi



Disclaimer

This project has received funding from the European Union's Horizon 2020 research & innovation programme under grant agreement No 951832. The document reflects only the authors' views. The European Union is not liable for any use that may be made of the information contained herein.

Dissemination level

Public

Type

DEC

OPTED
Observatory for Political Texts in European Democracies:
A European research infrastructure

Manual Annotation Interface

Deliverable 7.2

Authors: Kasper Welbers[1], Wouter van Atteveldt[1], & Farzam Fanitabasi[1]

[1] Department of Communication Science, Faculty of Social Science, Vrije Universiteit Amsterdam

Due date: December 2021

Executive Summary

The overall objective of WP7 is to establish routines and protocols to standardize the pre-processing of text, pending on the source and usage purpose. This work package focuses on assessing and providing prototypes of open science and open data structures in terms of data storage.

As the second step to achieve this objective, D7.2 introduces a novel manual annotation interface. The necessity of such a tool stems from the ongoing progress in automated content analysis methodology. This progress has enabled Computational Communication Science (CCS) researchers to outsource much of the large-scale and time-consuming efforts in content analysis to computers. Indeed, CCS aims at simultaneously making the content analysis work cheaper, faster and easier, as well as allowing the researchers to study communication at a much larger scale and from different angles than manual content analysis alone would have permitted. Nevertheless, this increase in using computers for content analysis does not alleviate the need to perform copious amounts of manually coding to develop, calibrate and validate the methods. Human coding serves as a gold standard to enable and improve automated annotations, and allowing researchers to easily conduct such manual coding and to ensure high-quality manual data, an integrated interface shall be highly useful for the OPTED community and text analysis researchers generally.

To this end, this deliverable introduces a novel manual annotation interface (called CCS Annotator). CCS Annotator is designed and implemented with the aim of making it easier to create coding jobs, disseminate the task to coders, and store the resulting annotation on political texts. This novel annotation interface is targeted specifically at the CCS community and is designed to streamline the set-up and deployment of the annotation tasks for the CCS field. Additionally, CCS Annotator can facilitate the development of reusable and shareable codebooks. CCS Annotator can be used by itself using only a standard web-browser, requiring no installation for either the researcher or coders. An additional server can be installed for fast and efficient distribution of jobs and collection of results. CCS Annotator supports various annotation modes, ranging from expert tasks such as labeling specific words and phrases, to simple crowd coding tasks like Tinder style swiping on mobile phones. Furthermore, CCS Annotator paves the way for a more systematic investigation of which codebooks, coder recruitment strategies, and which coding interfaces actually work best for what tasks. This facilitates research and studies into what coders find intuitive, how coding jobs should be defined, and facilitates the standardization of such tasks.

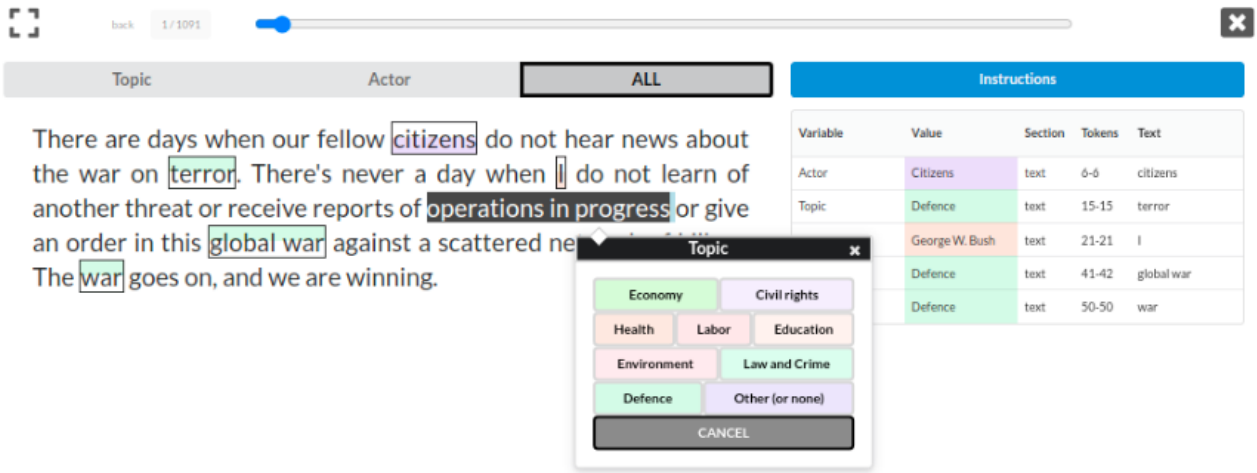
This deliverable consists of the CCS Annotator codebase - two frontend modules for the coding manager and coders, and a backend module for connecting it to AmCAT 4.0 (D7.1) - published via the OPTED website and publicly accessible on the GitHub repositories. Additionally, a technical document outlining the technical design of the CCS Annotator, and two user manuals (one for the coding manager and one for the coders) accompany this deliverable, helping users understand and utilize this tool as smoothly as possible.

1 CCS Annotator: A Manual Annotation Interface

The ongoing progress in automated content analysis methodology has enabled CCS researchers to outsource much of the heavy lifting in large scale content analysis to computers. Nevertheless, this does mitigate the need to perform copious amounts of manually coding to develop, calibrate and (crucially) validate the methods. It can be argued that the fundamental task of Computational Communication Science (CCS) is to both make content analysis cheaper, faster and or, and to allow the study of communication to happen at a much larger scale and from different angles than manual content analysis alone would have permitted.

Previous studies in CCS have emphasized the importance of rigorous manual coding, and the challenges which arise from using off-the-shelf dictionaries without calibrating and validating them with respect to the specific data on which they are used (Boukes, van de Velde, Araujo, & Vliegenthart, 2020; Chan et al., 2021; van Atteveldt, van der Velden, & Boukes, 2021). These challenges are amplified in case of unsupervised techniques, such as topic modelling, which the difficulty of validation has sometimes led to lack of perceived importance. Besides validation, more manual coding is also often the most effective way towards better results. It is well understood that the quantity and quality of training data, and the relation of this training data to the actual data of interest, is critical to the performance of machine learning algorithms and methods.

Figure 1.1 CCS ANNOTATOR: ANNOTATION MODE



To this end, this deliverable introduces a new, open-source annotation tool tailored to the relevant OPTED stakeholders and the wider CCS community. The goal of this tool is simple: by making the specific manual annotation tasks that the CCS field requires easier and more efficient to set- up and deploy for researchers, and more intuitive and easily accessible for coders, we aim to improve the commitment to and efficiency of manual content analysis and hence for more valid CCS text analysis applications.

1.1 Use-cases and Functional Requirements

To this end, CCS Annotator supports various use-cases and operational modes. It can be used in crowd-sourcing scenarios where the aim is to collect annotations from a large number of coders, and generate a consensus corpus accordingly. CCS Annotator can also be used in expert coding scenarios, where a predetermined group of expert coders (often with field knowledge) are given the task to annotate the text. To further facilitate the annotation and coding process, CCS Annotator also support two different modes of annotation: (i) The annotation mode which collects input codes from users for specific tokens/spans of text, and (ii) The question mode, where the coders answer a series of predefined questions about the text. These two modes are further explored in Section 1.3.

Moreover, manual annotation can have a great impact on the text analysis process. Often acting as a bottleneck due to its time consuming and effort intensive nature. A manual annotation project usually involves various steps, including defining annotation schema (codebook), creating an annotation guideline (instruction for coders), collective and pre-processing document according to the task, training (if needed) experts for the annotation task, and finally building a consensus corpus. Consequently, the usability and completeness of a tool have a direct impact on the annotation process and can either accelerate or slow it down. To this end, during the process of designing and implementing the CCS Annotator, a set of functional requirements have been specified. The requirements are:

- Specifying the Fields to Code: CCS Annotator provides the functionality to indicate a predetermined set of fields for each text unit to be annotated.
- **Specifying the Possible Code Values:** This functionality restricts the possible values for codes the coder can input. This is particularly important in more open-ended coding tasks.
- **Specifying the Annotation Context:** This pertains to the ability of the coding manager (the entity responsible for setting up the coding task) to specify the amount of extra text (context) which the coder should be able to see in order to annotate the given text unit.
- **Ease-of-coding:** As a critical requirement for scaling-up and facilitating the annotation task, CCS Annotator is designed to provide an intuitive interface for the coders, as well as an easy installation and setup process for the coding managers.
- **Mobile Interface:** Given the continuous increase in the number of coders who perform their assigned tasks on their mobile devices, it is very important for manual annotation tools to support mobile

devices and adapt to their screen size. CCS Annotator has a full screen/windowed mode which is specifically designed for this purpose.

- **Connection to Crowd-coding Platforms:** For many CCS Researchers who utilized a variety of crowd-coding platforms (e.g., Qualtrics), it is desirable to be able to pull and merge the coding results from these platforms, and/or connect them to their own manual annotation tool. CCS Annotator aims to provide this functionality for easily connecting to popular crowd-coding platforms.
- **Provenance:** In various CCS research scenarios, it is crucial to know and trace the source of the text, the date of publication, and all the pre-processing steps performed on the text. CCS Annotator provides the option to record all this information in the *provenance* field in the coding job definition.
- **Controlling the Coding Order:** In some CCS scenarios it is necessary to control the order in which certain text units are shown and coded by the users, to avoid information leakage, and biases to be introduced to the coding task. CCS Annotator provided the functionality to define the order, and the sampling mechanism for presenting the coding units to the coders.
- **Intercoder Reliability:** In many cases it is possible (and quite often observed) that coders may disagree with each other about the specific code/value for certain text units. To clarify and avoid such uncertainty to leak into the research output, measures such as *Intercoder Reliability* are utilized. By recording each coder's identity and annotation per text unit, CCS Annotator enables researchers to easily identify such cases and calculate the intercoder reliability.
- **Quality Control:** Lastly, to make sure that the annotations provided by certain coders are reliable, CCS researchers used *Gold Questions* (i.e., questions where the correct answer is known in advance to the coding job manager). The coders who fail to answer such questions correctly are emitted from the list of valid annotation providers. CCS Annotator provides the functionality to specify and check coders with these gold questions, and if needed revoke their access.

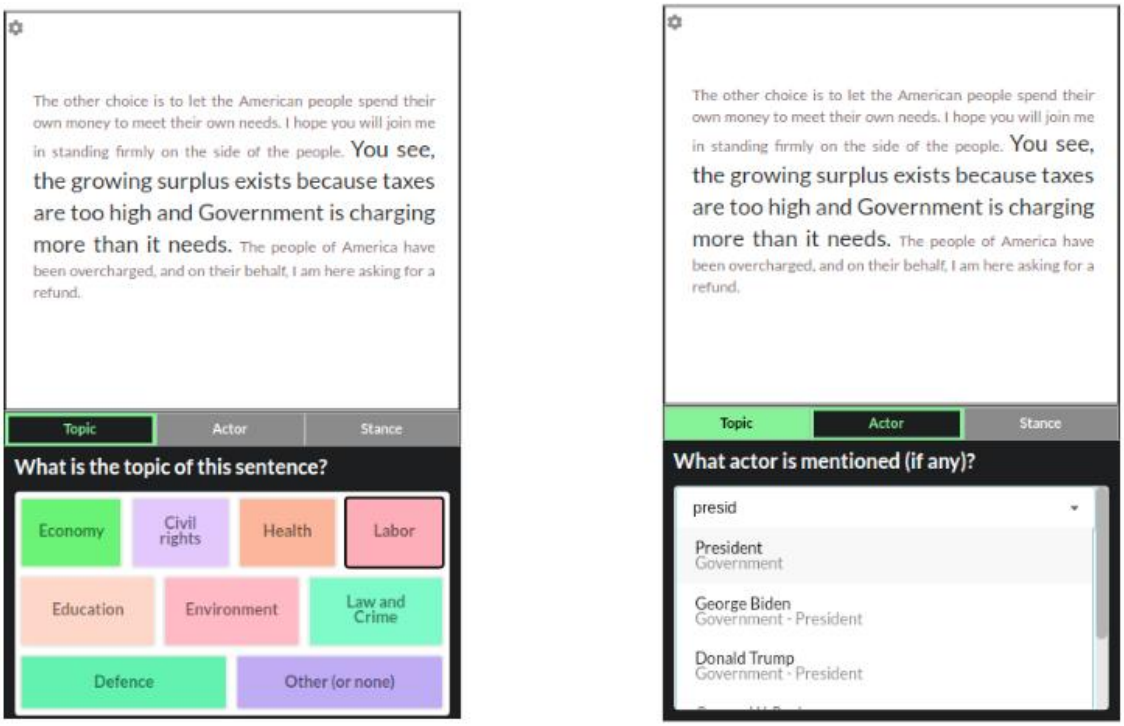
1.2 Relation to other Annotation Tools

There are various advanced annotation tools such as *brat*, *INCEpTION* or *Doccano*, that facilitate detailed coding at the level of words, phrases and relations. However, they are often complicated to utilize, due to being mainly developed for trained expert coders. While this approach is suitable for annotating complex linguistic features in a huge corpus, for many tasks in CCS, where it is only necessary to deploy a specific coding job to calibrate or validate the method for a certain analysis, this approach is not optimal. To alleviate this issue, crowd-coding platforms *lik,* and *Amazon Mechanical Turk* are designed and developed to address the need for fast and easy deployment of coding jobs with little to no training. Despite various challenges regarding how to most effectively employ crowd-coding, overall the prospects are positive that it can offer fast and good coding given "careful specification and design" (Lind, Gruber, & Boomgaarden, 2017; van Atteveldt et al., 2021). Nevertheless, their main limitations are that the annotation software of these platforms is often tied to the crowd-coding service and has limited customization options.

Furthermore, previous research has shown that CCS researchers and scholars often require a tool that is both powerful and can be utilized in crowd-coding scenarios. In such scenarios, the requirement is for the tool to be able to quickly deploy simple coding jobs with little to no training, but not always via a crowd-coding platform. Moreover, rather than using a completely anonymous crowd, CCS researchers would sometimes rather recruit students that have more affinity with our types of tasks, and that we could provide some more training if necessary. Other times there exist more complicated tasks that require labeling specific words and phrases, while keeping the task as simple as possible so that coders do not first need to learn how to use certain software.

Given the importance of manual coding in CCS, there is merit in developing a dedicated software. This also creates new opportunities for collaborating with regard to coding efforts. CCS Annotator is designed so that codebooks, and all other settings in creating a coding job, have a standardized format that can easily be reused and shared. This can foster collaboration and make coding efforts from different projects easier to merge

Figure 1.2 CCS ANNOTATOR: QUESTION MODE



(a) Button selection (b) Search with hierarchical codebook

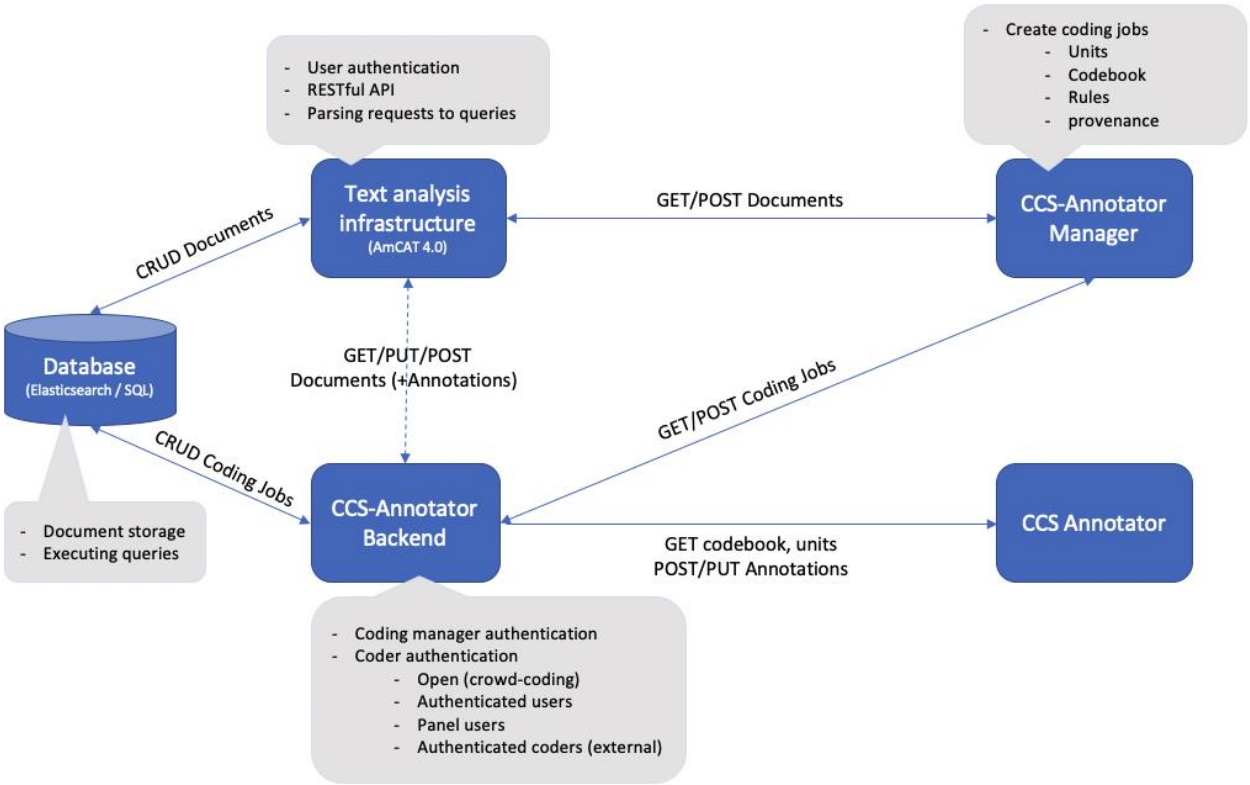
together. Furthermore, it paves the way for a more systematic investigation of what codebooks, coder recruitment strategies and coding interfaces actually work best for what tasks. For instance, for simple questions with at most 3 answers (e.g., yes/no/irrelevant), CCS Annotator has an AnnoTinder feature in which mobile users can answer with Tinder-style swiping. The coding task is not different from pressing buttons, but some people do find it more interesting, making it worth exploring if such formats have an effect on how intuitive and engaging a coding task is.

1.3 Annotation Modes

The current version supports two modes for coding texts.

- **Annotate mode**: Lets coders assign labels to specific words or phrases. Figure 1.1 illustrates what this looks like for a task where coders can label topics and actors. For simple tasks, coders can intuitively select words with their mouse or touchpad (mobile), but faster keyboard navigation is also supported. On selection, a popup lets users assign the label via a selected interface. For a small number of options this can be buttons. For a large number of options, such as the Comparative Agendas Project master codebook, this can also be a search box, and the codebook can have a tree structure to more easily search within categories.
- **Question mode**: Lets coders answer one or multiple (branching) questions about a given piece of text. Various formats for types of questions are available, and our goal is to add more and empirically test their efficacy for particular coding tasks. A key design principle for this annotation mode is that every type of question is fully and intuitively mobile phone compatible. Figure 1.2 illustrates what this looks like for a task where coders first answer whether/what topic is present in a sentence with buttons, and second whether/which actor is mentioned using a searchable list with a hierarchical codebook. Only if both topic and actor are present, a follow up question would inquire about whether/what stance the actor takes on the issue.

Figure 1.3 CCS ANNOTATOR MODULES

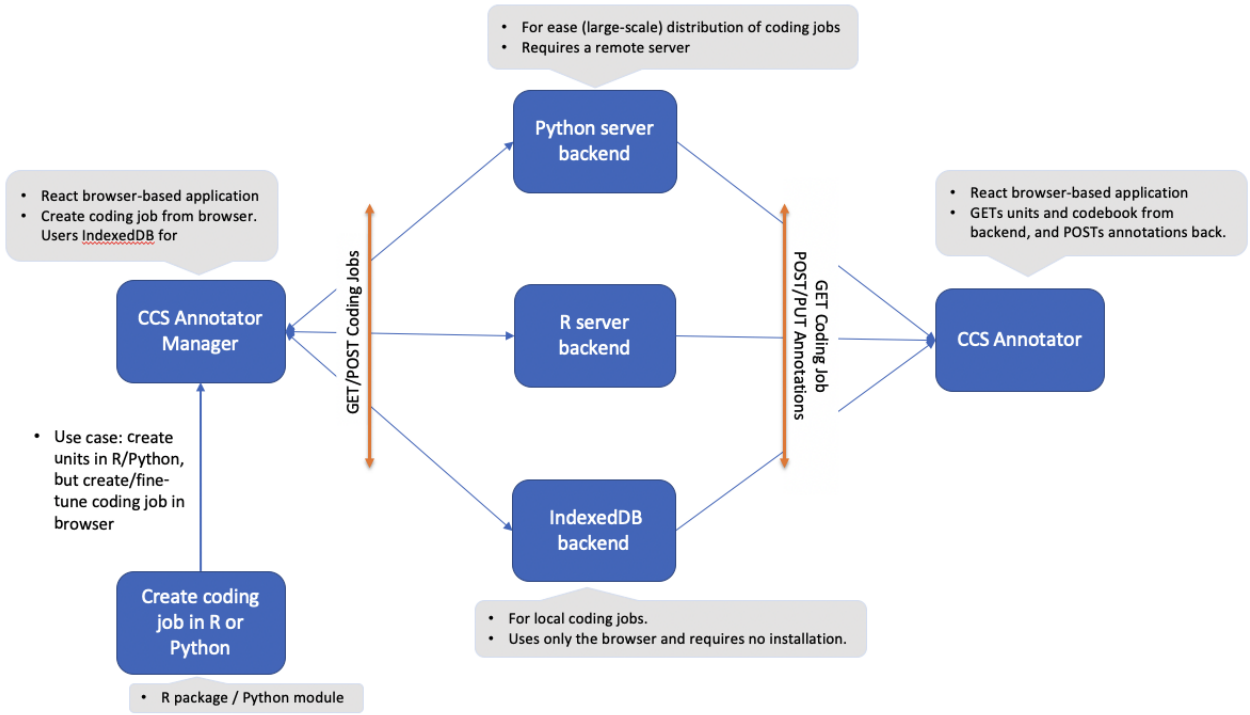


1.4 CCS Annotator Architecture & Modules

CCS Annotator consists of several modular components. Firstly, there are two separate frontend components: the *CCS Annotator*, and the *CCS Annotator Manager*. Both are completely browser-based applications developed in React. The CCS Annotator functions as a webpage where the coding itself takes place. Coders can either follow a link to a coding job, or upload a coding job from a file. The CCS Annotator Manager is an optional component in which researchers can create coding jobs. Alternatively, coding jobs can also be created in software such as R or Python.

These two frontend components actually do the vast majority of the work and can function without any backend. But by using a light backend on a server, coding jobs can more easily be distributed, and annotations can be collected and stored in real time. This approach of using a heavy front-end with a separate light back-end has two main advantages. Firstly, it makes it much easier for researchers to use the software, as the client does not need to be installed at all. The React application consists of HTML, CSS and Javascript that can be run by any regular browser. We host these files on a static file hosting like GitHub Pages for public access. This also means that coders never need to install anything, and can start coding directly via their desktop, laptop, smartphone or tablet. The second advantage is that the light backend makes it easy to create multiple versions for different purposes. We currently support the following use cases.

Figure 1.4 CCS ANNOTATOR DIFFERENT SETUPS



1.3.1 A Python-based Remote Server

This is the most powerful (and featureful) way to use the CCS annotator, in which it functions as a module for our larger text analysis infrastructure (D7.1: AmCAT 4.0). This facilitates reliable storage and easy sharing of the fruits of manual annotation labour. A schematic representation is presented in Figure 1.4.

1.3.2 A R-based Local Server

R is a very popular programming language for text analysis in CCS, so we provide a package to set up a local R server. The envisioned use case is that researchers working on a text analysis project in R can easily set up a coding job, do some coding, and immediately use the results. This would also enable more intuitive human-in-the-loop workflows.

1.3.3 Standalone version using the browser's IndexedDB API

For users that do not want to set up a server the tool can also be used as purely a browser application. It then uses IndexedDB API to store the data on the user's own device. Coding jobs and results can then be exchanged between researchers and coders via files. This is less convenient compared to using a server, but it can be used by anyone without having to install anything.

1.3.4 A custom backend

Power users can also create their own backend. For CCS Annotator this backend only needs to provide a REST api that supports GET codebook, GET units and POST/PUT annotations. To also interact with the CCS Annotator Manager, it only needs the additional GET/POST coding job.

2 Download and Installation

CCS Annotator and all its associated modules are freely and openly available through the OPTED website and GitHub repositories. The modules have their installation instructions described in detail in their README.me file.

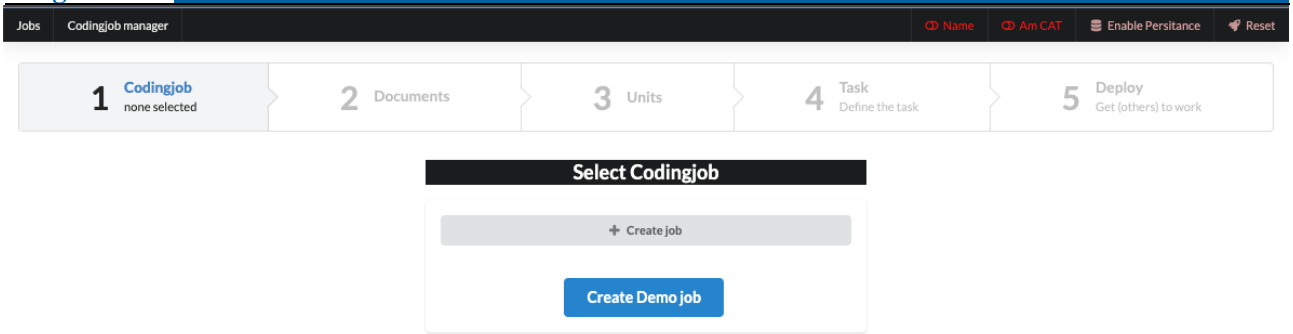
- CCS Annotator: https://github.com/ccs-amsterdam/CCS_annotator
- AmCAT Annotator Backend: https://github.com/ccs-amsterdam/amcat4annotator-backend
- AmCAT Core: https://github.com/ccs-amsterdam/amcat4
- CCS Annotator functional document: https://github.com/ccs-amsterdam/CCS_annotator/blob/main/README.md
- CCS Annotator managers guide: https://github.com/ccs-amsterdam/CCS_annotator/blob/main/MANUAL_manager.md
- CCS Annotator coders guide: https://github.com/ccs-amsterdam/CCS_annotator/blob/main/MANUAL_coder.md

3 Appendices

3.1 Appendix A: Screenshots of CCS Annotator Pages

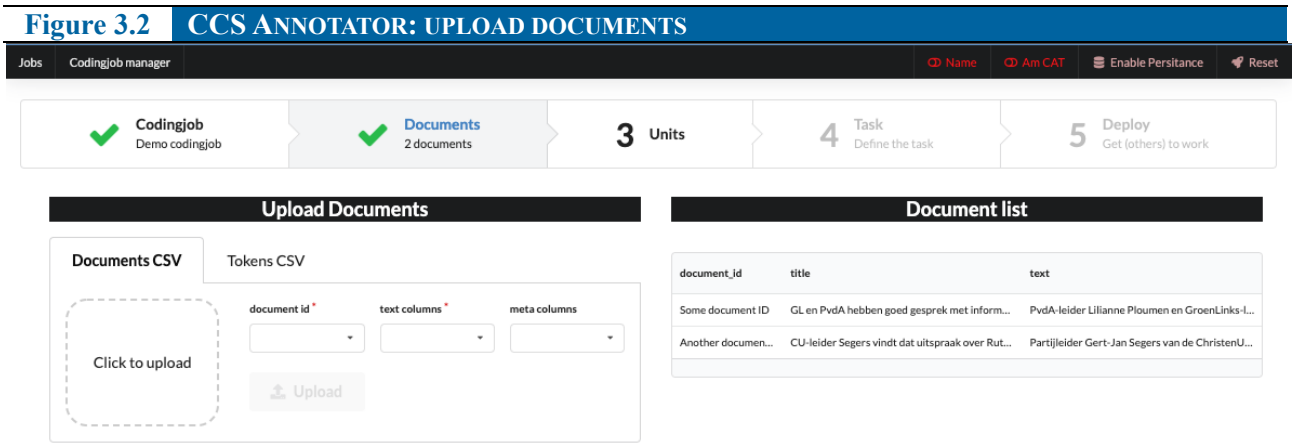
Create coding job:

Figure 3.1 CCS ANNOTATOR: CREATE CODING JOB



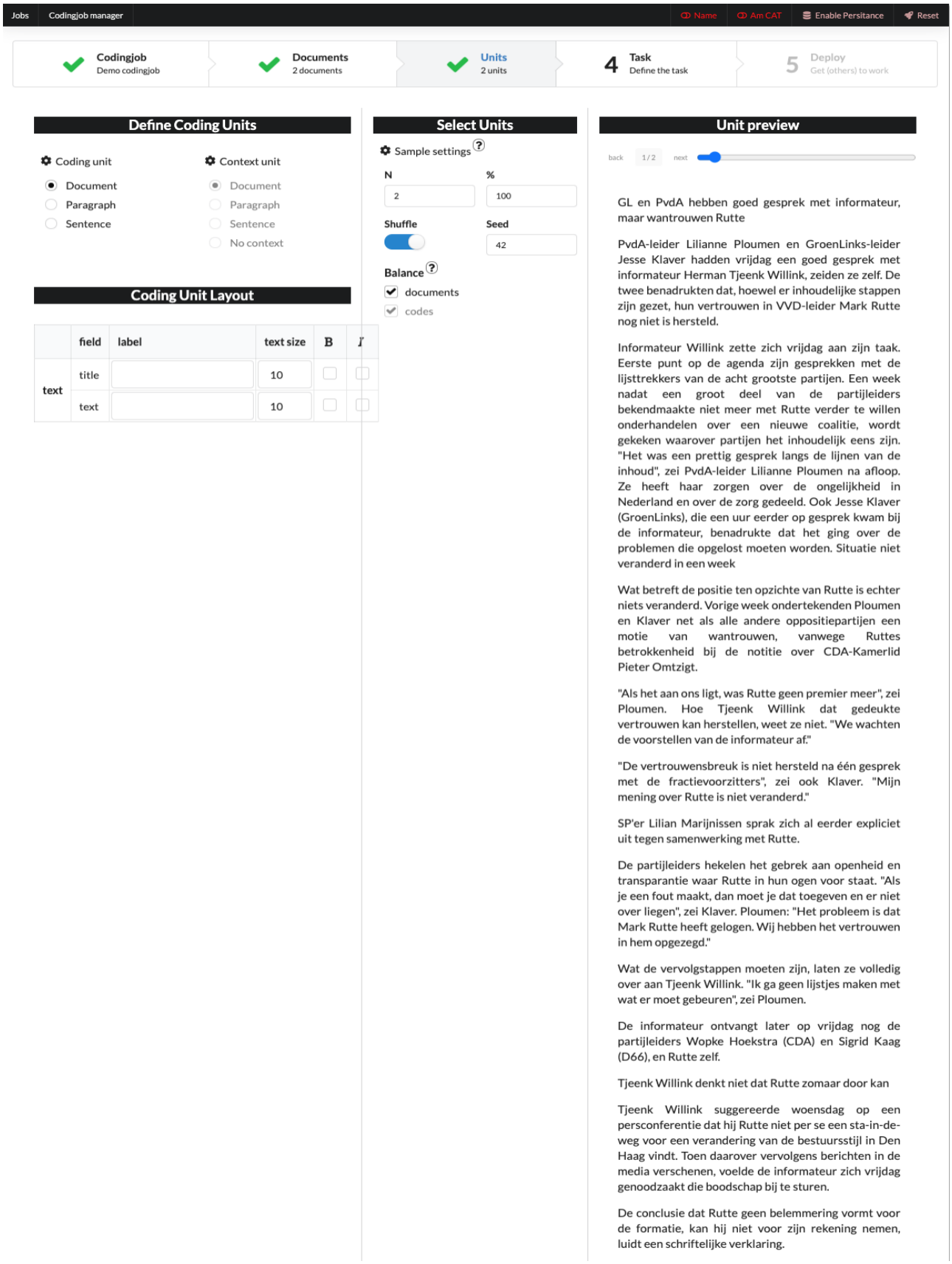
Upload documents:

Figure 3.2 CCS ANNOTATOR: UPLOAD DOCUMENTS



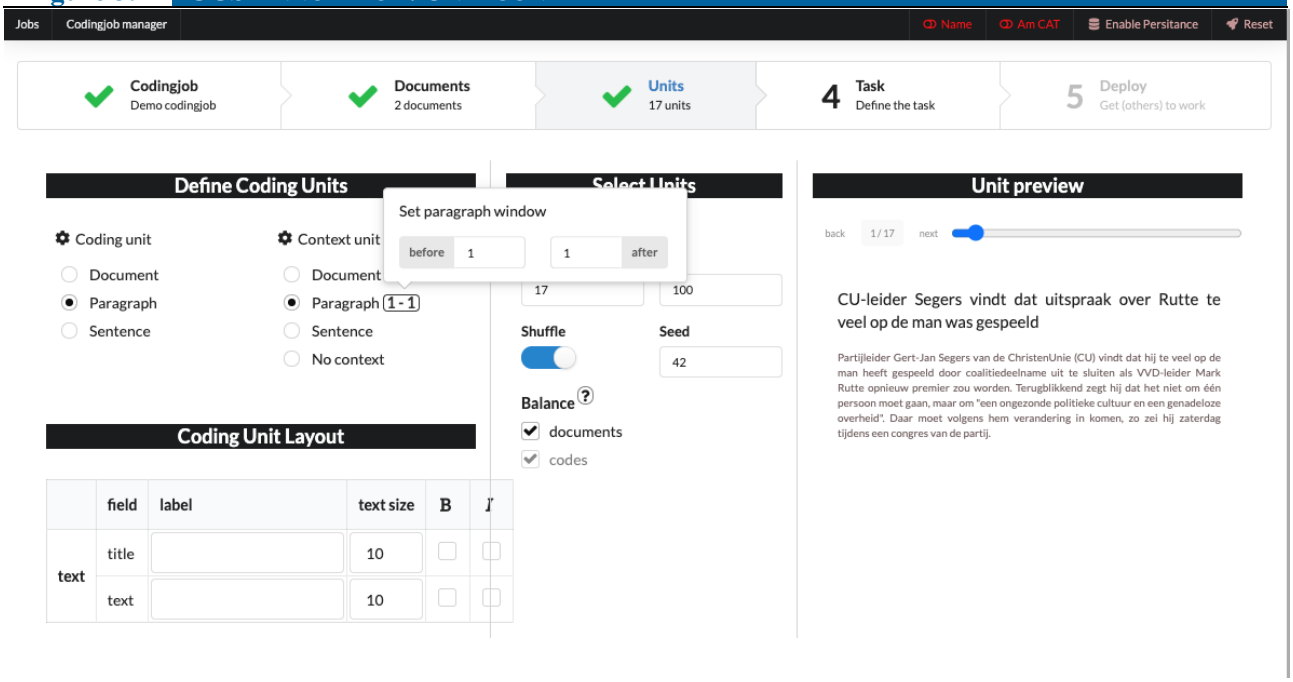
Define annotation units

Figure 3.3 CCS ANNOTATOR: UNITS



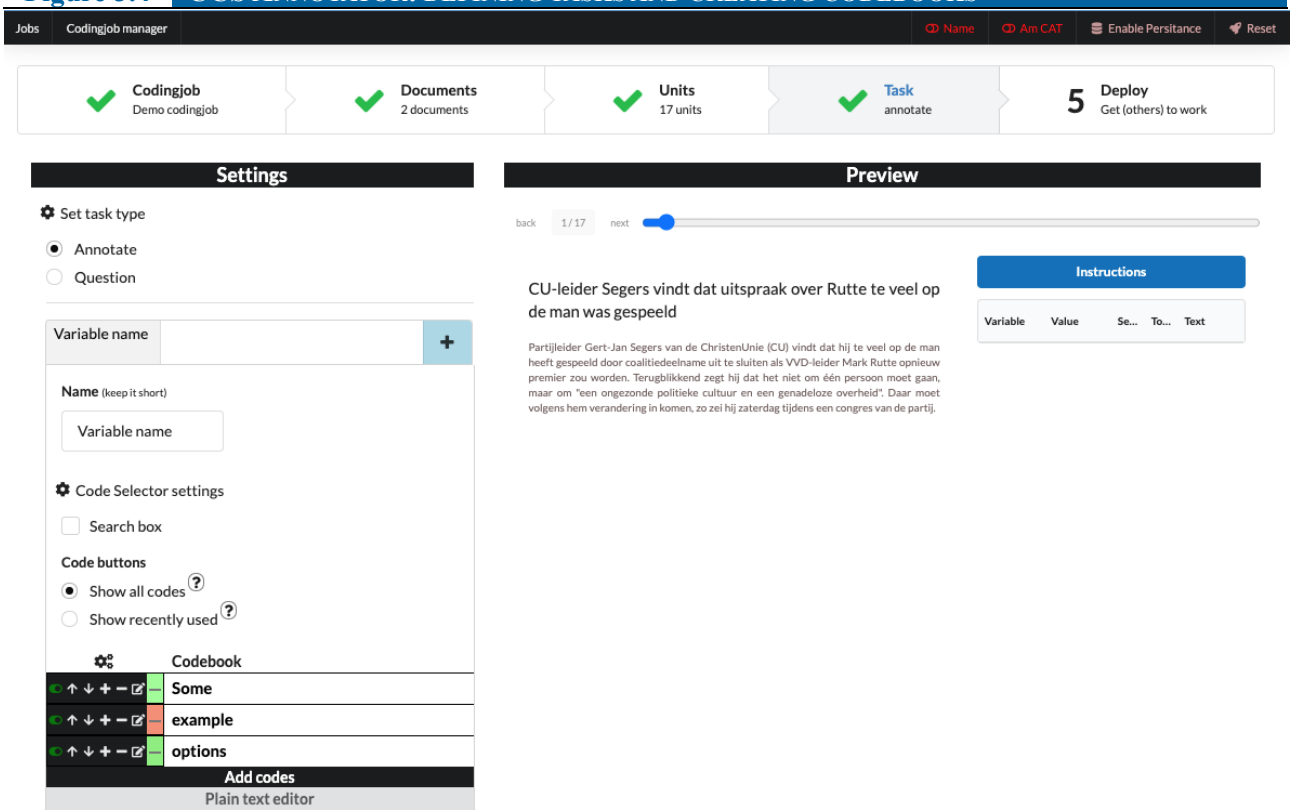
Specifying unit context:

Figure 3.4 CCS ANNOTATOR: UNIT CONTEXT



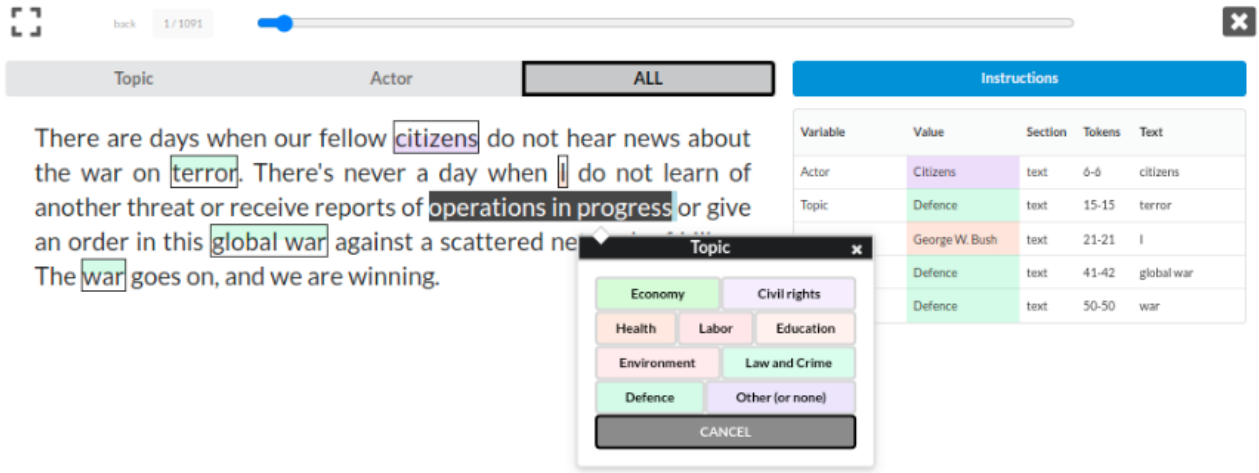
Indicating the task, codebook, and annotation mode:

Figure 3.4 CCS ANNOTATOR: DEFINING TASKS AND CREATING CODEBOOKS



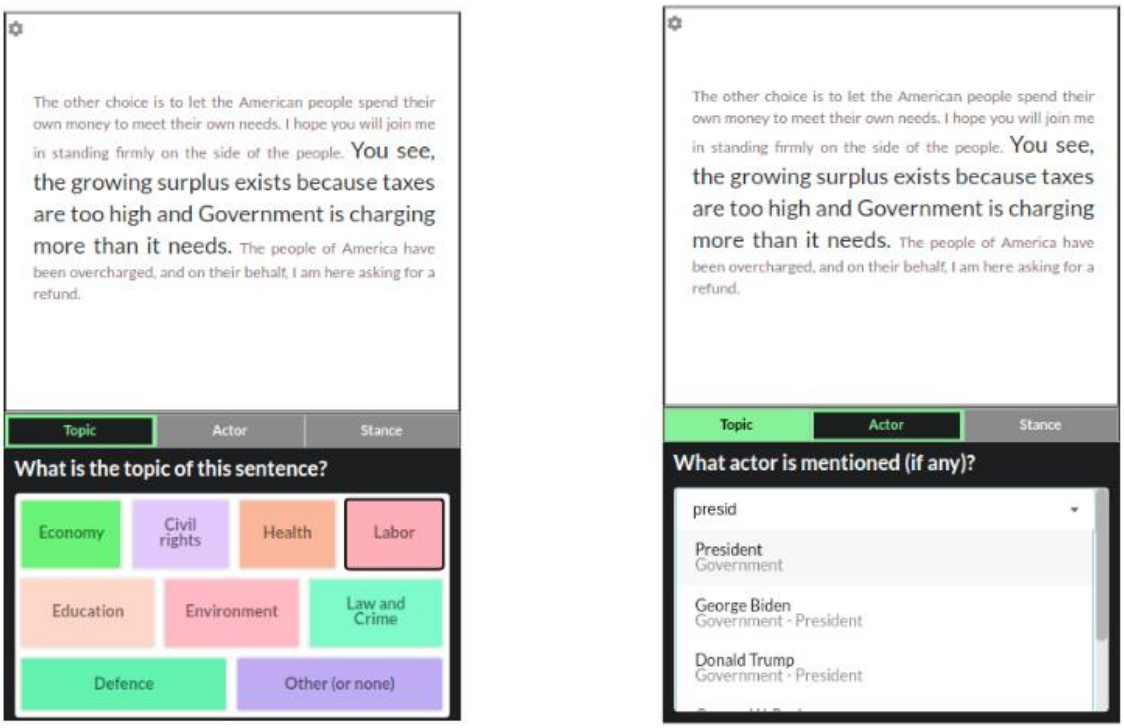
Annotation mode:

Figure 3.4 CCS ANNOTATOR: ANNOTATOR MODE



Question mode:

Figure 3.4 CCS ANNOTATOR: QUESTION MODE

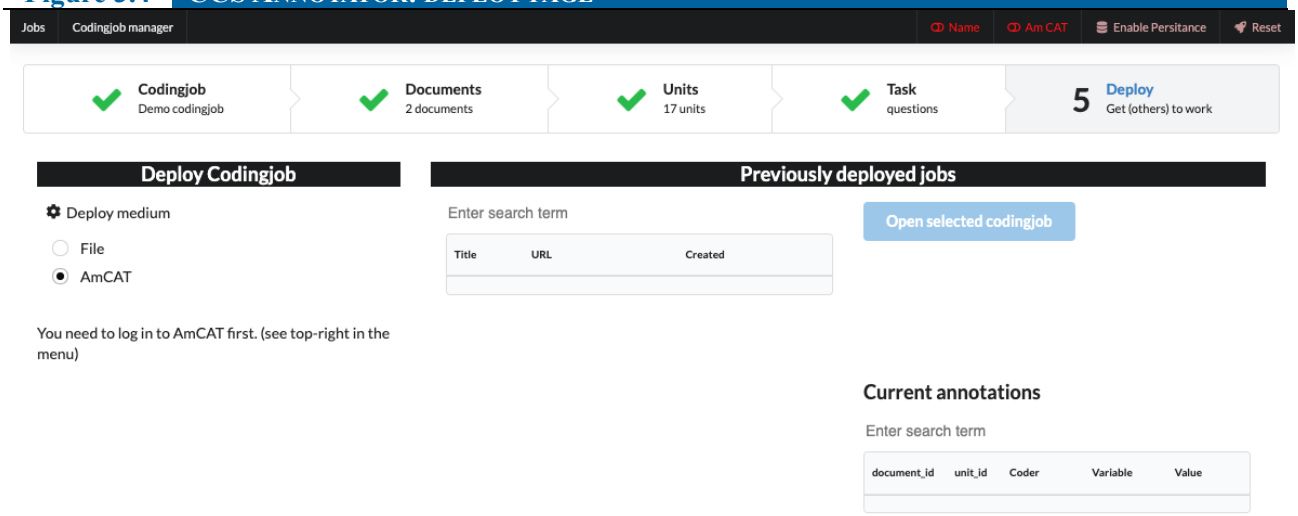


(a) Button selection

(b) Search with hierarchical codebook

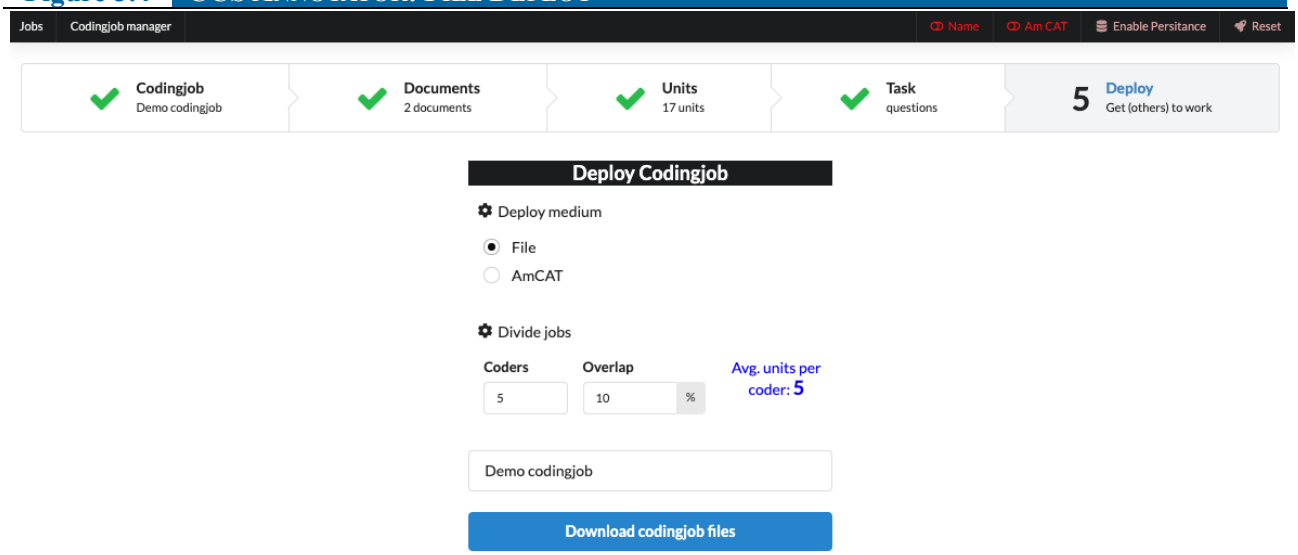
Deploy page:

Figure 3.4 CCS ANNOTATOR: DEPLOY PAGE



File deploy:

Figure 3.4 CCS ANNOTATOR: FILE DEPLOY



AmCAT deploy:

Figure 3.4 CCS ANNOTATOR: AMCAT DEPLOY