

Supplementary Information for MedTator: a serverless annotation tool for corpus development

Huan He, Sunyang Fu, Liwei Wang, Sijia Liu, Andrew Wen, Hongfang Liu*
Department of Artificial Intelligence and Informatics, Mayo Clinic, Rochester, MN, USA

*To whom correspondence should be addressed

Contents

CONTENTS	1
1 BACKGROUND AND TOOL DESIGN	3
1.1 SYSTEM ARCHITECTURE	3
1.2 PACKAGE REQUIREMENTS	5
1.3 COMPARISON WITH OTHER TOOLS	6
2 QUICK START	8
2.1 A MINIMAL ANNOTATION TASK	9
2.1.1 <i>Import schema and text files</i>	10
2.1.2 <i>Annotate files</i>	10
2.1.3 <i>Analyze the annotations</i>	12
2.1.4 <i>Export the annotations</i>	13
2.2 RUN YOUR OWN COPY	14
2.2.1 <i>Download Standalone Version</i>	14
2.2.2 <i>Fork Online Version</i>	16
3 ANNOTATION SCHEMA FILE	17
3.1 TASK NAME	18
3.2 CONCEPT NAME.....	18
3.3 CONCEPT ATTRIBUTE	19
3.3.1 <i>Attribute types</i>	19
3.3.2 <i>Special attribute</i>	20
3.3.3 <i>Default attribute value</i>	22
3.3.4 <i>Mandatory or optional</i>	22
3.4 SCHEMA SAMPLES.....	22
4 ANNOTATION DATA FILE	22
5 TOOL USAGE	24
5.1 ANNOTATION TAB	24
5.1.1 <i>Schema and annotation / text files import</i>	25
5.1.2 <i>Annotation file selection</i>	26
5.1.3 <i>Entity annotation</i>	27
5.1.4 <i>Document-level annotation</i>	29
5.1.5 <i>Relation annotation</i>	30

5.1.6	<i>Attribute modification</i>	31
5.1.7	<i>Hint Marks</i>	32
5.1.8	<i>Document display mode</i>	32
5.1.9	<i>Save annotations</i>	33
5.2	STATISTICS TAB.....	34
5.2.1	<i>Basic summary</i>	34
5.2.2	<i>Annotated tag statistics</i>	35
5.3	EXPORT TAB.....	35
5.3.1	<i>How to use the exported datasets?</i>	36
5.4	ADJUDICATION TAB.....	36
5.4.1	<i>IAA calculation</i>	37
5.4.2	<i>Download adjudication copy</i>	40
5.4.3	<i>Edit adjudication copy</i>	41
6	REFERENCES	43

1 Background and tool design

Natural language processing (NLP) and machine learning techniques have been widely applied in practice and research, which usually need to rely on high-quality annotated datasets.

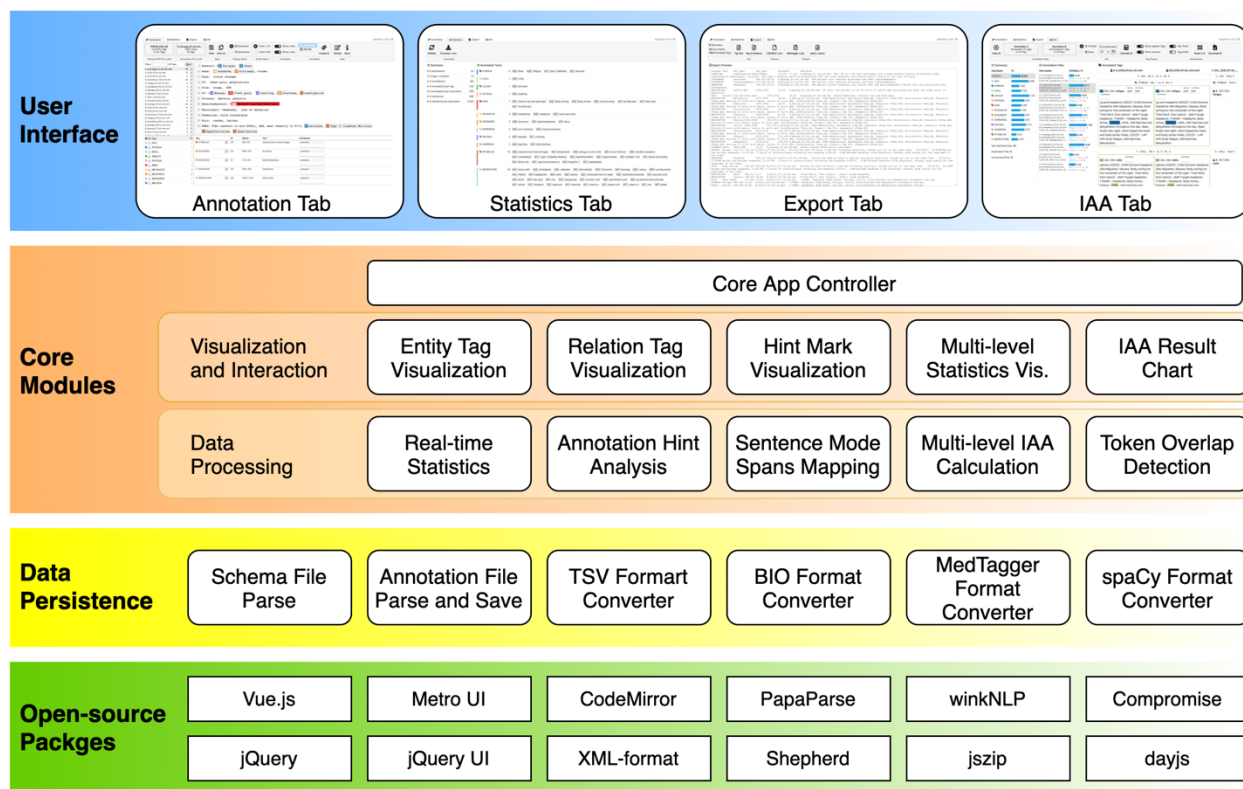
Therefore, manual annotation is required to collect additional information from document, and a suitable tool is needed to reduce the intensive labor work. To address this need, many text annotation tools have been developed for a variety of tasks, such as text classification, named-entity recognition, and sequence prediction.

However, while existing tools provide many powerful features to cover various needs in text annotation, it is still challenging for non-expert users or annotators to leverage these tools in their own research task. Based on the feedbacks from our domain experts and experienced annotators, we propose and implement MedTator to address the challenges.

1.1 System architecture

MedTator is implemented in pure frontend JavaScript with the annotation schema and files processed in client's web browser, which enables installation-free and cross-platform access for both administrators and annotators. Although MedTator is a pure frontend application that doesn't require any server components, its architecture design still follows the concept of the Model-View-Controller (MVC) pattern and a refinement of MVC, the Model-View-ViewModel (MVVM) pattern. The MVVM pattern helps to design a blueprint for developers to build frontend / client applications with more responsive user interaction and feedback, while avoiding costly duplication of code (e.g., DOM manipulation and CSS update) and effort across the overall architecture.

Due to the complexity of the annotation tasks, we designed four tabs and each tab focuses on a certain task to avoid users' recognition overload. Although the task for each tab is different, the functions and data structure used by each tab can be shared. Therefore, we leverage the features provided by the Vue.js and other packages to implement MedTator's architecture and the core functions needed for annotation tasks.



Supplementary Figure 1 Tool architecture based on open-source packages

As shown in the Supplementary Figure 1, the architecture of MedTator includes four layers, namely user interface layer, core modules layer, data persistence layer, and open-source packages layer.

The user interface layer contains the four tabs for the core annotation tasks, which are built based on Metro UI. It provides the similar experience of other well-known desktop applications. In the core module layer, we implement a Vue.js based core app controller to route the requests from users to the core functions, such as importing schema and annotation files and IAA calculation. As the intensive requirements of rendering tags and other visual effects, we implement some modules related to visualization. For example, when showing the relation tags, a polyline will be drawn on the editor in SVG (Scalable Vector Graphics) format to indicate the entities to be linked. To get the correct coordinates of the polyline in different display modes (i.e., document mode, and sentence mode), we developed modules to get the relative tag coordinates in the editor and map the coordinates to a SVG path in different coordinate

system. The data persistence layer can handle the requests of reading and writing files in various formats.

1.2 Package requirements

The functions and features of MedTator are based on many open-source packages, which are served from public free content delivery network (CDN) services. So that users won't need to install any runtime environment on server or client to use it (i.e., no need to install Java, Python, R, or any other runtime). A list of used open-source packages and their details are shown in Supplementary Table 1.

Supplementary Table 1 Open-source packages used in MedTator

Package Name	Version	Description
Metro UI 4	4.3.2	Metro 4 is an open-source toolkit for developing with HTML, CSS, and JS for quick prototyping responsive web pages.
jQuery	3.4.1	jQuery is a fast, small, and feature-rich JavaScript library for HTML document traversal and manipulation, event handling, Ajax, etc.
jQuery UI	1.12.0	jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library.
Vue.js	2.6.11	Vue.js is an open-source Model–View–ViewModel frontend JavaScript framework for building user interfaces.
jszip	3.2.0	JSZip is an efficient JavaScript library for creating, reading and editing .zip files with simple API set.
dayjs	1.8.36	Day.js is a minimalist JavaScript library that parses, validates, manipulates, and displays dates and times.
CodeMirror	5.62.0	CodeMirror is a versatile text editor implemented in JavaScript for editing code in web browser.
PapaParse	5.3.1	Papa Parse is a fast in-browser CSV (or delimited text) parser for JavaScript, which is reliable according to EFC 4180.
Shepherd	8.3.1	Shepherd is a JavaScript library for guiding users through the main features of a web application.
winkNLP	1.8.0	winkNLP is a JavaScript NLP library that supports stemmer, lexicon, tokenizer, lemmatizer, etc.
Compromise	13.11.4	Compromise is a JavaScript NLP library that supports sentence split, token normalization, named-entity recognition, etc.
xml-formatter	2.4.0	xml-formatter is a JavaScript library for converting XML into human readable format while respecting the xml:space attribute.

1.3 Comparison with other tools

According to the recent literature on annotation tools (Neves and Ševa, 2021), we selected and installed some highly-ranked or popular open-source text annotation tools to assess their availability and usability. In addition, we also assessed some tools that used in our previous research and practice. The results are summarized as follows.

Supplementary Table 2 Existing tools for text annotation

Tool Name	Type	System Requirements	Advanced Features
WebAnno (Eckart de Castilho <i>et al.</i> , 2016)	Web-based	Server: Java Runtime 8+ Apache Tomcat 8.5 MySQL Server 5+	Multi-user support, project and user management, progress tracking, pre-annotation
brat (Stenetorp <i>et al.</i> , 2012)	Web-based	Server: Linux- or UNIX-like server Python 2.5 +	Comprehensive visualization, search integrated, multi-language support, automatic annotation, collaboration, fully configurable, search
FLAT (Gompel and Reynaert, 2013)	Web-based	FoLiA Document Server and FLAT Server: Python 3 + MySQL, PostgreSQL, or other	User management, multi-configuration support, multi-perspective, corpus query
Anafora (Chen and Styler, 2013)	Web-based	Server: Linux or UNIX-like server Apache, Python and Django	Project and user management, schema design, adjudication,
BioQRator (Kwon <i>et al.</i> , 2013)	Web-based	Server code not available for local installation.	Literature search
PubTator Central (Wei <i>et al.</i> , 2013, 2019)	Web-based	Server code not available for local installation.	PubMed search, multi-format export
INCEpTION (Klie <i>et al.</i> , 2018)	Stand-alone Web-based	Standalone: Java Runtime 11+ Server: Java Runtime 11+ Apache Tomcat 9+ MariaDB Server 10.5+	Active learning, Wikidata or DBpedia data support, project and user management, multi-format support, text search, IAA
Label Studio https://labelstud.io/	Web-based	Python or docker environment	Team project management, multi-label, multi-media, REST API support
eHOST (South <i>et al.</i> , 2012)	Stand-alone	Java Runtime Environment	Pre-annotation, machine-assisted annotation with UMLS and SNOMED-CT API, adjudication

MAE (Stubbs, 2011; Rim, Kyeongmin, 2016)	Stand-alone	Java Runtime Environment	IAA calculation based on multiple algorithms, adjudication
---	-------------	--------------------------	--

As shown in the above table, although existing tools may provide powerful features to cover various needs of text annotation, they usually require users to install a runtime environment before annotators could start annotation. For example, most web-based tools provide project and user management for better authentication and multi-project support, which may be helpful for large annotation teams to work collaboratively. Therefore, a central database, such as MySQL and MariaDB, needs to be installed to save information related to permissions and project settings. Other features usually also need some packages to be installed. As a result, users must solve the installation issues related before the annotators could run any tool for an annotation task.

This installation issue seems to be due to the needs for various features, but in fact the root cause could be the lack of basic computing infrastructure and fundamental functions in web techniques in the past. For example, the project and data management for multi-user annotation usually requires centralized storage and authentication. In the past, these services are not available or not easy to setup for individuals or small teams. Nevertheless, as a benefit of the popularity of the cloud computing, this kind of service could be easily obtained and integrated into local machine from public cloud computing platforms, or own private cloud. Then, the tool itself could focus on its unique functions, and users could use their own local tools to manage data and project. Moreover, as the public cloud services become more popular, it is possible to develop, distribute, and evaluate a web-based application through public services to enable community engagement.

Another benefit is the evolving of HTML5 and modern web browsers. As the development of HTML5 techniques, the functions of modern web browser increase a lot. Especially for those features (e.g., local storage, complex visualization, NLP, machine learning algorithm, and in-memory database) which were only available in web plugins such as Adobe Flash, Microsoft SilverLight, and Java Applet, are embedded in modern web browsers as default abilities or available through public content delivery networks. These improvements greatly empowered

the development of the comprehensive web-based application. As a result, it is possible to build better tools based on these improvements to save time for users.

Therefore, we designed and implemented MedTator as a serverless application, which could run on public cloud services such as GitHub Pages or run locally as a standalone program. All the libraries needed could be loaded from public CDN services or local disk. Moreover, users' own server installation could be very simple, which only requires a few clicks on web pages, and it is optional. Users can also just download the standalone version and run it fully offline to avoid any internet access.

2 Quick Start

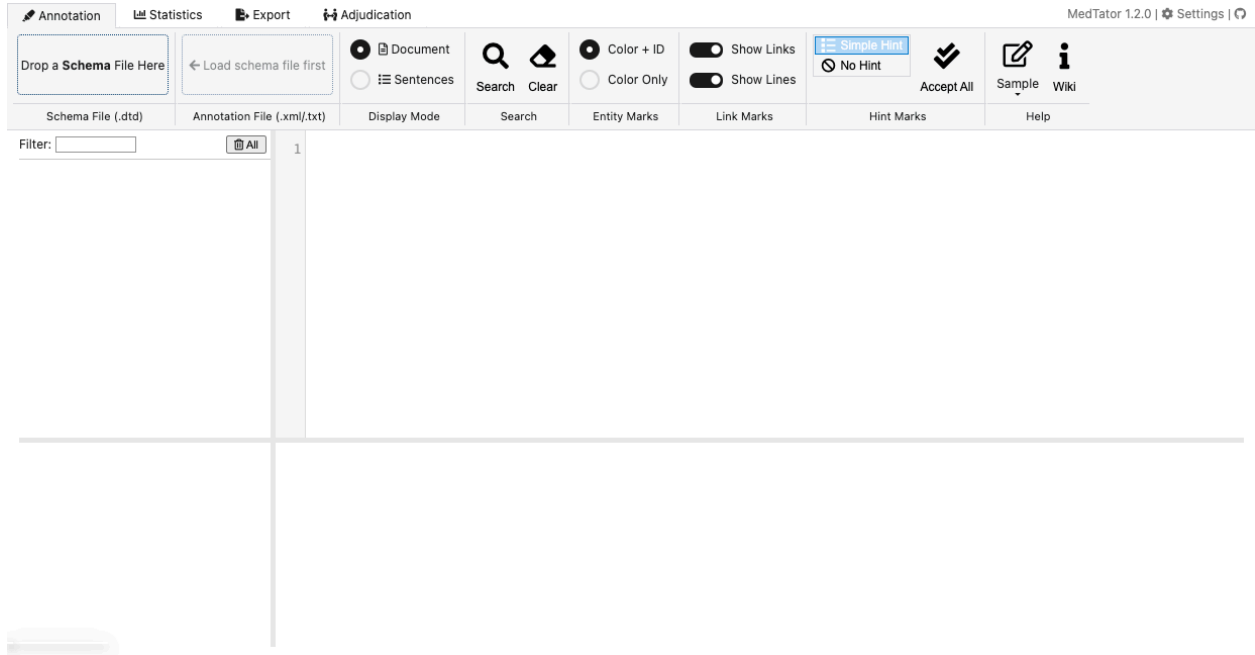
MedTator doesn't require any server or client runtime environment to be installed. Annotators could use the latest web browser to run MedTator, including:

- Chromium: <https://www.chromium.org/getting-involved/download-chromium>
- Microsoft Edge: <https://www.microsoft.com/en-us/edge>
- Google Chrome: <https://www.google.com/chrome/>
- Vivaldi: <https://vivaldi.com/download/>
- Opera: <https://www.opera.com/>

and other Chromium-based browsers.

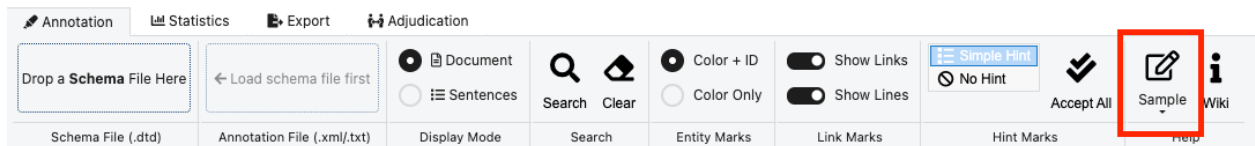
Due to the limited support to HTML5 in Microsoft Internet Explorer, MedTator couldn't run in Microsoft Internet Explorer. As we used the latest HTML5 File System Access API, the "Save" and "Save As" function may not be available in those web browsers that are not compatible with this API.

You could use the public version MedTator to start annotation quickly by accessing this URL: <https://ohnlp.github.io/MedTator/> . Or you can download the standalone version and use it offline. Then, the following interface would be displayed for you start. You could open your own schema file and text files for annotation.



Supplementary Figure 2 The initialized user interface of MedTator

If you don't have schema or text file yet, you could also try our online sample by clicking the "Sample" button in the menu as shown in the Supplementary Figure 3:



Supplementary Figure 3 The "Sample" button in the annotation tab for loading sample data

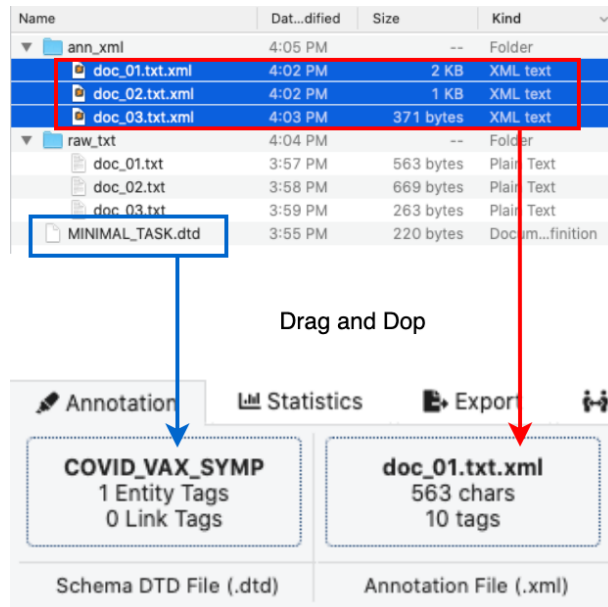
After clicking the "Sample" button, a sample dataset will be loaded to demonstrate the main features of MedTator, and you could explore all the four tabs (e.g., Annotation, Statistics, Export, and IAA) to try the functions in each tab. More details of the functions in each tab are described in the "Usage" section.

2.1 A minimal annotation task

In the MedTator repository, there is a `sample/` folder, which contains a minimal annotation task "MINIMAL_TASK" to demonstrate how to use MedTator to annotation. In this task, we only need to annotate the symptoms related to COVID-19 vaccination (e.g., headache, fever, pain, etc.) and there are only three text files, namely `doc_01.txt`, `doc_02.txt`, and `doc_03.txt`.

2.1.1 Import schema and text files

as shown in the following figure, you can drag and drop the `.dtd` file to schema file box (the details of schema `.dtd` file are specified in “Annotation schema file” section), and the 3 `.xml` files to the annotation file box (the details of annotation `.xml` file are specified in “Annotation data file” section). MedTator will read and load those files from your local disk directly to your web browser.



Supplementary Figure 4 a minimal annotation task – drag and drop the schema and xml files

2.1.2 Annotate files

As you can see, `doc_01` and `doc_02` have been annotated. You could check if there is any missing in these two files. After checking the first two files, only one file is left for you to annotate, which is the `doc_03`.

The screenshot shows the MedTator interface with the 'Annotation' tab selected. The main text area displays a paragraph from 'doc_01.txt.xml' with several words highlighted in blue boxes and labeled with 'SYMP' tags. The tags are: s10 dizziness, s11 lightheadedness, s2 headache, s3 headaches, s5 seizures, s6 no loss of consciousness, and s7 no convulsions. Below the text is a table with columns: Tag, ID, Spans, Text, and Attributes. The table lists three SYMP tags with their respective IDs (S1, S2, S3), spans (45-60, 81-89, 116-125), and text (lightheadedness, headache, headaches). The Attributes column shows 'certainty' as 'positive' and 'comment' as 'NA' for each tag.

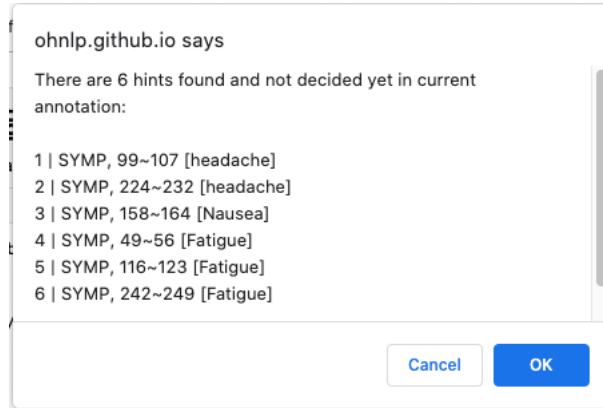
Supplementary Figure 5 a minimal annotation task – annotated doc_01

Then, click on the “doc_03.txt.xml” in the file list and the text will be displayed in the tag editor. As shown in the following figure, although we haven’t annotated this file yet, MedTator has already found some potential tags and shown the hints as dotted boxes based on the annotated tags in the doc_01 and doc_02. You could click on each hint box to add it.

The screenshot shows the MedTator interface with the 'Annotation' tab selected. The main text area displays a paragraph from 'doc_03.txt.xml'. The text is: "Patient took Janssen COVID-19 Vaccine EUA (3/6). Fatigue began 4 hours after vaccine. (3/6) Severe headache, severe Fatigue, moderate fever, mild aches, mild Nausea beginning the next day (3/7) and lasting until (3/8). Mild headache and mild Fatigue all day (3/8)". Several words and phrases are enclosed in dotted boxes, indicating potential tags. The table below the text shows 0 SYMP tags for this document.

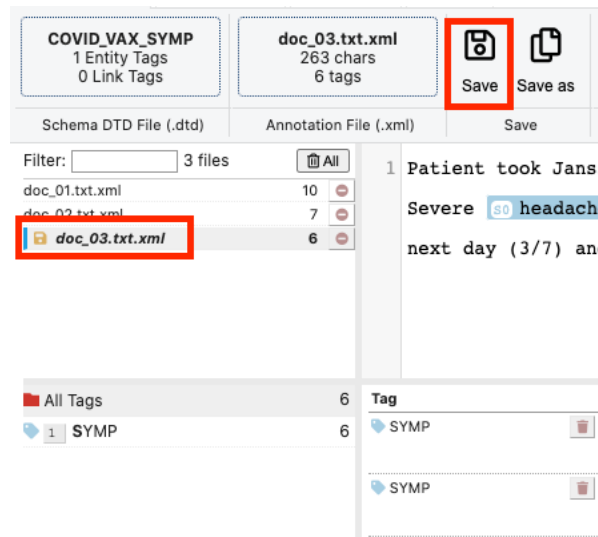
Supplementary Figure 6 a minimal annotation task - doc_03

Or you could just click the “Accept All” in the menu bar to accept all hints:



Supplementary Figure 7 a minimal annotation task – accept all hints

Once you update the annotations in any file, you will find a yellow disk icon will be displayed on the left of the file name, which indicates that this annotation file is changed (e.g., added new tags, deleted tags, or updated attribute values). You need to save this file otherwise the changes won't be saved. You could click on the yellow disk icon or the “Save” button in the menu to save the current annotation file.



Supplementary Figure 8 a minimal annotation task - save the file

2.1.3 Analyze the annotations

When the annotation is finished, we could check the overall statistical result on the annotated tags and the detailed list of all the texts by using the “Statistics” tab. For example, as shown in the left panel, the statistical result shows that there are 23 annotated tags found across 3 files.

Moreover, in the right panel, there are 17 unique tokens or phrases identified for the SYMP concept. And the count of each token or phrase and which file it comes from are also listed. You could check if there is any mistake in the annotation and go back to the file to correct it by clicking the file name.

The screenshot shows the MedTator interface with the 'Statistics' tab selected. On the left, a 'Summary' panel displays the following statistics:

- # of documents: 3
- # of tags in schema: 1
- # of annotations: 17
- # of annotations per tag: 3.00
- # of annotations per document: 5.67
- # of sentences: 18
- # of sentences per document: 6.00

The 'Annotated Tags' panel on the right shows 17 unique tokens for the SYMP concept, each with a count of 1 and a link to the source document:

- lightheadedness (doc_01.txt.xml)
- headache (doc_01.txt.xml)
- headaches (doc_01.txt.xml)
- neck pain (doc_01.txt.xml)
- seizures (doc_01.txt.xml)
- no loss of consciousness (doc_01.txt.xml)
- no convulsions (doc_01.txt.xml)
- bladder (doc_01.txt.xml)
- bowel incontinence (doc_01.txt.xml)
- dizziness (doc_01.txt.xml)
- Nausea (doc_02.txt.xml)
- Fatigue (doc_02.txt.xml)
- chills (doc_02.txt.xml)
- diarrhea (doc_02.txt.xml)
- dehydration (doc_02.txt.xml)
- head and body aches (doc_02.txt.xml)
- Night sweats (doc_02.txt.xml)

Supplementary Figure 9 a minimal annotation task – statistics on the annotated tags

2.1.4 Export the annotations

Once the analysis is finished, we could send the annotation files (e.g., the 3 .xml files) to downstream tasks directly. To streamline the data processing, MedTator supports exporting the annotation files to other formats used by downstream tasks. For example, MedTator could export all the annotated tags with the sentence context as a tab-separated file (e.g., .tsv file):

The screenshot shows the 'Export' panel in MedTator. The 'Tag & Sentence' option is highlighted with a red box. Below the export options, an 'Export Preview' section displays a tab-separated file (TSV) with the following columns: concept, text, doc_span, sen_span, document, and sentence.

concept	text	doc_span	sen_span	document	sentence
SYMP	dizziness	26,35	26,35	doc_01.txt.xml	The patient also endorses dizziness and some lightheadedness.
SYMP	lightheadedness	45,60	45,60	doc_01.txt.xml	The patient also endorses dizziness and some lightheadedness.
SYMP	headache	81,89	19,27	doc_01.txt.xml	She denies current headache, but states that she gets headaches 3-4x/wk that are associated with photophobia.
SYMP	headaches	116,125	54,63	doc_01.txt.xml	She denies current headache, but states that she gets headaches 3-4x/wk that are associated with photophobia.
SYMP	neck pain	202,211	30,39	doc_01.txt.xml	She denies vision changes and neck pain.
SYMP	seizures	242,250	29,37	doc_01.txt.xml	The patient has a history of seizures, but denies any seizure like activity at this time with no loss of consciousness, no convulsions, and no bladder or bowel incontinence.
SYMP	no loss of consciousness	307,331	94,118	doc_01.txt.xml	The patient has a history of seizures, but denies any seizure like activity at this time with no loss of consciousness, no convulsions, and no bladder or bowel incontinence.
SYMP	no convulsions	333,347	120,134	doc_01.txt.xml	The patient has a history of seizures, but denies any seizure like activity at this time with no loss of consciousness, no convulsions, and no bladder or bowel incontinence.
SYMP	bladder	356,363	143,150	doc_01.txt.xml	The patient has a history of seizures, but denies any seizure like activity at this time with no loss of consciousness, no convulsions, and no bladder or bowel incontinence.
SYMP	bowel incontinence	367,385	154,172	doc_01.txt.xml	The patient has a history of seizures, but denies any seizure like activity at this time with no loss of consciousness, no convulsions, and no bladder or bowel incontinence.
SYMP	Nausea	110,116	110,116	doc_02.txt.xml	During the next 60 hours I had the following reactions, though not all at the same time: 02/19/21: 10:00PM Nausea and extreme headache, 11:00 PM - 2 hours of uncontrollable shivering and headache 2/20/21: 2:AM Extreme headache (like Migraine), Nausea, Body aching for the remainder of the night.
SYMP	Fatigue	383,390	31,38	doc_02.txt.xml	7:00AM - Headache, Body Aches, Fatigue, chills, mild diarrhea and dehydration throughout the day.
SYMP	chills	392,398	40,46	doc_02.txt.xml	7:00AM - Headache, Body Aches, Fatigue, chills, mild diarrhea and dehydration throughout the day.
SYMP	diarrhea	405,413	53,61	doc_02.txt.xml	7:00AM - Headache, Body Aches, Fatigue, chills, mild diarrhea and dehydration throughout the day.
SYMP	dehydration	418,429	66,77	doc_02.txt.xml	7:00AM - Headache, Body Aches, Fatigue, chills, mild diarrhea and dehydration throughout the day.

Supplementary Figure 10 a minimal annotation task – export the annotations

As shown in the above figure, the exported `.tsv` file contains 6 columns, which includes the spans location in the document, spans location in the sentence, and the surrounding sentence of each tag.

2.2 Run your own copy

As MedTator is a serverless application, (i.e., based on pure frontend techniques without server side) there are two ways to run your own copy:

1. Standalone version: MedTator itself is just a single HTML file which contains everything needed. So, you can just open the HTML file directly to use it offline. Moreover, we cached all libraries used in the static folder, so you can use it even without internet access.
2. Online version: You could fork your own copy on GitHub and run it with your own domain name which is provided by GitHub.

2.2.1 Download Standalone Version

You could find the release link on the repo homepage <https://github.com/OHNLP/MedTator> :

File/Folder	Commit Hash	Commit Date	Commits
hehuan2112 update demo animation	e011582	2 days ago	113 commits
.vscode	add cdn/local switch and two libs	last month	
docs	release 1.2.0	2 days ago	
sample	pre-release 1.2.0 test	2 days ago	
templates	pre-release 1.2.0 update style	2 days ago	
.gitignore	update 1.1.3	10 days ago	
LICENSE	update 1.1.3	10 days ago	
README.md	update demo animation	2 days ago	
config.py	pre-release 1.2.0	3 days ago	
requirements.txt	init	4 months ago	
web.py	add samples in the release script and zip	8 days ago	

Supplementary Figure 11 release links on the MedTator repo homepage

Then, you could find the release zip file that only contains the standalone version:

OHNLP / MedTator (Public)

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

Releases / v1.2.0

Release version 1.2.0 Latest

Compare ✎ 🗑

hehuan2112 released this 2 days ago · 1 commit to main since this release 🔗 v1.2.0 🔗 89a8b31

The adjudication function is upgraded to support further editing in the annotation tab. Moreover, the visual and interactive designs are updated in annotation tab and adjudication tab to help you explore the annotated tags easier. A lot of visual styles are updated and some UI issues are fixed. More documents are added to the sample dataset for demo purpose.

Just download the MedTator-1.2.0.zip, unzip it, and open the standalone.html with Chromium-based browser (e.g., Google Chrome, Microsoft Edge, Vivaldi, etc.)

For more details, you could check MedTator README.md and wiki page.

▼ Assets 3

- MedTator-1.2.0.zip 4.48 MB
- Source code (zip)
- Source code (tar.gz)

© 2021 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Supplementary Figure 12 download release zip file

In addition to the release version, you could also download the latest development version by downloading the whole repo:

OHNLP / MedTator (Public)

Code Issues 1 Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 3 tags

Go to file Add file Code

hehuan2112 update demo animation

- .vscode add cdn/local switch and two libs
- docs release 1.2.0
- sample pre-release 1.2.0 test
- templates pre-release 1.2.0 update style
- .gitignore update 1.1.3
- LICENSE update 1.1.3 10 days ago
- README.md update demo animation 2 days ago
- config.py pre-release 1.2.0 3 days ago
- requirements.txt init 4 months ago
- web.py add samples in the release script and zip 8 days ago

Clone ?

HTTPS SSH GitHub CLI

git@github.com:OHNLP/MedTator.git 📄

Use a password-protected SSH key.

Open with GitHub Desktop

Download ZIP

About

A Serverless Web Tool for Corpus Annotation

- Readme
- Apache-2.0 License
- 1 star
- 3 watching
- 3 forks

Releases 3

- Release version 1.2.0 Latest 2 days ago
- + 2 releases

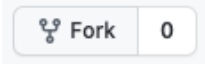
Packages

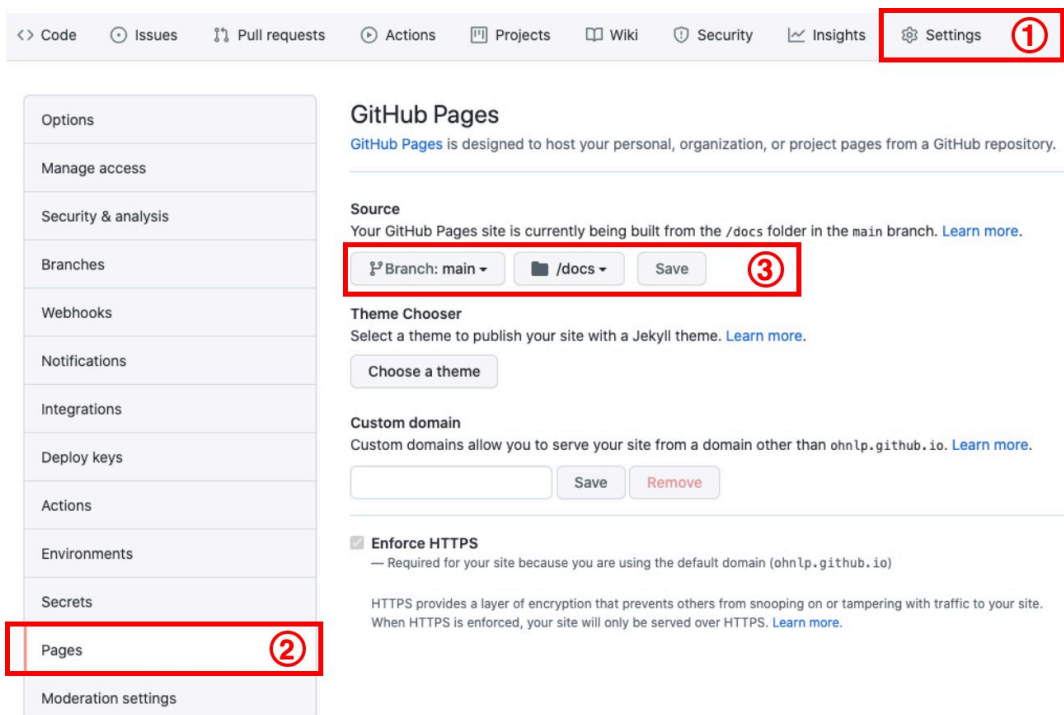
Supplementary Figure 13 download repo as a zip file

Unzip the downloaded zip file, and double click the `docs/standalone.html` to open the latest development version of the standalone MedTator.

2.2.2 Fork Online Version

You can also run MedTator through public GitHub pages services.

- First, go to the homepage of the MedTator repository <https://github.com/OHNLP/MedTator>.
- Secondly, you could find a “Fork” button  in the top right, next to the star button. Click this “Fork” button and follow the instruction to fork MedTator repository to your own GitHub account.
- Thirdly, go to the settings of your forked repo and switch the “Pages” section. Set the source to branch “main” and folder “docs”, then save.



Supplementary Figure 14 GitHub pages configuration

Then, GitHub will assign a customized domain name for this forked MedTator. After a few minutes, you could access your own MedTator copy with that customized domain name. For

example, if your GitHub account name is `username123`, you could find your forked MedTator in <https://username123.github.io/MedTator> by default.

In addition to the above default configurations, you could also specify different branch or folder to server as MedTator homepage according to your own situation. More details about forking a repo on GitHub could be found at <https://docs.github.com/en/get-started/quickstart/fork-a-repo> and more details about the GitHub pages could be found at <https://docs.github.com/articles/configuring-a-publishing-source-for-github-pages/>.

3 Annotation schema file

MedTator supports customized annotation schema for different tasks. Users could define an annotation schema by creating a text file. We adopt the same DTD (Document Type Definitions) file format used by MAE (Stubbs, 2011; Rim, Kyeongmin, 2016) as our schema file definition, which includes the following three parts:

- **Task name:** the name of this schema, which is used as the task identification
- **Concept name:** the concept to be tagged in this task
- **Concept attribute:** the attribute of the concept that describes certain aspects

The schema file is a plain text file with a `.dtd` extension, and it follows the basic specification for DTD declaration. We only implemented the necessary specifications required by defining our annotation task, so it doesn't support full functionality of DTD declarations. The schema file could be created and edited in any text editor or code editor, such as Vim, GNU Emacs, Visual Studio Code, Sublime Text, or any other editor.

Before annotation begins, you need to create a schema file for annotators. To demonstrate how to define an annotation schema file for MedTator, here we present a simple sample schema file for the COVID-19 vaccine adverse event annotation task. In addition, we also present more sample schema files in our repository. You can design you own schema file based on the existing files.

```
<!ENTITY name "COVID_VAX_AE">
```

```

<!-- #PCDATA makes an entity concept -->
<!ELEMENT AE ( #PCDATA ) >
<!ATTLIST AE certainty ( positive | negated | possible ) #IMPLIED "positive" >
<!ATTLIST AE comment CDATA "NA" >

<!ELEMENT SVRT ( #PCDATA ) >
<!ATTLIST SVRT severity ( mild | moderate | severe | NA ) #IMPLIED "NA" >
<!ATTLIST SVRT comment CDATA "NA" >

<!-- No #PCDATA makes a relation concept -->
<!ELEMENT LK_AE_SVRT EMPTY >
<!ATTLIST LK_AE_SVRT arg0 IDREF prefix="link_AE" #IMPLIED>
<!ATTLIST LK_AE_SVRT arg1 IDREF prefix="link_SVRT" #IMPLIED>
<!ATTLIST LK_AE_SVRT comment CDATA "NA" >

```

The details of this schema file are as follows.

3.1 Task name

As shown in the first line of the above sample, the task name "COVID_VAXAE" is defined with the `!ENTITY` tag and the name of the task is in double quotes.

```
<!ENTITY name "COVID_VAX_AE">
```

The task name will be used as the root tag element in the annotation XML file in the following annotation process. Therefore, if the task name is modified in the future, the old annotation files will NOT be opened by the new schema file due to the task name difference.

3.2 Concept name

The concept name is defined with the `!ELEMENT` tag. As shown in our sample schema, we defined three concepts of two types in this annotation task.

We define two entity tags by indicating the `(#PCDATA)`. The first concept name `AE` is for the adverse event:

```
<!ELEMENT AE ( #PCDATA ) >
```

And the second concept `SVRT` is for the severity:

```
<!ELEMENT SVRT ( #PCDATA ) >
```

In addition, we define one relation tag `LK_AE_SVRT` for the relation of adverse event and severity by indicating the `EMPTY` in the schema.

```
<!ELEMENT LK_AE_SVRT EMPTY >
```

The concept name will be used in the annotation file as the XML tag name for annotations. So, the concept name could NOT be repeated in one annotation schema.

3.3 Concept attribute

Concept attribute is used to extend additional information for the annotated tags. You could add as many attributes as you need for the concept defined for a concept. For example, as shown in the sample schema, we defined two attributes (i.e., certainty and comment) for the AE concept:

```
<!ATTLIST AE certainty ( positive | negated | possible ) #IMPLIED "positive" >
<!ATTLIST AE comment CDATA "" >
```

The concept attribute is defined with the `!ATTLIST` tag, followed by the concept name, the attribute name, and the attribute type.

3.3.1 Attribute types

There are four types of attributes: ID, IDREF, CDATA, and value set.

- `ID` type is used for the `id` attribute only. For each concept, there is one, and only one `ID` type attribute. Since MedTator will automatically assign an `id` attribute to a concept, you don't need to specify it. More details about `id` attribute will be discussed in the following section "id attribute".
- `IDREF` type is used for link tag, it indicates an attribute is linked to another entity tag. This type is used in the `argN` attribute only. More details about `argN` attribute will be discussed in the following section "argN attribute".
- `CDATA` type is used for text value, which indicate an attribute is just text content. You could put any text content in this type of attributes.
- Value set type is used to specify a fixed list of values for an attribute. Users could select a value from the pre-defined list instead of input text manually. As shown in our sample schema, the values are defined in parentheses and delimited by `|` symbol. For example, in the certainty attribute for the AE concept, we defined a value set with three values, `(positive | negated | possible)`. And in the severity attribute of the SRVT concept, we defined four values, `(mild | moderate | severe | NA)`.

In addition to the user-defined attributes, MedTator will automatically add the following attributes to a concept when importing schema.

3.3.2 Special attribute

3.3.2.1 `id` attribute

When annotating a document, an id is needed as an identifier for each tag. So MedTator will create an id when annotators create a tag. To make the id easy for users to understand and compatible with MAE, MedTator creates an id by combining the first letter of the concept name and an incremental integer number. For example, when annotating the adverse event concept `AE`, the tags will have the ids `A1`, `A2`, `A3`, etc. When annotating the severity concept `SVRT`, the tags will have the ids `S1`, `S2`, `S3`, etc. When annotating the link entity `LK_AE_SVRT`, the tags will have the ids `L1`, `L2`, `L3`, etc.

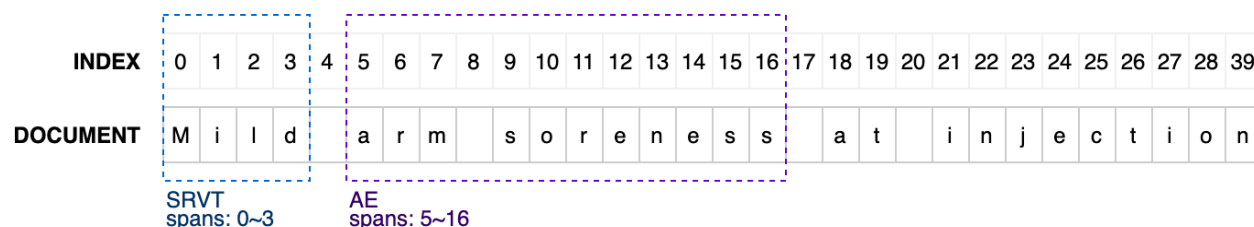
You could also specify prefix to a concept by define the id attribute with prefix field, for example:

```
<!ATTLIST AE id ID prefix="AD" #REQUIRED >
```

We highly recommend you choosing an easy-to-identify concept name, so that its `id` prefix would be easier to read when annotating. For most of time, you don't need to define the id attribute by yourself as shown in our sample, MedTator will automatically process the id prefix if there are multiple concepts with same prefix.

3.3.2.2 `spans` attribute

When annotating a document, a `spans` attribute will be created to indicate the tag's character offset indices, i.e., where the tag starts and where the tag ends in the document. For example, we annotate two tags in a short document "Mild arm soreness at injection":



Supplementary Figure 15 the spans attribute for entity tag

As shown in the above figure, the index number is character-based which means any character, such as alphabet, comma, semi-comma, and quote, will all be counted. The SRVT tag "Mild"

starts from index 0 and ends at 3, so the spans attribute is "0~3". The AE tag "arm soreness" starts from index 5 and ends at 16, so the spans attribute is "5~16".

Like MAE, the `spans` attribute could also be used for document-level annotation. By setting the spans to "#IMPLIED", MedTator will make this concept support "non-consuming" annotation, i.e., the annotated tag is applied to the whole document instead of a text fragment. For example:

```
<!ATTLIST AE spans #IMPLIED >
```

If you add the above line to the schema, the `AE` concept will support "non-consuming" annotation. Then, when annotating a document, the `spans` attribute will be set to `-1~-1` to indicate this is a document-level tag.

3.3.2.3 `argN` attribute

When annotating a link tag, the `argN` attribute will be created to indicate the entity tag related in this link. You could define as many `argN` attributes as needed in one link concept. As shown in the sample schema, we defined two `argN` attributes, namely `arg0` and `arg1`.

```
<!ATTLIST LK_AE_SVRT arg0 IDREF prefix="link_AE" #IMPLIED>  
<!ATTLIST LK_AE_SVRT arg1 IDREF prefix="link_SVRT" #IMPLIED>
```

The `argN` attribute must be an `IDREF` type attribute. It would be clearer to specify the prefix for annotators to understand what kind of tag this attribute should link to. In our sample schema, the prefix of the `arg0` indicates this attribute is for an AE tag, and the `arg1` is for a SVRT tag. Then, while annotating a document, the prefix will be used to generate the fields in the annotation XML file. More details will be discussed in the "Annotation data file" section.

The prefix field is optional, so you could also define these two attributes as follows:

```
<!ATTLIST LK_AE_SVRT arg0 IDREF #IMPLIED>  
<!ATTLIST LK_AE_SVRT arg1 IDREF #IMPLIED>
```

In addition, the `argN` attribute is also optional for a link concept. MedTator will create two `argN` attributes with prefix field `from` and `to` for a link concept without any `argN` attribute.

Although MedTator has this ability, it's recommended that defining the attributes clearly for the convenience of future update and maintenance.

3.3.3 Default attribute value

While defining attributes, you could set default value for an attribute. The default value is placed in quotes at the end of an attribute. For example, in our sample schema, we set the default value "positive" for the `certainty` attribute, and "NA" for the `comment` attribute.

```
<!ATTLIST AE certainty ( positive | negated | possible ) #IMPLIED "positive" >
<!ATTLIST AE comment CDATA "NA" >
```

For the default value for the value set (e.g., the `certainty`), it must be included in the value set. Otherwise, the value couldn't be set correctly. We recommended that set a proper default value for most of attribute to save the annotation time, especially when you are sure about how the values are in your own annotation task.

3.3.4 Mandatory or optional

The attribute value could be mandatory or optional by specifying the `#REQUIRED` or `#IMPLIED` in the attribute definition. If an attribute is set `#REQUIRED`, MedTator will show an asterisk and to indicate it is required.

3.4 Schema samples

To better understand the schema design, we provide the following sample schemas for test. You could also use the schema sample as a start to customize your annotation task.

- `MINIMAL_TASK`: a minimal annotation task for basic function demonstration.
- `COVID_VAX_AE`: a small annotation task for entity and relation annotation.

Those sample schemas and annotated files are available in our MedTator repository in the `sample/` folder.

4 Annotation data file

We adopt the same annotation file format used by MAE (Stubbs, 2011; Rim, Kyeongmin, 2016) to save annotations. The annotations are saved in XML format file, which follows the settings defined in the schema file. The basic structure of the annotation XML file is as follows:

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

```
<TAGS></TAGS>
</TASK_NAME>
```

The annotation XML file has a root element named as the annotation task name. Within that root element, there are two elements, TEXT element and TAGS element. The TEXT element contains the raw text of a text file for annotation, which comes from the .txt file. The TAGS element contains all the tags annotated, with detailed attribute values (e.g., id, spans, text, etc.). The element names in the TAGS elements are defined in the sample schema, i.e., each concept name is used as an element name in the XML file.

For example, using the sample schema file, we annotate a text file `pain_10.txt`, which looks like the following:

A spontaneous report was received from a consumer concerning a 78 years old male patient, who received Moderna's COVID-19 vaccine (mRNA-1273) and experienced terrible pain on the left side of his upper body, it hurt so much, blood clot in his left and right lung and blood clots in right groin.

Then, with this text file and the sample schema, we annotate 3 tags, an entity tag for AE concept, an entity tag for the SRVT concept, and a link tag for LK_AE_SVRT concept:

1 A spontaneous report was received from a consumer concerning a 78 years old male patient, who received Moderna's COVID-19 vaccine (mRNA-1273) and experienced ^{LO} **S0 terrible** **A0 pain** on the left side of his upper body, it hurt so much, blood clot in his left and right lung and blood clots in right groin.

Tag	ID	Spans	Text	Attributes
SVRT	S0	158-166	terrible	severity: NA, comment: <input type="text"/>
AE	A0	167-171	pain	certainty: positive, comment: <input type="text"/>
LK_AE_SVRT	LO			link_AE: A0 AE - pain, link_SVRT: S0 SVRT - terrible, comment: <input type="text"/>

Supplementary Figure 16 Sample annotation

Then, when user saves the annotation, MedTator will create an annotation XML file that would look like the following:

```
<?xml version="1.0" encoding="UTF-8" ?>
<COVID_VAX_AE>
<TEXT><![CDATA[A spontaneous report was received from a consumer concerning a
78 years old male patient, who received Moderna's COVID-19 vaccine (mRNA-
1273) and experienced terrible pain on the left side of his upper body, it
hurt so much, blood clot in his left and right lung and blood clots in right
groin.]]></TEXT>
```

```
<TAGS>
<SVRT spans="158~166" text="terrible" id="S0" severity="NA" comment=""/>
<AE spans="167~171" text="pain" id="A0" certainty="positive" comment=""/>
<LK_AE_SVRT id="L0" link_AEID="A0" link_AEText="pain" link_SVRTID="S0"
link_SVRTText="terrible" comment=""/>
</TAGS>
</COVID_VAX_AE>
```

As shown in this sample, the content in the .txt file are saved in the `TEXT` element. The three tags we annotated are saved as three elements, `<AE>`, `<SRVT>`, and `<LK_AE_SVRT>`. The attributes of each concept are saved in the element field such as id, spans, text, and severity.

5 Tool usage

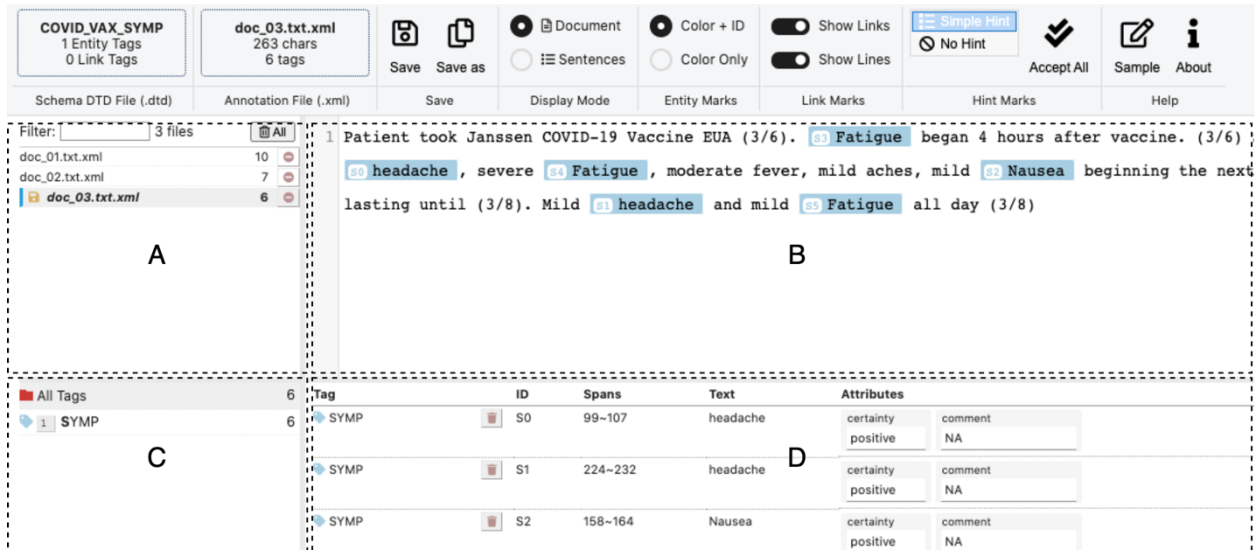
At present, MedTator has four tabs to cover the core annotation steps, including document annotation, corpus statistics, annotation export and IAA calculation. Each tab provides functions related to one core phase in the annotation workflow. The details of each tab are as follows.

5.1 Annotation tab

This tab allows the user to annotate texts according to pre-defined schema by coordinated four views, including:

1. The file list view (Supplementary Figure 17 (A)) shows the summary of files and the annotation status of each file. It supports filtering files by the file name.
2. The tagging view (Supplementary Figure 17 (B)) shows the content of the selected file and the visualized entity tags, relation tags, and annotation hints in selected file.
3. The concept list view (Supplementary Figure 17 (C)) shows all entity and relation concepts in the schema and the count of each concept annotated in the selected file. By clicking on each tag name, you can filter the tag list to show the selected tag.
4. The tag list view (Supplementary Figure 17 (D)) shows the detailed information of the annotated tags, such as spans, text, and attributes. The attributes are defined in the schema file and will be displayed as drop-down list or input box. For relation tags, the drop-down list of entity tags will update automatically when entity tags are changed.

The selected linked entity tag will be display in the attributes, and you can change the linked tag in the drop-down list.

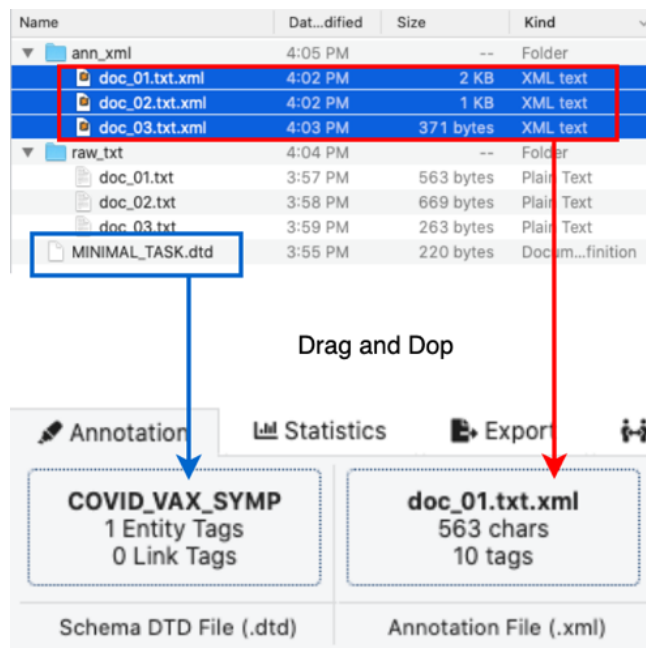


Supplementary Figure 17 the annotation tab that contains 4 views, (A) the file list, (B) the tagging view, (C) the concept list, and (D) the tag list. (The actual interface you see may be different from what is shown in the above screenshot due to tool update).

5.1.1 Schema and annotation / text files import

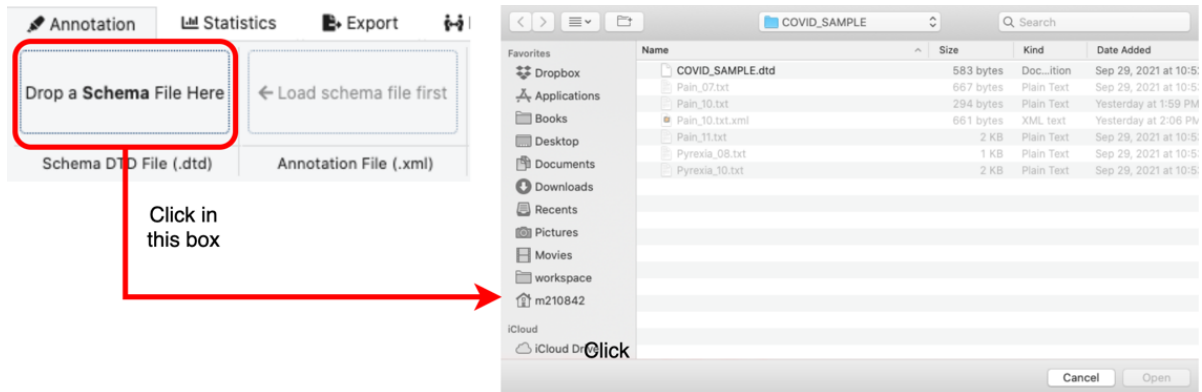
MedTator supports two ways to import schema file and annotation files.

The first one: drag and drop files from file explorer (e.g., Finder on MacOS) to the box.



Supplementary Figure 18 import schema file by dragging and dropping file

And the second one: click on the schema box to open the file select dialog and upload file.

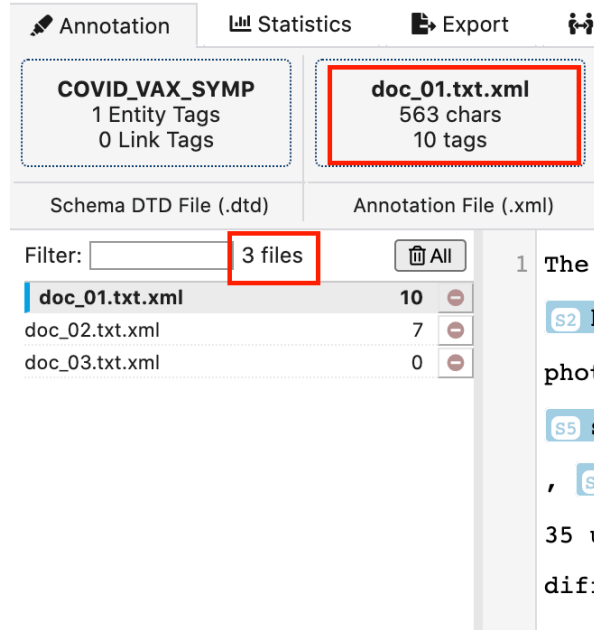


Supplementary Figure 19 import schema file by clicking the schema box

When the schema file is imported, the concept list view will show the concept names. And when the annotation files are imported, the file list view will show the file names and the total number of files.

5.1.2 Annotation file selection

MedTator support multi-document annotation and the imported files are listed in the file list view. The number of imported files is displayed at the top the file list. To help users find the file to annotated easier, there is a filter box displayed at the top the file list, which support file name matching.

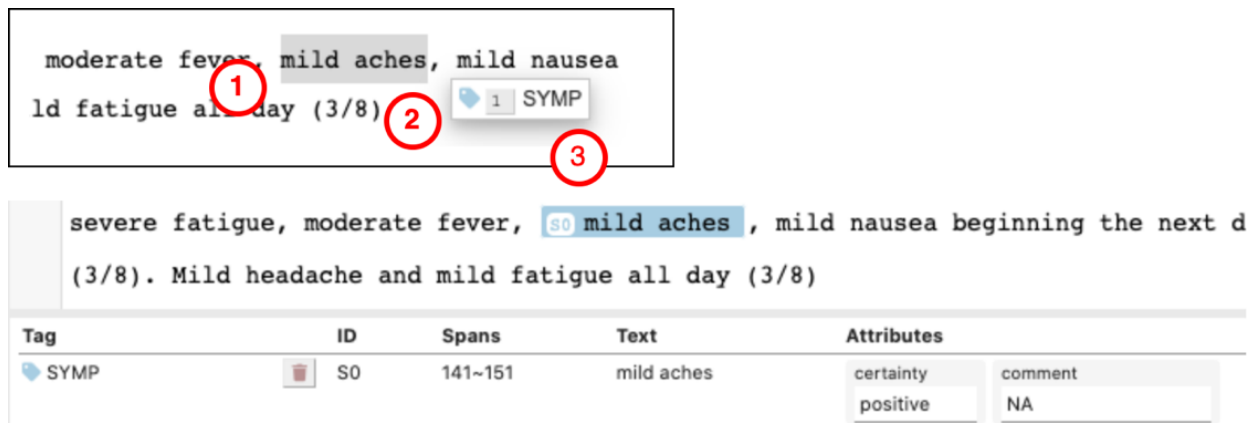


Supplementary Figure 20 the current working file and total number of files

5.1.3 Entity annotation

The entity tag can be annotated through the tagging view by three steps:

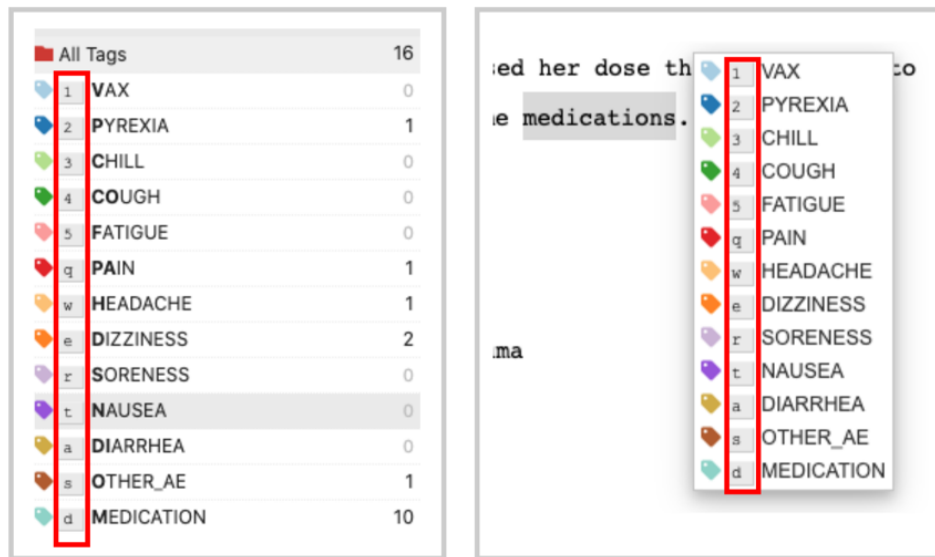
1. Highlighting the text to be tagged.
2. Right click in the tagging view (or tap with two fingers on trackpad in MacOS).
3. Click the entity name in the popup menu.



Supplementary Figure 21 annotate tag by highlighting and clicking the popup menu

In addition to the entity annotation by clicking, MedTator also supports shortcut keys for quick annotation.

In the concept list view and the popup menu, there is a number or a letter on the left of each concept name, which is the shortcut key for that concept. For example, as shown in the following figure, the number key 1 is assigned to the VAX concept, 2 to the PYREXIA, 3 to CHILL, 4 to COUGH, etc.



Supplementary Figure 22 Annotation shortcut keys

With the shortcut keys, the entity annotation could be done in just two steps:

1. Highlighting the text to be tagged.
2. Press the corresponding shortcut key.

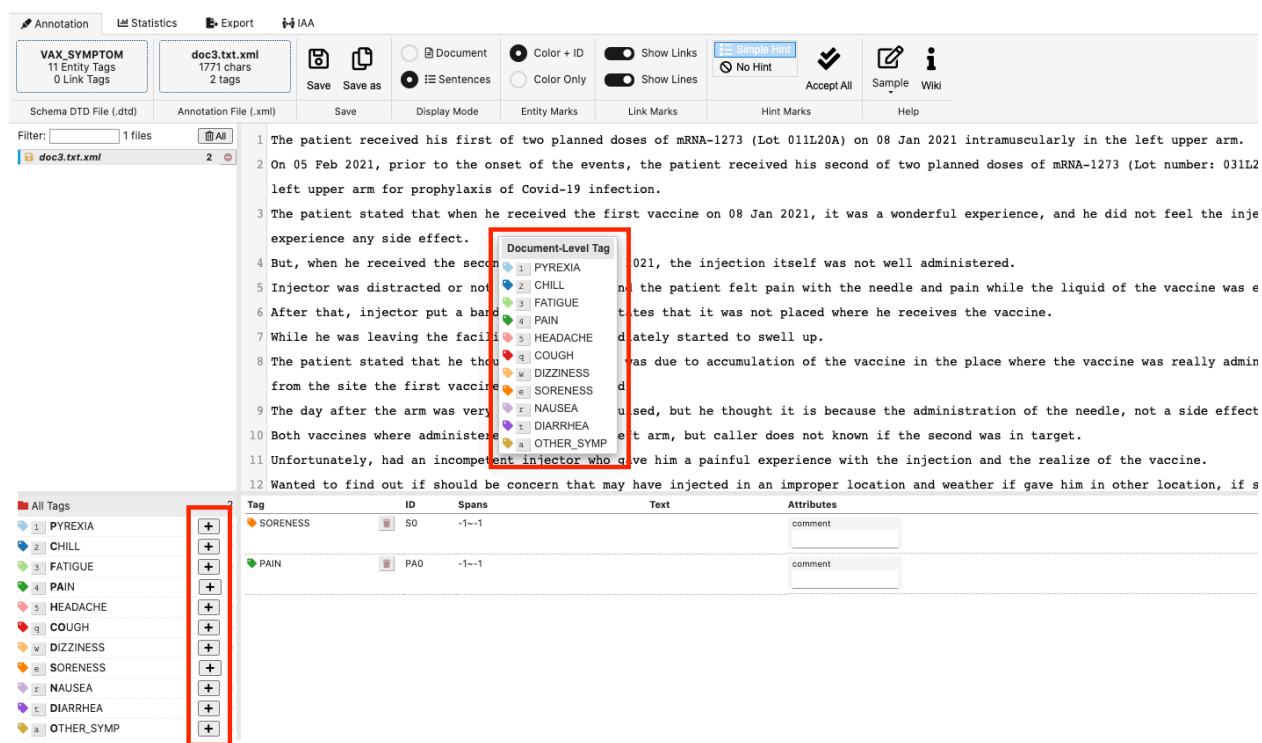
For example, when annotating a headache concept, you could first highlight the token “headache” in the tagging view, then press the shortcut key **w**.

5.1.4 Document-level annotation

MedTator supports document-level annotation with customized schema file for document-level annotation task. As introduced in the schema file design, by setting the spans attribute for an entity concept, the said entity concept can be used for document-level annotation.

To add a document-level annotation, the process is similar to the entity annotation, but just takes two steps:

1. Right click in the tagging view (or tap with two fingers on trackpad in MacOS).
2. Click the concept name in the popup menu.



Supplementary Figure 23 document-level annotation

Or, as shown in the above figure, you could also click the “+” button in the concept list to add a document-level tag.

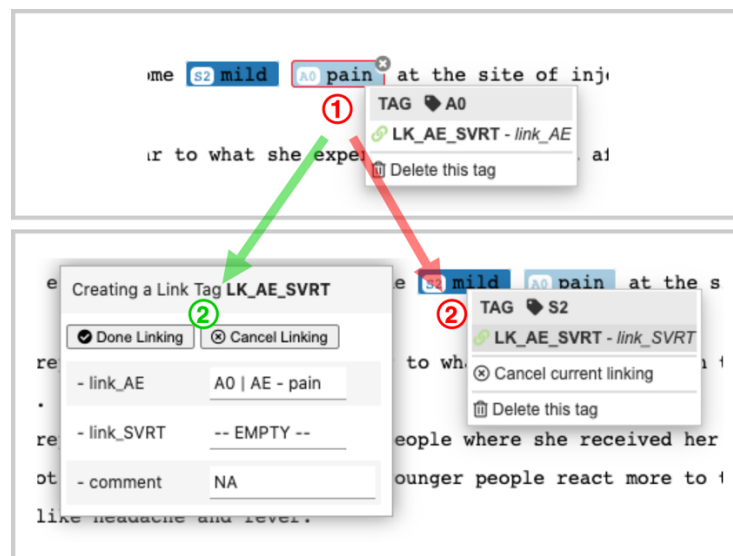
5.1.5 Relation annotation

MedTator provides two methods to annotate a relation tag.

5.1.5.1 Add relation tag in tagging view

The relation concept could be added by the following steps:

1. Click on the annotated tag, a popup menu will be displayed which contains available relation concepts. You could select the one which is needed.
2. A floating panel would be displayed based on the relation concept decided by the previous step, you could (1) click on the tag to be added and select the attribute from the popup menu. Or (2) use this floating panel to change other attributes and finish relation annotation.



Supplementary Figure 24 add relation tag in tagging view in two ways

For example, as shown in the above figure, we have added two entity tags, i.e., a severity tag “mild” and an AE tag “pain”. First, you could click on the “pain” tag, a popup menu will be displayed, and you could click the “LK_AE_SVRT – link_AE” option in this menu to add “pain” tag as an attribute in a new LK_AE_SVRT relation tag. Secondly, you could click on the “mild” tag and select “LK_AE_SVRT – link_SVRT” attribute to finish the relation annotation.

Or you will find that a floating panel is display with all the attributes in the `LK_AE_SVRT` tag. You could select the `link_SVRT` attribute from the dropdown menu and click the “Done Linking” button to add a new relation tag.

5.1.5.2 Add relation tag in concept list

In addition to the previous method, the relation concept could also be added by two steps:

1. Click the “+” button in the concept list.
2. Modify the entity link the tag list.

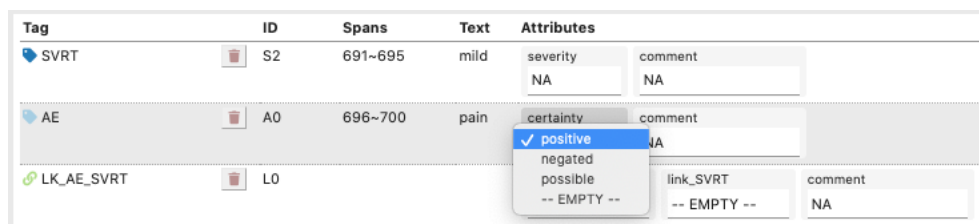


Supplementary Figure 25 add relation tag in concept list

For example, as shown in the above figure, two entity tags have (i.e., an `AE` concept and a `SVRT` concept) have been annotated. To add a new relation tag, first click the “+” button of the `LK_AE_SVRT` concept in the concept list. Then, a new empty `LK_AE_SVRT` tag will be created and displayed in the tag list, you could modify the attributes and select the existing entities in the dropdown selections to complete the details.

5.1.6 Attribute modification

The attributes of each tag can be modified in the tag list.



Supplementary Figure 26 attribute value modification

For each type of attribute defined in the schema file, MedTator provides the following method to modify values:

- The ID type attribute which is used in the relation tags is displayed as a dropdown box, whose values come from the annotated entity tags. The entity tag id, concept name, and extracted text will be displayed as the option for reference.
- The CDATA type attribute is displayed as an input box, in which you could modify the text as needed.
- The value set type attribute is displayed as a dropdown box, in which you could select pre-defined values.

5.1.7 Hint Marks

MedTator could show annotation hints based on the annotated tags.

6 The consumer reported receiving her first injection of the Pfizer BioNTech COVID-19 vaccine 2 weeks ago and the only side effect she experienced was some **S2 mild** **A0 pain** at the site of injection that lasted only a day.

7 The consumer reports the **A pain** was similar to what she experienced in the past after a flu shot and a shingles shot.

8 The consumer reported **what** she was told by people where she received her injection that people react more to the second shot compared to the first, and younger people react more to the injections that older people for side effects like headache and fever.

9 The consumer also stated, "Here is the question, my mother who is going to be 97 and I, who is going to be 70 received the first Pfizer Shot a couple (Covid-19 Vaccine) of weeks ago, we are due for the second shot for next Thursday actually and I was told we had no side effects at all thank goodness just the sore arm.

10 Now I was told for the second shot that you usually do get some kind of side effects, so my question is my mother and I taking a shot at the same time, I live along with her and should I have somebody with me just what are the odds we are going to have side effects from the second shot, I guess that's my question." The outcome of the event **S mild** **A pain** at the site of injection was recovered while other event was unknown.

11 Information on the Lot/Batch Number has been requested.

Supplementary Figure 27 annotation hints

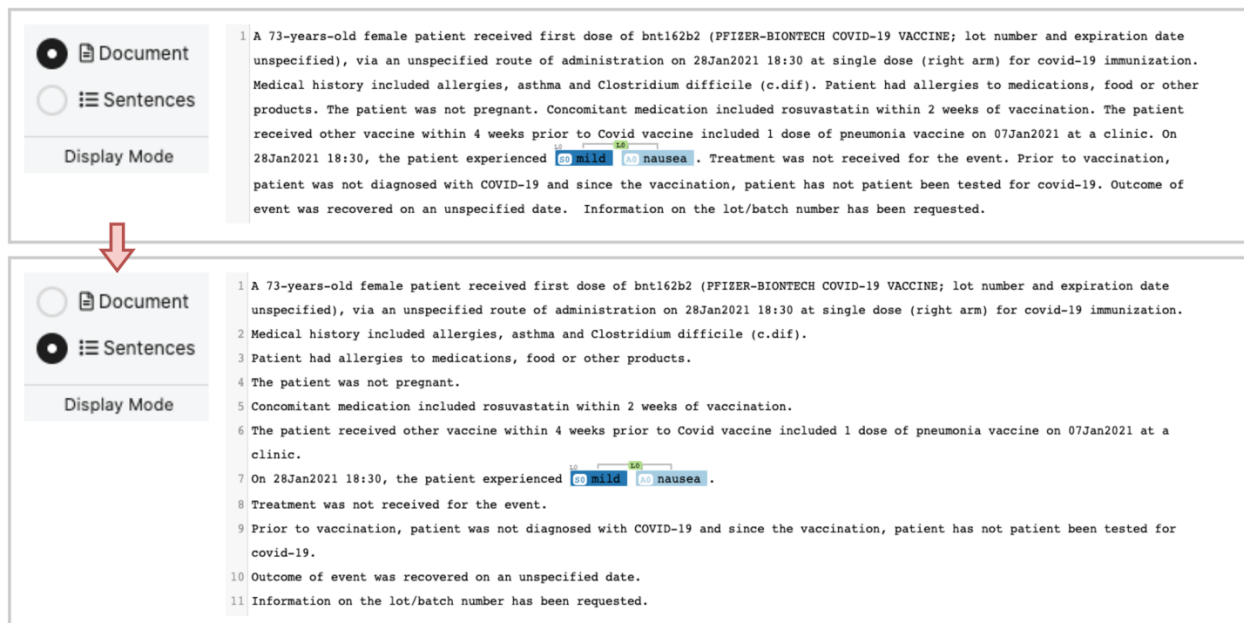
For example, when the “pain” is annotated as an **AE** tag. All the other “pain” appears in all documents which haven’t be annotated will be wrapped in a dotted box with a concept prefix “A”. And it’s the same for the annotated “mild” tag.

5.1.8 Document display mode

MedTator supports two different display mode for showing the document in the tagging view. As shown in the following figure, you could select different display in the menu:

1. Document mode: In this mode, the content in the selected file will be displayed in its original format.
2. Sentence mode: MedTator splits the document into sentence by an open-source JavaScript library “compromise” and creates a mapping of the offsets of each sentence in the original document. The blank lines will be removed in this mode.

While in the sentence mode, the annotations are still saved with their original spans. MedTator will calculate the offsets automatically when rendering the annotated tags and exporting the annotations.

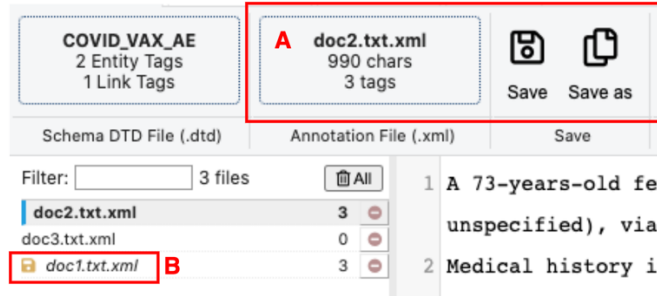


Supplementary Figure 28 document display mode

The sentence detection algorithm could be changed in the setting panel. The default algorithm is a character-based detection algorithm which has better performance than other options.

5.1.9 Save annotations

By using the HTML5 techniques, MedTator supports saving file to local disk with the File System Access API (https://developer.mozilla.org/en-US/docs/Web/API/File_System_Access_API).



Supplementary Figure 29 two ways of saving annotation

As shown in the above figure, MedTator provides two ways to save annotation file:

- A: Save the current working file. By clicking the “Save” button, the current working file, which is the “doc2.txt.xml” will be saved. Moreover, by clicking the “Save as” button, MedTator will ask the user to save current file to a new copy instead of saving to current working file.
- B: Save a specific file. By clicking the yellow disk icon that is on the left of a file name, the corresponding file will be saved. In the above figure, when clicking the yellow disk icon, the “doc1.txt.xml” will be saved. The current working file “doc2.txt.xml” will **NOT** be saved, because the clicked yellow disk button is linked to the “doc1.txt.xml”.

5.2 Statistics tab

MedTator provides real-time statistics on the annotated tags. Whenever a new annotation is added or existing annotation is modified, the statistics can be updated in this tab.

5.2.1 Basic summary

MedTator provides a basic summary on the annotations in the Annotation tab. For example, if we import the COVID_VAX_AE sample, the Annotation may look like the following:

Summary	Value
# of documents:	3
# of tags in schema:	2
# of annotations:	6
# of annotations per tag:	1.50
# of annotations per document:	2.00
# of sentences:	34
# of sentences per document:	11.33

Supplementary Figure 30 annotation tab with the COVID_VAX_AE sample data imported

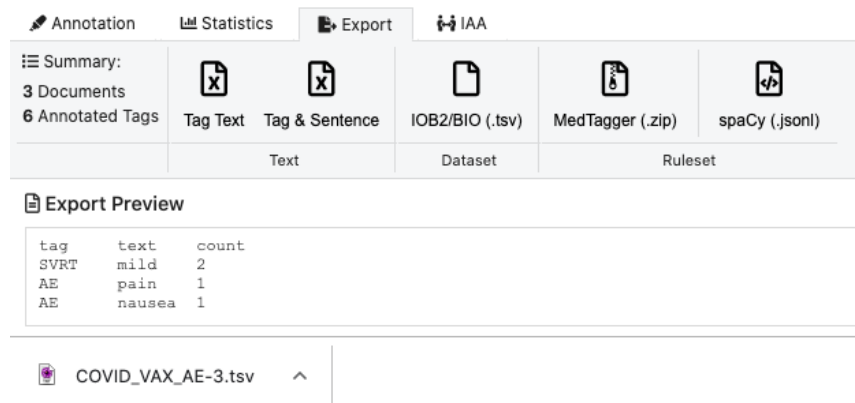
The left panel will show a basic summary, such as number of document and tags.

5.2.2 Annotated tag statistics

In addition to the basic summary, the statistics tab will also show the detailed annotation tags in each entity concept with the source file location. For example, as shown in the figure, the “mild” SVRT tag is annotated twice in two files. One is in the doc1.txt.xml, the other one is in the doc2.txt.xml file.

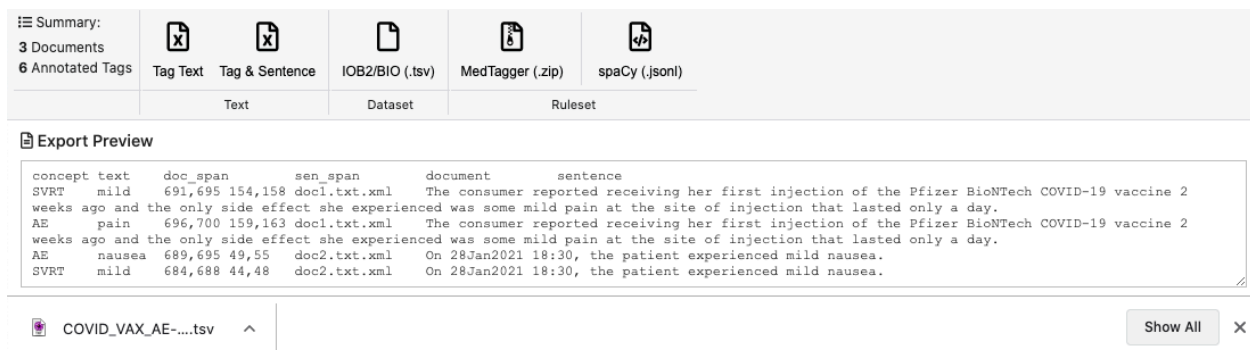
5.3 Export tab

MedTator could export the annotations to different format for downstream tasks. The detailed format may be changed in future. For example, by clicking the “Tag Text” button, MedTator will create a .tsv file which contains the concept name, annotated text, and the count:



Supplementary Figure 31 export as tag and text

By clicking the “Tag & Sentence” button, MedTator will create a .tsv file which contains more columns on the context information of each annotated tag.



Supplementary Figure 32 export as tag with context sentence

In addition to the text, MedTator could also export the annotations to .tsv file in IOB2/BIO format for name-entity recognition, to MedTagger ruleset package (Liu *et al.*, 2012), or jsonl format for spaCy ruleset. More formats will be added in future to support other downstream tasks.

5.3.1 How to use the exported datasets?

The exported IOB2/BIO format files can be used for name-entity recognition training / evaluation task. For example, you can fine-tune a BERT-based model with the exported IOB2 / BIO format files. For more technical details, see HuggingFace document (https://huggingface.co/docs/transformers/v4.13.0/en/custom_datasets).

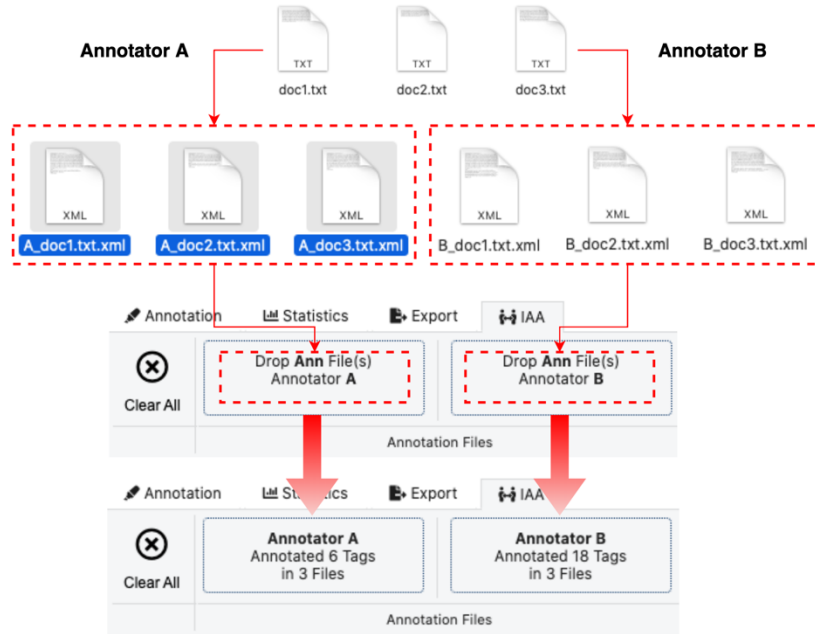
The exported MedTagger ruleset package could be used by MedTagger IE rule engine (<https://github.com/OHNLP/MedTagger>) .

The exported jsonl format pattern file can be used by spaCy NLP rule-based entity recognition module. It contains entity patterns defined by spaCy (<https://spacy.io/usage/rule-based-matching#entityruler>) and can be read to load patterns for named entity and text classification labelling.

5.4 Adjudication tab

MedTator supports IAA calculation and adjudication of two annotators in this tab.

Before start IAA calculation and adjudication, you need to load the schema file in the MedTator. Then, the documents need to be annotated by two annotators.



Supplementary Figure 33 annotation files from two annotators

For example, in the COVID_VAX_AE sample dataset, we have three documents, namely doc1.txt, doc2.txt, and doc3.txt. Two annotators annotated separately and finally got two annotations on each document, A_doc1.txt.xml, A_doc2.txt.xml, and A_doc3.txt.xml are from annotator A, while B_doc1.txt.xml, B_doc2.txt.xml, and B_doc3.txt.xml are from annotator B. Then you could drag and drop the A_doc1.txt.xml, A_doc2.txt.xml, and A_doc3.txt.xml to the annotator A box, B_doc1.txt.xml, B_doc2.txt.xml, and B_doc3.txt.xml to the annotator B box. MedTator will read those files and show the number of tags and files in each box.

5.4.1 IAA calculation

F1-score is used to assess the IAA. MedTator uses a span-based method to get the F1-score. Given the majority of the annotation tasks may have imbalanced sample distribution, F1-score can provide reasonable measure. F1-score is a well-established metric in the information retrieval community, which measures the combination of positive predictive value (precision) and sensitivity (recall) of the test object (Hripcsak and Rothschild, 2005).

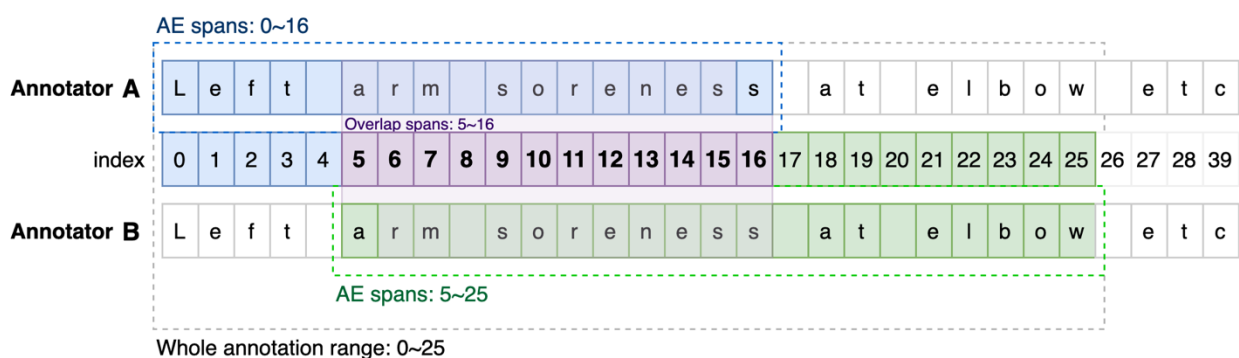
$$F_1 = 2 \times \frac{Precision \times Recall}{(Precision + Recall)} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

To calculate the IAA, we could use annotator A's annotation as ground truth (correct) and annotator B's as prediction. Then, the TP (true positive) is the number of those tags annotated by both annotator A and B. The FP (false positive) is the number of those tags annotated by annotator B but not annotated by A. The FN (false negative) is the number of those tags annotated by annotator A but not annotated by B. By counting the TP, FP, and FN in each file, we could get the F1-score of two annotators. Moreover, by setting the conditions such as file range and concept, F1-scores of different levels could be obtained (e.g., overall F1-score, a specific concept's F1-score, and single document F1-score).

To count the number of TP, FP, and FN, we need to set a threshold, overlap ratio, in the algorithm to determine whether two annotated tags are agreed by two annotators or not. In the actual corpus annotation, it's very common that the annotations by different annotators do not exactly match with each other. So that it's necessary to accept the difference in the annotation to some extent.

MedTator supports two modes for detecting the agreement. The first mode is exact match, which requires the two annotated tags are exactly same (i.e., exactly same offset in the document and length). The second mode is part match, which uses an overlap ratio to determine the agreement. The detailed calculation method of the second mode is as follows.

After import the annotations from two annotators, you could specify the overlap ratio which is the threshold to measure whether both annotators have an agreement on same text.

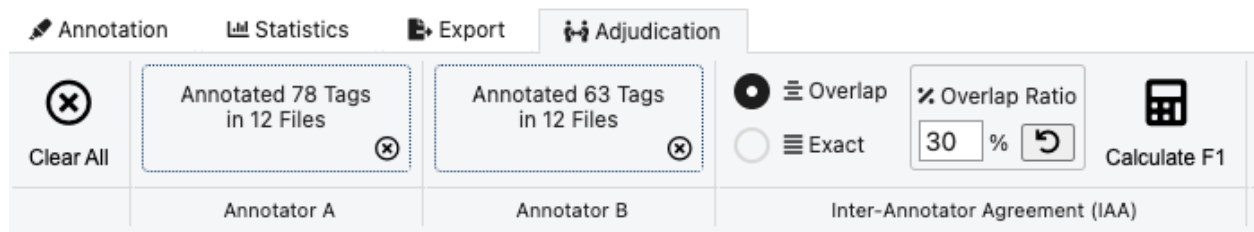


Supplementary Figure 34 the overlap ratio for IAA calculation

By default, the overlap ratio is 50%, which means both annotators would have an agreement on an annotated text in the same concept if both annotated it and the overlap of the annotated

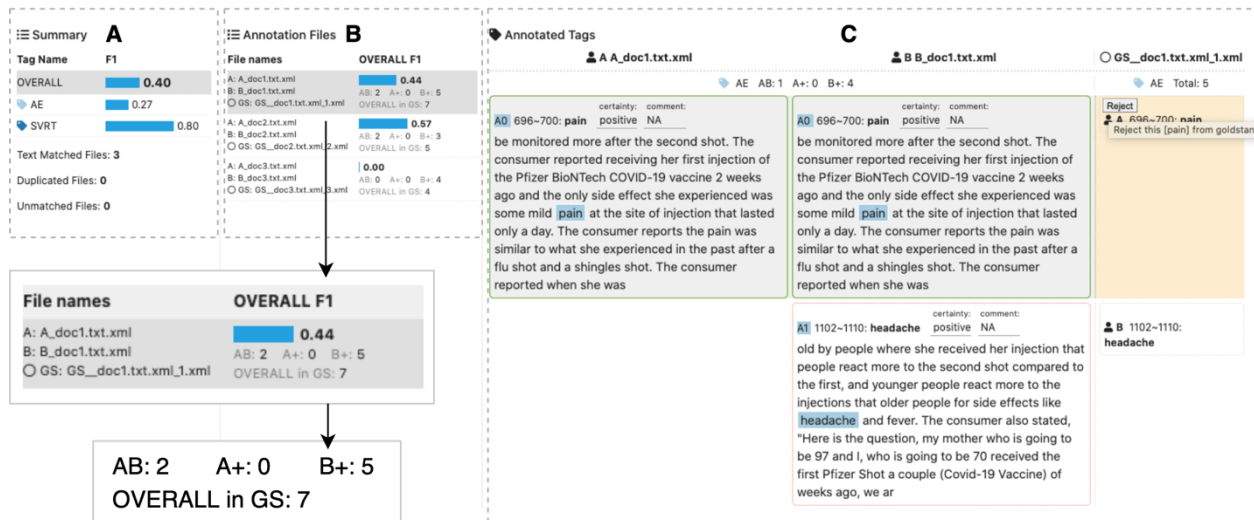
tag is equal or greater than 50%. For example, as shown in the above figure, annotator A annotated an AE concept “left arm soreness”, while annotator B annotate the “arm soreness at elbow”. These two annotated tags are not exactly match each other, so it is needed to calculate how much the overlap is to decide whether two annotators have an agreement on the annotation. As we can see in the figure, the spans of the overlapped part is 5~16, which is 12 characters, and the whole annotation covers spans 0~25, which is 26 characters. So the overlap ratio of these two tags is $12 / 26 = 46.15\%$. As it is smaller than the defined threshold 50%, there is no agreement on this annotation. The results of all tags from both annotators will be used in the calculation of the file-level, concept-level, and overall IAA score.

After setting the overlap ratio, you could click the “Calculate F1” button to calculate the F-score from different levels.



Supplementary Figure 35 calculate the IAA F-score

Then MedTator will use the given overlap ratio to calculate the F-score, the result would look like the following according to the annotations:



Supplementary Figure 36 IAA calculation result

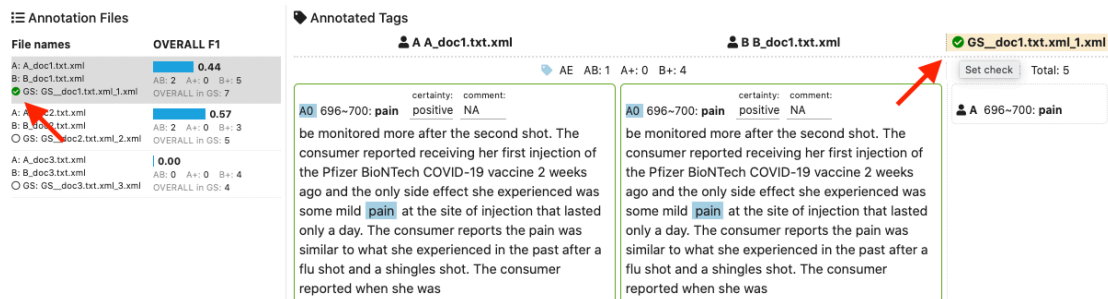
The IAA result contains three panels, (A) summary showing the overall F1 and the concept level F1, (B) the file level result showing the detailed results of the selected concept level grouped by file, and (C) the document level result showing the detailed tags. All panels linked with each other, when clicking on the concept or the file item, other views will be updated accordingly. As show in the above figure, when selected the “OVERALL” F1 in the summary, the files will show the results of all tags. For the doc1.txt.xml, annotator A and B achieves a F1 result of 0.44. the label “AB: 2” indicates that both annotator A and B agree on the 2 annotations, “A+: 0” indicates that there is no annotation that only agreed by annotator A, and “B+: 5” indicates that there are 5 tags that are only annotated by annotator B.

Then, the document level panel shows the detailed tags, order by the concept. In this panel, the results are displayed in three columns, the first column is the annotations from annotator A, the second column is the from annotator B, and the last column is for adjudication. In each column, the tags are displayed in dotted boxes with the attributes and context text. If a tag is agreed by both annotators, it will be displayed as a green dotted box in both first and second column. If a tag is only annotated by one annotator, it will be displayed as a red dotted box in one column.

5.4.2 Download adjudication copy

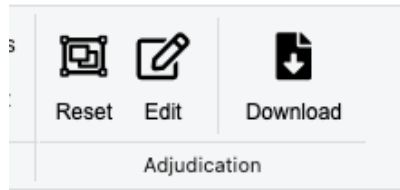
The adjudication column in the document level will generate a default gold standard based on the annotations from both annotator A and B. You could accept or reject a tag by clicking the “Accept” or “Reject” button displayed on the top left of a tag box. When the adjudication on one document is finished, you could set a green checked mark on this document.

This check mark is just for a visual reminder, and it won’t affect the annotations.



Supplementary Figure 37 set check mark in adjudication

You could download the adjudication results of all documents in a zip file by clicking the “Download” button in the menu:



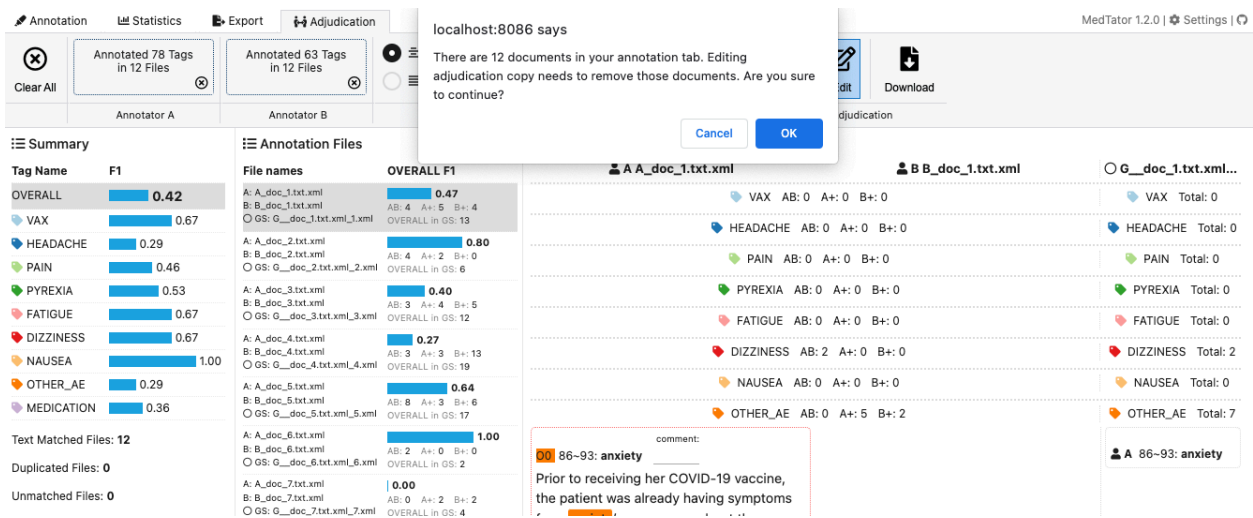
Supplementary Figure 38 download the adjudication result

Then you could use the exported file as gold standard for downstream tasks.

5.4.3 Edit adjudication copy

You could also further edit the adjudication copy by clicking the “Edit” button. MedTator will send current adjudication copy to the annotation tab for further editing.

Before sending, it will ask for confirm if there are documents annotating:



Supplementary Figure 39 send the adjudication copy

Once you confirm, the adjudication will be sent to annotation tab:

Annotation Statistics Export Adjudication MedTator 1.2.0 | Settings |

VAX_AE_MED 9 Entity Tags 1 Link Tags G_doc_1.txt.xml_1.x... 2327 chars 13 tags

Save Save as Document Sentences Search Clear Color + ID Show Links Show Lines Simple Hint No Hint Accept All Sample Wiki

Schema File (.dtd) Annotation File (.xml/.txt) Save Display Mode Search Entity Marks Link Marks Hint Marks Help

Filter: 12 files All

G_doc_1.txt.xml_1.xml 13
 G_doc_2.txt.xml_2.xml 6
 G_doc_3.txt.xml_3.xml 12
 G_doc_4.txt.xml_4.xml 19
 G_doc_5.txt.xml_5.xml 17
 G_doc_6.txt.xml_6.xml 2
 G_doc_7.txt.xml_7.xml 4
 G_doc_8.txt.xml_8.xml 27
 G_doc_9.txt.xml_9.xml 2
 G_doc_10.txt.xml_10.xml 4
 G_doc_11.txt.xml_11.xml 2
 G_doc_12.txt.xml_12.xml 1

administrator came to me (pharmacist on duty) to notify me that she was **not feeling well**. I went to go check on her and she stated that she felt as though she was going to **faint**. I advised her to stay seated, asked if she would like an **ice pack** and water, and stepped out to get those for her while remaining vigilant (because I wanted to be in the same room as her for observation/safety). I brought her the **ice pack**, she held it on her chest and head and said it felt nice. After about a minute I stepped out again to ask a technician to get a water for her. When I came back, the patient was still seated but had a blank stare and her mouth was slightly open. I repeated her name out loud and nudged her shoulder several times, however the patient was **unresponsive completely**. I immediately looked for the red emergency kit in the consultation room but did not see one (I later found out that there was an **EpiPen** on the table in the room, but I did not see it at the time due to the urgency of the situation). I rushed back into the pharmacy, told the technicians to call 911, and took an **EpiPen** off the shelf in the pharmacy. I ran back into the room and the patient's **face was very red**, her **eyes were bulging slightly**, and it looked like she was trying to reach her hand up or move it, but was unable to. I injected her with the **EpiPen** immediately, and within roughly 5 seconds she came back to consciousness and looked at me and said "I'm sorry". EMS arrived extremely fast and was there a few minutes after using the **EpiPen**. She was then taken by ambulance to the hospital. Patient had a friend, with her at the store who was scheduled to receive the vaccine as well, but after the incident she decided otherwise. Later that evening I gave patient's friend a phone call to check and see how patient was doing, and to also let her know that patient's vaccination card was at the pharmacy if she wanted to pick it up. That evening around 6:30-7pm, patient's husband came to the pharmacy to get her card, he updated us that she was being

All Tags	Tag	ID	Spans	Text	Attributes
13					
1 VAX					
0 HEADACHE					
0 PAIN					
0 PYREXIA					
0 FATIGUE					
2 DIZZINESS					
0 NAUSEA					
7 OTHER_AE	OTHER_AE	O3	989-1012	unresponsive completely	comment
4 MEDICATION					
0 TREATMENT					
	OTHER_AE	O4	406-411	faint	comment
	OTHER_AE	O5	1407-1424	face was very red	comment
	OTHER_AE	O6	1430-1456	eyes were bulging slightly	comment
	MEDICATION	M0	1568-1574	EpiPen	comment
	MEDICATION	M1	470-478	ice pack	comment

Supplementary Figure 40 edit the adjudication copy

MedTator will switch to the annotation tab and show the adjudication copy. The annotator label will be displayed in the tag list to show the initial adjudication result:

- Green **AB**: represents the annotation is agreed by both annotators.
- Orange **A**: represents the annotation is annotated by annotator A, but not agreed by B.
- Blue **B**: represents the annotation is annotated by annotator B, but not agreed by A.

In the tagging view with “Color Only” entity marks selected, the annotator label will be added at each entity mark beginning. You can add, modify, and delete any tags in this view.

6 References

- Chen,W.-T. and Styler,W. (2013) Anafora: A Web-based General Purpose Annotation Tool. *Proc. Conf. Assoc. Comput. Linguist. North Am. Chapter Meet.*, **2013**, 14–19.
- Eckart de Castilho,R. *et al.* (2016) A Web-based Tool for the Integrated Annotation of Semantic and Syntactic Structures. In, *Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH)*. The COLING 2016 Organizing Committee, Osaka, Japan, pp. 76–84.
- Gompel,M. van and Reynaert,M. (2013) FoLiA: A practical XML format for linguistic annotation – a descriptive and comparative study. *Comput. Linguist. Neth. J.*, **3**, 63–81.
- Hripcsak,G. and Rothschild,A.S. (2005) Agreement, the F-Measure, and Reliability in Information Retrieval. *J. Am. Med. Inform. Assoc. JAMIA*, **12**, 296–298.
- Islamaj,R. *et al.* (2020) TeamTat: a collaborative text annotation tool. *Nucleic Acids Res.*, **48**, W5–W11.
- Klie,J.-C. *et al.* (2018) The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In, *Proceedings of the 27th COLING : System Demonstrations*. Santa Fe, New Mexico, pp. 5–9.
- Kwon,D. *et al.* (2013) BioQRator: a web-based interactive biomedical literature curating system. In, *Proceedings of the BioCreative IV Workshop*. Washington, DC, USA, pp. 241–246.
- Liu,H. *et al.* (2012) Towards a semantic lexicon for clinical natural language processing. *AMIA. Annu. Symp. Proc.*, **2012**, 568–576.
- Neves,M. and Ševa,J. (2021) An extensive review of tools for manual annotation of documents. *Brief. Bioinform.*, **22**, 146–163.
- Rim, Kyeongmin (2016) MAE2: Portable Annotation Tool for General Natural Language Use. In, *Proceedings of the 12th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*. Portorož, Slovenia.
- South,B. *et al.* (2012) A Prototype Tool Set to Support Machine-Assisted Annotation. In, *Proceedings of the 2012 BioNLP Workshop*. ACL, Montréal, Canada, pp. 130–139.
- Soysal,E. *et al.* (2018) CLAMP – a toolkit for efficiently building customized clinical natural language processing pipelines. *J. Am. Med. Inform. Assoc.*, **25**, 331–336.
- Stenetorp,P. *et al.* (2012) brat: a Web-based Tool for NLP-Assisted Text Annotation. In, *Proceedings of the EACL 2012: Demonstrations*. ACL, Avignon, France, pp. 102–107.
- Stubbs,A. (2011) MAE and MAI: Lightweight Annotation and Adjudication Tools. In, *Proceedings of the 5th Linguistic Annotation Workshop*. Association for Computational Linguistics, Portland, Oregon, USA, pp. 129–133.
- Wei,C.-H. *et al.* (2013) PubTator: a web-based text mining tool for assisting biocuration. *Nucleic Acids Res.*, **41**, W518–W522.
- Wei,C.-H. *et al.* (2019) PubTator central: automated concept annotation for biomedical full text articles. *Nucleic Acids Res.*, **47**, W587–W593.