



Towards User-centered Corpus Development: Lessons Learnt from Designing and Developing MedTator

Time Flies Like a Banana: Advancing Natural Language Processing Techniques

S107

Huan He, PhD

Mayo Clinic

Twitter: #AMIA2022



Disclosure



I and my spouse/partner have no relevant relationships with commercial interests to disclose.

Learning Objectives



After participating in this session the learner should be better able to:

- Learn the challenges of developing a text annotation tool
- Learn the design and our experience in tool development

Outline



1. Background and challenges
2. Architecture design
3. Lessons we learned

Background - NLP in healthcare



NLP has been widely applied in healthcare-related domains

Uncovering **Hidden Symptoms** Identify **Lung Cancer Screening Patients**

Extracting Biomedical Factual **Knowledge** Identify Reasons for **Statin Non-Adherence**

Examiner Note **Extraction** Sampling **Adverse Drug Events** Identifying the **Risk for Hospitalizations**

Identify **Social Needs** Unraveling the “Other” Diagnosis of **Suspected Suicide Attempts**

Analyze ADRD Caregivers’ Mental Health Needs Clinical **Relation Extraction** Confirm MGUS **Diagnoses**

Build Disease-specific **Symptom Vocabulary** Identifying **Transgender** and Gender **Diverse**

Clinical Trial Eligibility **Criteria Representation** Predict Functional **Pain Score** Temporal **Reasoning**

Predict **Oral Cancer Risk** Identifying **Menopausal Status** Gleaning **Prediabetic** Risk and Factors

Extracting **Cancer Phenotypes** Analyzing **COVID-19 Vaccination** Sentiment Dynamics

NLP in AMIA 2022 Annual Symposium

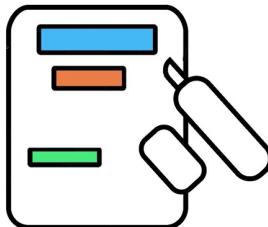
Background - Gold Standard Corpus



Gold Standard
Corpus (GSC)



High-quality GSC is used for developing NLP tools, training machine learning models, fine-tuning models, and providing the basis for evaluation to test the performance of NLP systems



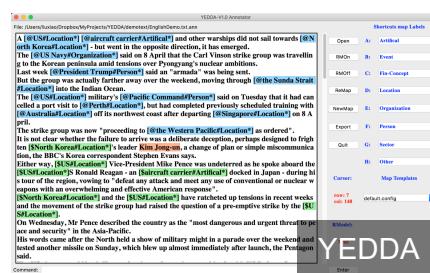
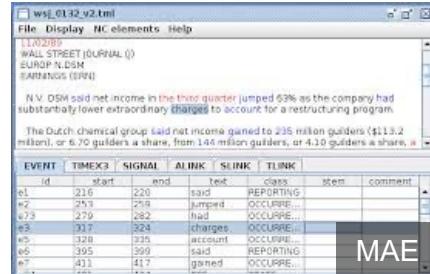
Text Annotation Tool

Building such a high-quality corpus requires considerable time and expertise to go through the whole annotation process with a comfortable annotation tool to facilitate the annotation process.

Background - Existing tools

Standalone tools

- MAEAnalec
- YEDDA
- BioAnnotate
- Callisto
- Ellogon
- Glozz
- MMAX2
- NOMAD
- RAD

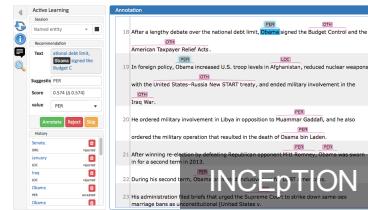
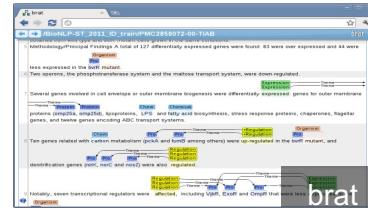


...

Web-based tools

- brat
- INCEpTION
- Anafora
- eHost
- BioQRator
- Catma
- FLAT
- WebAnnoMAT
- TextAE

...



TeamTat

Background - Multi-site Annotation

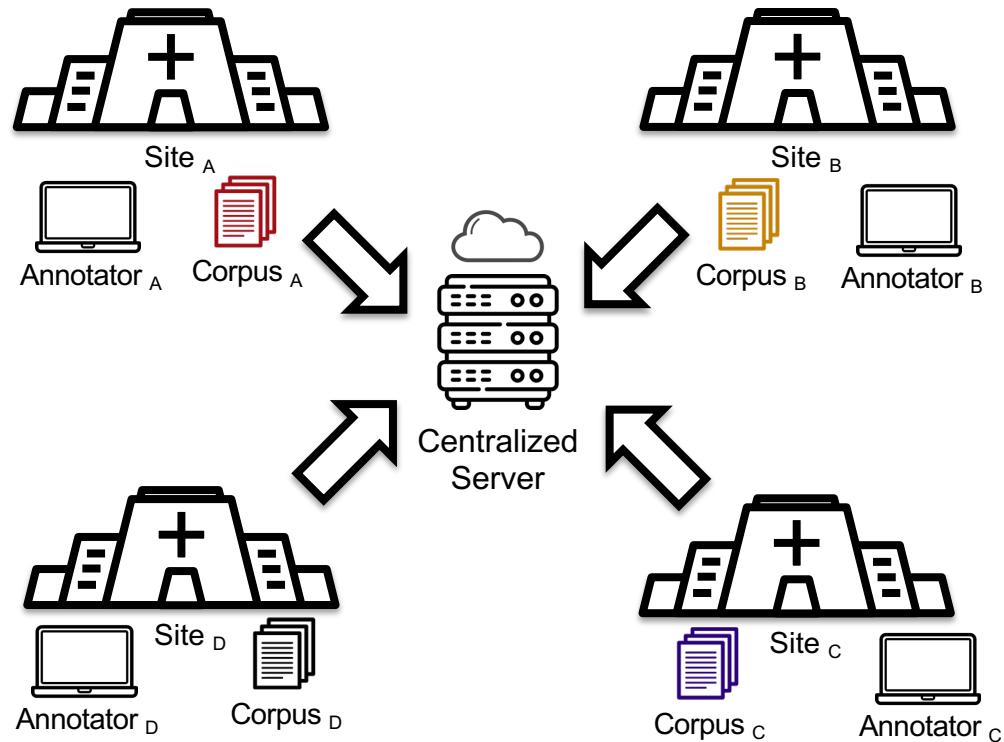
Centralized annotation

Each site uploads raw corpus to a centralized server, and annotators can log in to this server to annotate.

Major concerns

Protected health information (PHI) concerns.

Permission, de-identification, format conversion, etc.



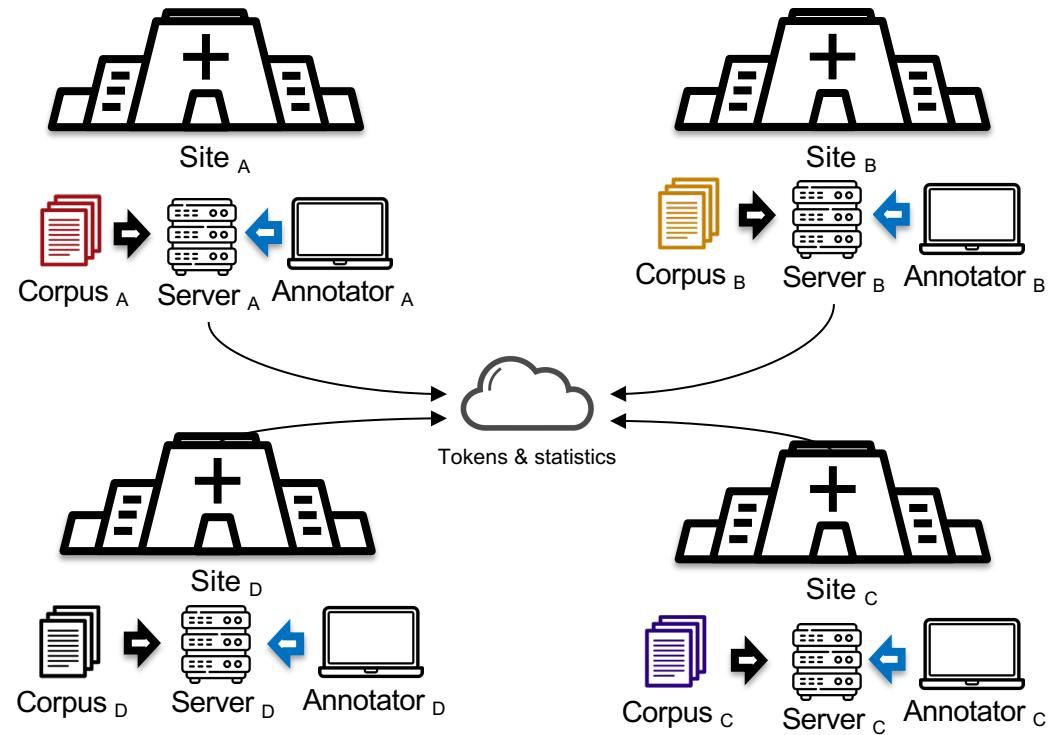
Background - Multi-site Annotation

Own annotation servers

Each site builds its own annotation server and uploads raw corpus to its own server.

Major concerns

Internal server setup, maintenance, management, network permissions, data import and export, etc.



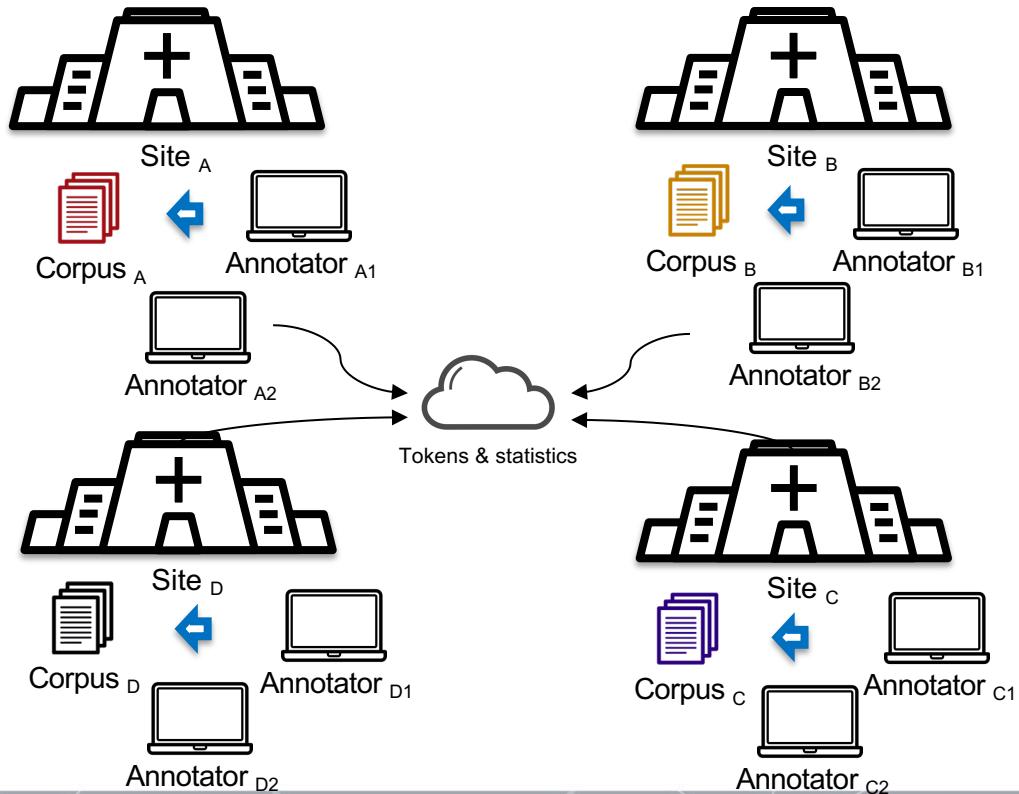
Background - Multi-site Annotation

Standalone tool

Each annotator installs the standalone tool on their local machine and annotates the corpus locally.

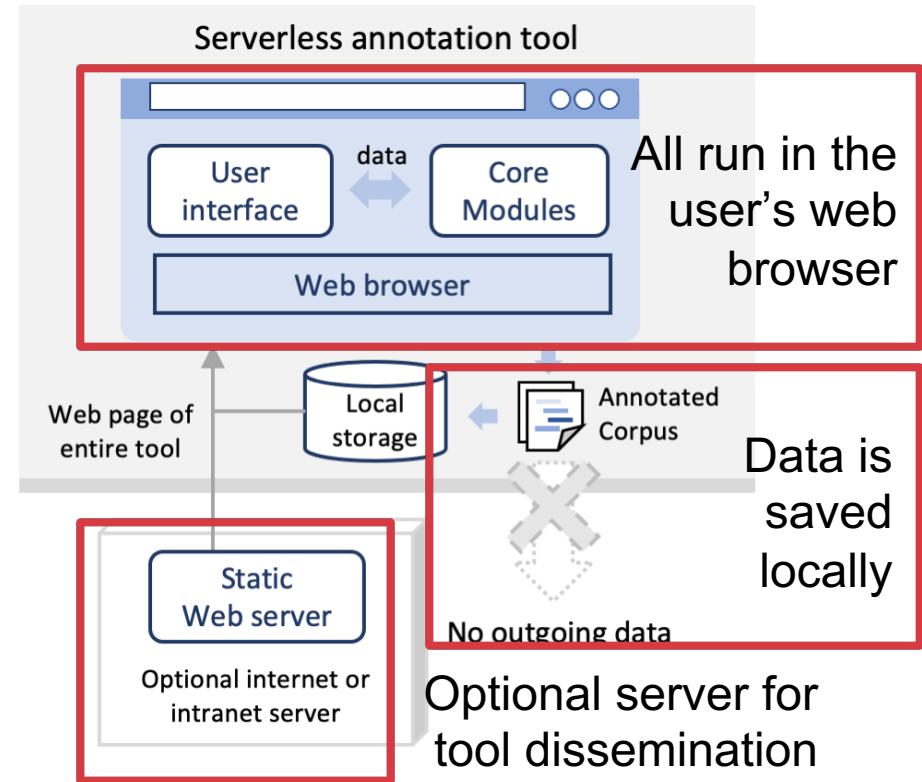
Major concerns

Permission of tool installation, runtime management, tool upgrade, environment upgrade, cross-platform compatibility, etc.



Background - Challenges of multi-site

1. Privacy concerns. Corpus with PHI should never be sent out to any external systems.
2. Concerns related to server resources, runtime setup, deployment, permissions, access controls, etc.
3. Concerns related to local runtime installation, management, upgrade, licenses, etc.



Multi-site Annotation based on MedTator

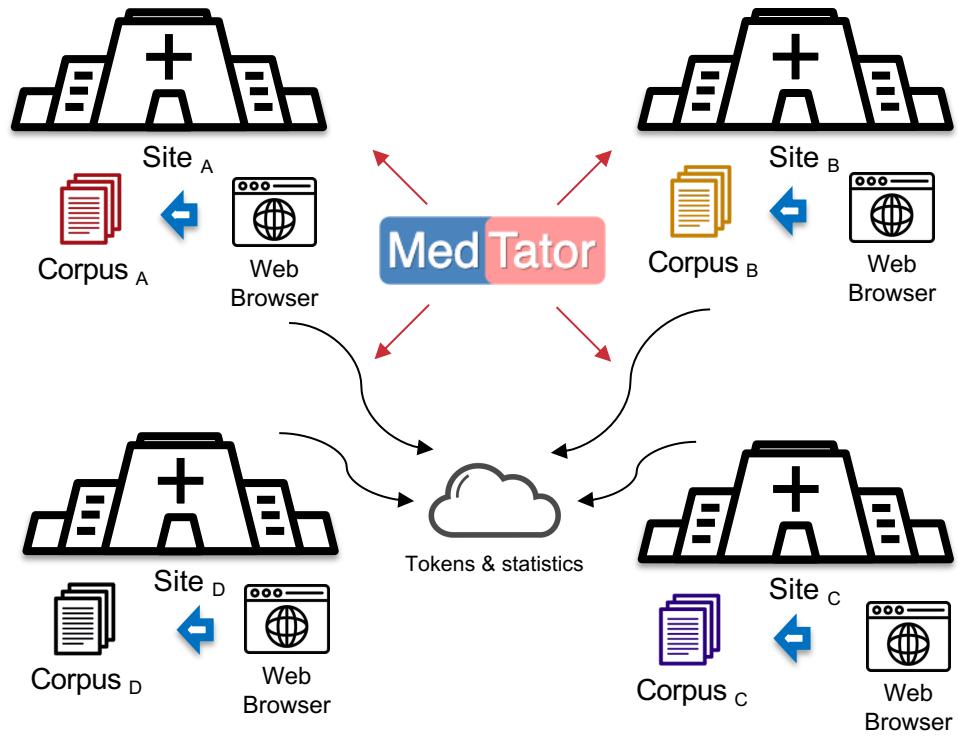
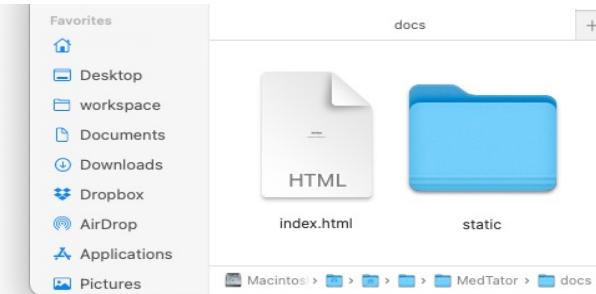
Serverless MedTator

1. No data is sent out:

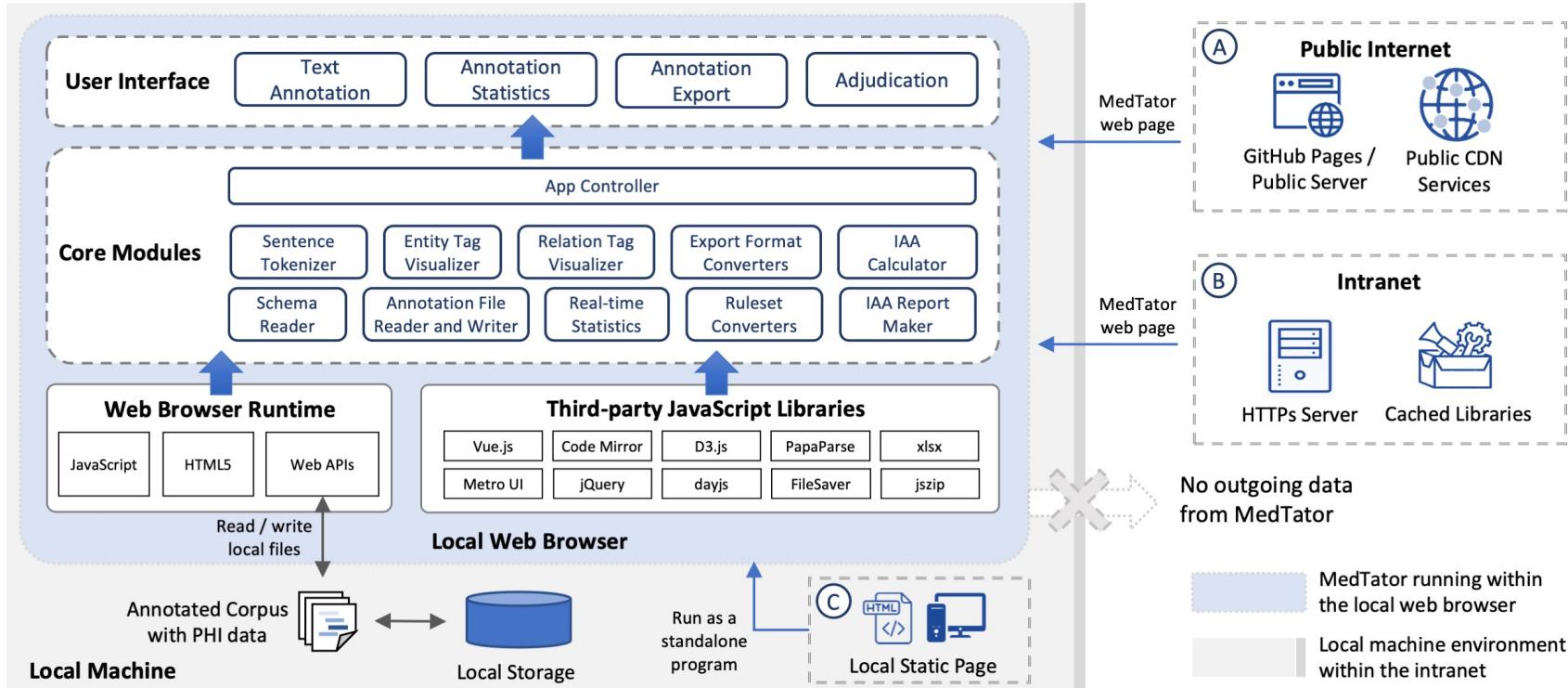
- Run locally in user's web browser
- Operate text files on locally

2. No "install":

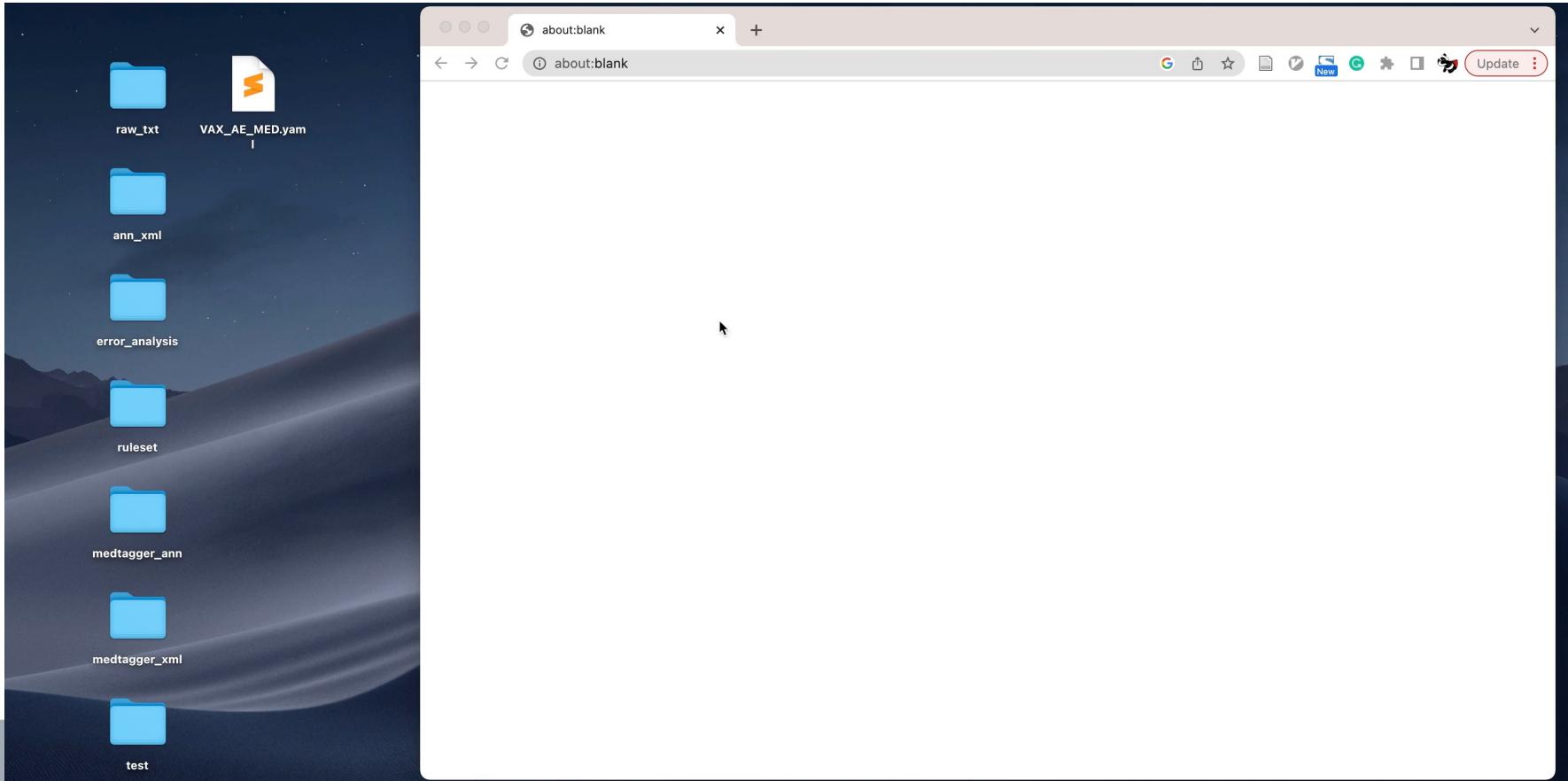
- No need to install any runtimes
(e.g., Java, Python, Nginx, etc.)



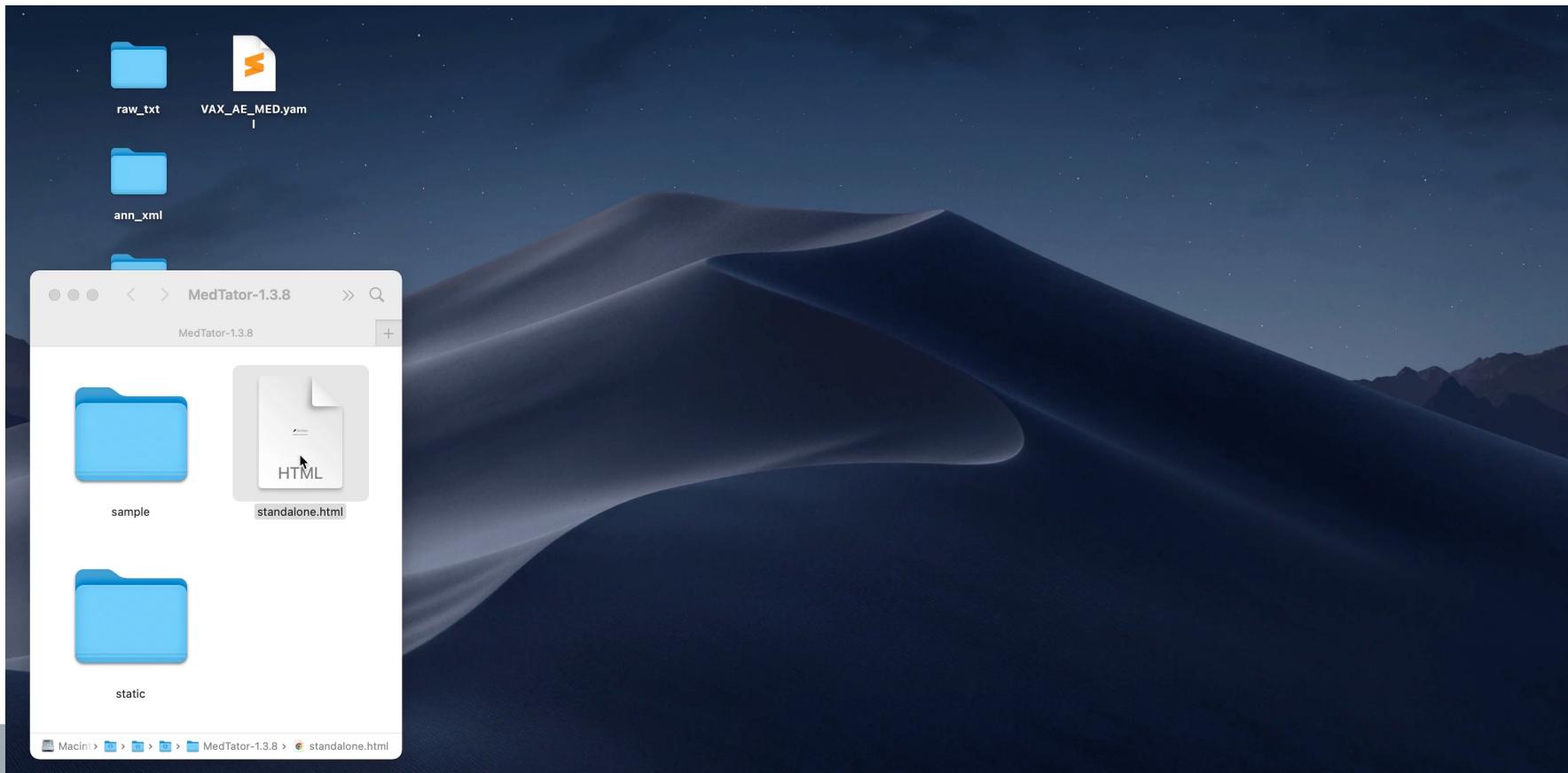
MedTator: Serverless Architecture Design



MedTator - Run as a web App



MedTator - Run as a local App



MedTator - Tool Comparison



In terms of prerequisites for installation, MedTator shows a significant advantage

Tool	Source Code	Prerequisites for server installation on server
BioQRator	https://github.com/dongseop/bioqrator	MacOS or Linux; Ruby, Ruby on Rails, MySQL
brat	https://github.com/nlplab/brat	Linux; Apache 2, GeniaSS, Python 2.5
Catma	https://github.com/forTEXT/catma	Java, Maven, Jetty
Djangology	https://sourceforge.net/projects/djangology/	Python, Django, database server
ezTag	https://github.com/ncbi-nlp/ezTag	Ruby, Ruby on Rails, MySQL
FLAT	https://github.com/proycon/flat	Apache / Nginx, Python, foliadocserve, pynlpl, Django
MAT	http://mat-annotation.sourceforge.net/	Python, Java
PDFAnno	https://github.com/paperai/pdfanno	Node.js
TextAE	https://github.com/pubannotation/textae	Node.js
WAT-SL	https://github.com/webis-de/wat	Java, Docker
WebAnno	https://webanno.github.io/webanno/	Java, Apache Tomcat, MySQL

Neves M, Ševa J. An extensive review of tools for manual annotation of documents. Brief Bioinform. 2021 Jan 1;22(1):146–63.

Lessons we learned



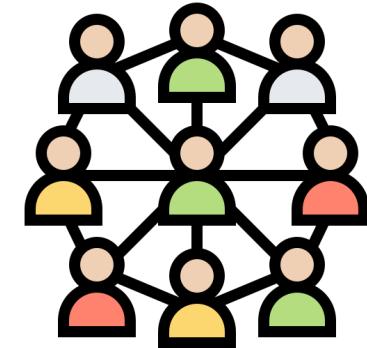
Lesson 1

Apply a user-centered design approach to meet the needs of various users



Lesson 2

Non-functional requirements are critical for adoption



Lesson 3

Community participation makes it thrive

Lesson 3: Community participation



Iterative improvement based on community feedback



Oct. 30 - Nov. 3, 2021

San Diego, CA



Informatics Summit

**March 21-24, 2022 |
Chicago, IL**



National
COVID
Cohort
Collaborative

*Collaborative Analytics
Workstream
Subgroup of
Natural Language
Processing*



N3C NLP: Focusing on Concept Extraction

**N3C: Collaborative Analytics Workstream
Natural Language Processing (NLP)**

July 7th, 2022

@data2health
@ncats_nih.gov
<https://covid.cd2h.org/>

Office hours and forums

Lesson 3: Community participation

Participants from open-source community

On the maximum length of "_ VALUE_TYPE_ "#4

A screenshot of a GitHub issue thread titled "Unable to load schema #10". The issue was opened by shashank140195 22 days ago and has 5 comments. The first comment from shashank140195 says: "When you save the schema which by default gets saved by the .yml extension, you cannot load it back. It is all empty." A reply from hehuan2112 says: "Thank you for your feedback on this issue. By default, the schema editor should save the schema as .yaml format, so if the extension is changed to .yml, the current MedTator couldn't recognize it correctly. You can try to rename your schema file extension to .yaml for now." Another reply from shashank140195 says: "And we just found that there is a bug related to the open schema dialog of the .yaml format. We have added this issue to our next release plan. We will fix the dialog, add the .yml in the supported extension list, and show some information for similar issues instead of just showing empty list."

Report bugs and issues

The screenshot shows the MedTator web application. At the top, there are links for "Manual", "Annotation Schema", and "Home". Below the "Manual" link, it says "Huan He edited this page on Jun 24 · 1 revision". The main content area has a heading "Annotation Tab" with the sub-section "Schema Editor". It says: "This tab allows the user to annotate including: 1. A: the file list view shows the 2. B: the tagging view shows the annotation hints in selected f 3. C: the concept list view show concept annotated in the sel 4. D: the tag list view shows the attributes." To the right, there is a "Foreword" section with the text: "MedTator is a serverless web tool that focuses on the core steps related to corpus annotation." Below that is a "What does 'serverless' mean?" section with the text: "'Serverless' does not mean that you cannot put MedTator on a server (you certainly can access MedTator by any web server), nor does it mean that MedTator lacks any annotation functionality. The 'serverless' means that MedTator can process data 100% within your web browser and no need for a server support." On the left side, there is a sidebar with navigation links: "Pages 8", "Find a page...", "Home", "Foreword", "What does 'serverless' mean?", "Background", "System Architecture", "Open-source Packages", "Run Your Own Copy", "Download Desktop Version", "Fork Online Version", "Comparison with Other Tools", "Annotation Best Practices", "Annotation Data", "Annotation Schema", "FAQ", "Manual", "Prepare Dataset", and "Quick Start". At the bottom, there is a "Manual" link and a "Add a custom sidebar" button.

Wiki for documentations (e.g., manual, annotation schema design, best practice, etc.)

Lesson 3: Community participation

Continuous updating bi-weekly for fixing issues and adding features

1.3.8 (2022-10-27)

1.3.2 (2022-08-04)

1.3.0 (2022-07-21)

1.2.41 (2022-06-23)

1.2.27 (2022-04-14)

1.2.6 (2022-02-03)

- Added schema of .yml extension
- Added auto-save (experiments)
- Fixed scheme editor open bug

- Added functions for error handling
- Added Math.js for static analysis
- Added ECharts for visualization
- Added a sample database
- Updated the UI design

- Refactored file reading workflow
- Refactored annotation file
- Refactored code structure
- Designed JSON/YAML formats
- Added tokenization exception handling

- Added meta-data structure for annotations
- Added color label to annotation file
- Added functions for updating annotator files
- Added download all tags for adjudication
- Added sorting logic

- 1.2.32 (2022-05-26)
- Added sample schema dropdown for editor to select
 - Added sample schema files for standalone version
 - Added help to schema editor with wiki
 - Updated the tooltip information for annotation menu
 - Updated scripts for creating sample DTD files
 - Fixed the initial problem
 - Fixed schema editor
 - Fixed path missing

1.2.15 (2022-03-17)

- Added sort button and menu for annotations
- Added dynamic sort indicator for annotations
- Added a button for quickly delete annotations
- Fixed concept name overflow in the UI
- Fixed ribbon menu layout bug

1.2.24 (2022-03-17)

- 1.2.48 (2022-07-1)
- Added Cohen's Kappa calculation
 - Added dropdown for annotations
 - Added confusion matrix
 - Added schema editor for annotations
 - Added helper functions for annotations
 - Added schema editor for annotations
 - Added dropping file in the filelist
 - Updated the drop zone size
 - Updated the schema and IAA summary
 - Updated the JSON files of annotations
 - Fixed switch button status when switching between annotations

1.2.13 (2022-03-03)

- Added using attribute IAA calculator
- Added menu for attribute selection
- Added dropping file in the filelist
- Updated the drop zone size
- Updated the schema and IAA summary
- Updated the JSON files of annotations
- Fixed switch button status when switching between annotations

1.2.29 (2022-04-28)

- Added the sort for IAA
- Updated IAA result border
- Updated sample dataset

- 1.2.20 (2021-12-12)
- Added the sort for IAA
 - Updated IAA result border
 - Updated sample dataset
 - Fixed the message
 - Fixed the message

1.2.10 (2022-02-17)

- Added a new section in the statistics report
- Added a new statistics report interface
- Added colored cells to the statistics report
- Added importing annotation files by dropping folder
- Added memory check in settings
- Updated the statistics summary with more items
- Updated the XML zip file name with timestamp

1.2.0 (2021-12-12)

- Added hover border for annotations
- Added show action for annotations
- Added download action for annotations
- Added fixed tab for annotations
- Added download action for annotations

1.0.2 (2021-11-08)

- Added a sentence splitting / tokenization method for fast splitting.
- Added a side bar for managing settings.
- Added entity tag locating. Users could click on the spans to locate the entity tag in the editor. The editor will jump to the line where the clicked tag is and highlight that tag.

1.0.1 (2021-10-15)

- Added sample data for AMIA 2021 workshop.

Acknowledgments



Co-authors and team members



Sunyang Fu



Liwei Wang



Andrew Wen



Sijia Liu



Sungrim Moon



Kurt Miller



Hongfang Liu



Donna Ihrke



Katelyn Cordie



Samuel McKinven

Community Participants



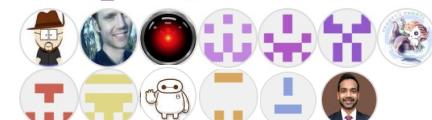
INFORMATICS PROFESSIONALS. LEADING THE WAY.

OHNLP



National
COVID
Cohort
Collaborative

*Collaborative Analytics
Workstream
Subgroup of
Natural Language
Processing*



This work was supported by the National Center for Advancing Translational Sciences of the National Institutes of Health under award number U01TR002062.

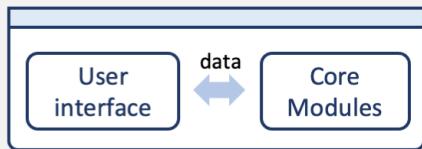
Thank you!

Email me at: He.Huan@mayo.edu



Appendix - Architecture Comparison

(A) Standalone annotation tool



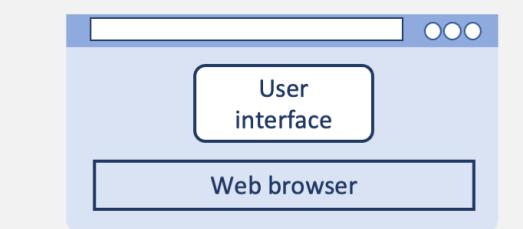
Local Machine

No outgoing data

Local machine environment within the intranet

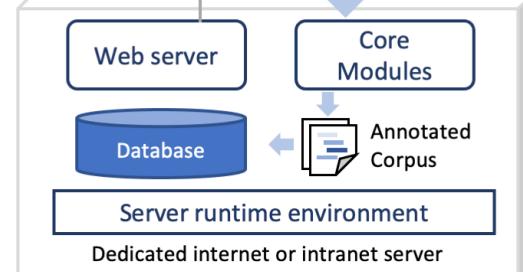
Server environment on the internet or intranet

(B) Typical web-based annotation tool



Web page of user interface

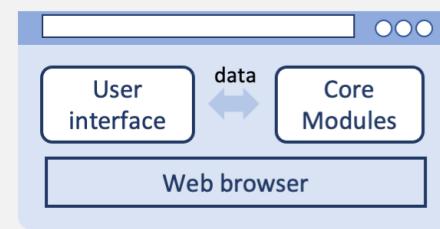
Documents and annotations



Server runtime environment

Dedicated internet or intranet server

(C) Serverless annotation tool



Web page of entire tool

Static Web server

Optional internet or intranet server

No outgoing data

Appendix - Limitations

Web browser file system access API is needed to support core features

mdn web docs References Guides

References > Web APIs > File System Access API

RELATED TOPICS

- File System Access API
- Interfaces
 - [FileSystemHandle](#)
 - [FileSystemFileHandle](#)
 - [FileSystemDirectoryHandle](#)
 - [FileSystemWritableFileStream](#)
- Methods
 - [window.showOpenFilePicker\(\)](#)
 - [window.showSaveFilePicker\(\)](#)
 - [window.showDirectoryPicker\(\)](#)
 - [DataTransferItem.getAsFileSystem](#)

File System Access API

Secure context: This feature is available only in [secure contexts](#) (HTTPS), in some or all [supporting browsers](#).

The File System Access API allows read, write and file management capabilities.

Concepts and Usage

This API allows interaction with files on a user's local device, or on a user-accessible network file system. Core functionality of this API includes reading files, writing or saving files, and access to directory structure.

Most of the interaction with files and directories is accomplished through handles. A parent [FileSystemHandle](#) class helps define two child classes: [FileSystemFileHandle](#) and [FileSystemDirectoryHandle](#), for files and directories respectively.

These handles represent the file or directory on the user's system. You must first gain access to them by showing the user a file or directory picker. The methods which allow this are [window.showOpenFilePicker](#) and [window.showDirectoryPicker](#). Once these are called, the file picker presents itself and the user selects either a file or directory. Once this happens successfully, a

Browser compatibility

[Report problems with this compatibility data on GitHub](#)

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	WebView Android	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet
FileSystemHandle <small>⚠</small>	✓ 86	✓ 86	✗ No	✗ No	✓ 72	✓ 15.2	✗ No	✓ 86	✗ No	✗ No	✓ 15.2	✓ 14.0
isSameEntry <small>⚠</small>	✓ 86	✓ 86	✗ No	✗ No	✓ 72	✓ 15.2	✗ No	✓ 86	✗ No	✗ No	✓ 15.2	✓ 14.0
kind <small>⚠</small>	✓ 86	✓ 86	✗ No	✗ No	✓ 72	✓ 15.2	✗ No	✓ 86	✗ No	✗ No	✓ 15.2	✓ 14.0
name <small>⚠</small>	✓ 86	✓ 86	✗ No	✗ No	✓ 72	✓ 15.2	✗ No	✓ 86	✗ No	✗ No	✓ 15.2	✓ 14.0
queryPermission <small>⚠ ⚠</small>	✓ 86	✓ 86	✗ No	✗ No	✓ 72	✗ No	✗ No	✓ 86	✗ No	✗ No	✗ No	✓ 14.0
requestPermission <small>⚠ ⚠</small>	✓ 86	✓ 86	✗ No	✗ No	✓ 72	✗ No	✗ No	✓ 86	✗ No	✗ No	✗ No	✓ 14.0

✓ Full support ✗ No support ⚠ Experimental. Expect behavior to change in the future.

⚠ Non-standard. Check cross-browser support before using.