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

Huan He, PhD, Sunyang Fu, PhD, Liwei Wang, MD, PhD, Andrew Wen, MS, Sijia Liu, PhD, Sungrim Moon, PhD, Kurt Miller, MS, and Hongfang Liu, PhD
Department of Artificial Intelligence and Informatics, Mayo Clinic, Rochester, MN, USA

Abstract

A gold standard annotated corpus is usually indispensable when developing natural language processing (NLP) systems. Building a high-quality annotated corpus for clinical NLP requires considerable time and domain expertise during the annotation process. Existing annotation tools may provide powerful features to cover various needs of text annotation tasks, but the target end users tend to be trained annotators. It is challenging for clinical research teams to utilize those tools in their projects due to various factors such as the complexity of advanced features and data security concerns. To address those challenges, we developed MedTator, a serverless web-based annotation tool with an intuitive user-centered interface aiming to provide a lightweight solution for the core tasks in corpus development. Moreover, we present three lessons learned from the designing and developing MedTator, which will contribute to the research community's knowledge for future open-source tool development.

Introduction

Natural language processing (NLP) has been widely applied in healthcare-related domains for clinical practice and research field [1,2]. While a high-quality annotated corpus is usually indispensable for developing NLP models, building such a corpus requires considerable time and expertise to go through the whole annotation process with a comfortable annotation tool[3]. Consequently, there have been many comprehensive annotation tools developed in the past to facilitate the annotation process [3], and they can provide many advanced features to help users reduce the burden of annotation in many ways. Most of the tools are open-source that can be obtained freely.

For example, brat[4] is a popular web-based tool with advanced visualization functionalities. PubTator Central integrates PubMed to provide many features such as literature search and biological entity identification for biomedical literature. ezTag[5] supports annotating PubMed abstracts and PubMed Central full-text articles in BioC format with customized lexicons. INCEpTION [6] is a general-purpose annotation tool that incorporates weak labels generated from existing NLP systems. TeamTat[7] provides the collaborative annotation ability to help the researcher to manage a team annotation project. MyMiner supports the biomedical document annotation task with pre-defined biological entities such as proteins and genes[8]. LabelStudio[9] provides application programming interfaces (APIs) for developers to integrate the annotated data with other systems. In addition to the above-mentioned open-source tools, there are also a lot of commercial web-based tools, such as prodigy, tagtog, LightTag, etc. Those tools can provide more advanced features such as multi-media data annotation, assistance based on machine learning, online team collaboration, etc.

However, while existing tools provide many powerful features, it is still challenging for research teams to leverage them in their own research task due to three main factors. First, while existing tools bring many advanced corpus annotation features, annotators must spend extra effort to learn, evaluate, and apply those features to their own tasks. Secondly, existing web-based tools often need a correctly configured web server environment as one of the prerequisites, which requires extra technical support. Finally, and most importantly, due to patient privacy concerns, clinical documents containing protected health information (PHI) cannot be sent to external servers, and users usually can only annotate the documents in a restricted environment (e.g., no internet and no permission of tool installation). As a result, the ability to run in a restricted machine is required for an annotation tool to be adopted in clinical settings.

To address these challenges, we developed MedTator[10], a serverless web-based annotation tool aiming to provide an intuitive and interactive user interface that focuses on the core steps of corpus annotation. More importantly, there are three lessons we learned during the development and maintenance, which are about the needs of users, the non-functional requirements, and the community participants. These lessons we learned are applicable to a range of similar tools and thus benefit the clinical research community.

We have released the source code on GitHub under the Apache 2.0 license: https://github.com/OHNLP/MedTator. The documentation, such as the user manual and sample annotations can be found in the MedTator wiki in the same GitHub repository. In addition, a public online demo of MedTator is also available with several sample datasets at https://ohnlp.github.io/MedTator.

System Design

Motivation, methods, and system features

Clinical document annotation was always demanding in our past clinical NLP research projects, and one of the most valuable assets of these projects is the annotated corpus. To fully utilize those annotated corpora, a tool is needed to explore the documents and the annotated tags for corpus analysis. Therefore, MedTator was initially developed as a web-based visualization tool for exploring the annotated corpus.

Over time, additional new features such as revising annotated tags, tag statistics, and loading customized schema, were added to meet the needs of different projects. The newly added features were welcomed by researchers, which suggests that MedTator can be further improved into a text annotation tool. Then, we began the design process to shape the functional requirements and non-functional requirements for MedTator as follows.

Firstly, we collected and installed popular annotation tools from literature reviews to understand the workflow and user experience of existing tools to form initial design ideas.

Secondly, we invited researchers who have extensive experience in NLP and clinical research as domain experts and interviewed them to assess our initial ideas and the prototype. With their suggestions, we summarized the goal of MedTator as "a lightweight, easy-to-use text annotation tool" and the core value is simplicity, which guides the design decisions in the development process.

We iterated through weekly design and development cycles to discuss the latest progress and gather feedback from our domain experts and early users. Each cycle began with a prototype demonstration of the latest features developed (e.g., new functionalities or visual designs), and the latest prototype will be uploaded to our internal server to be tested by domain experts and users. The comments on the prototype will be collected to form new functional and non-functional requirements. Those new requirements were evaluated by our domain experts to decide whether to be included in the development plan or not. In addition, we also held training sessions and development meetings with annotators and researchers to review the features and issues as needed. Their comments were also utilized in the design and development. To manage the workload and roadmap, we used GitHub Project and Microsoft Teams to track the design and development tasks.

After about 3-4 months of iterative design and development cycles, the prototype was upgraded to a fully functional annotation tool with the following core features implemented to meet the functional requirements:

- **Text annotation**: this is the core task in the whole annotation process. MedTator provides core functionalities to meet the needs of this task, such as annotated file parsing and saving; text span annotation at levels of characters, words, sentences, and documents; semantic relationship annotation between multiple text spans; multiple document annotation; and customizable annotation schema for various tasks.
- **Annotation statistics**: the statistics on annotated tags are demanded by researchers. MedTator supports statistics on the annotated corpus from multiple aspects, which can be used in all stages of the annotation process to understand the annotated corpus. For example, at the initial stage of the annotation guideline development, the overall statistical results on the small size sample corpus can be used to revise the schema and guideline; during the iterative annotation, the results can be useful for tracking the annotation process; and in the final evaluation stage, the results can help to evaluate the annotation quality.
- **Adjudication**: to resolve the inconsistencies among annotations from different annotators, MedTator supports adjudication on multiple annotation versions to help researchers to create the gold standard corpus. For example, the measurement of the inter-annotator agreement (IAA) is needed to help researchers estimate the overall annotation quality. Moreover, the agreements among multiple annotation versions need to be measured from multiple levels (e.g., overall agreement, concept level, document level, tag level, attribute level, etc.) to help researchers locate and investigate the disagreements.
- **Annotation export**: MedTator supports the conversion of the annotated corpus to different formats. The annotated corpus itself is usually not the final goal of our research projects but for the corpus analysis and system development. According to our previous experience in clinical research, the annotated corpus can be used in many ways, including but not limited to: corpus analysis to understand the cohort[11], knowledge discovery [12], training machine learning or deep learning models[13], and building NLP systems [14], etc. Therefore, multiple formats, such as CSV, BioC [15], and MedTagger rulesets [16], are supported by MedTator for various downstream tasks or other software.

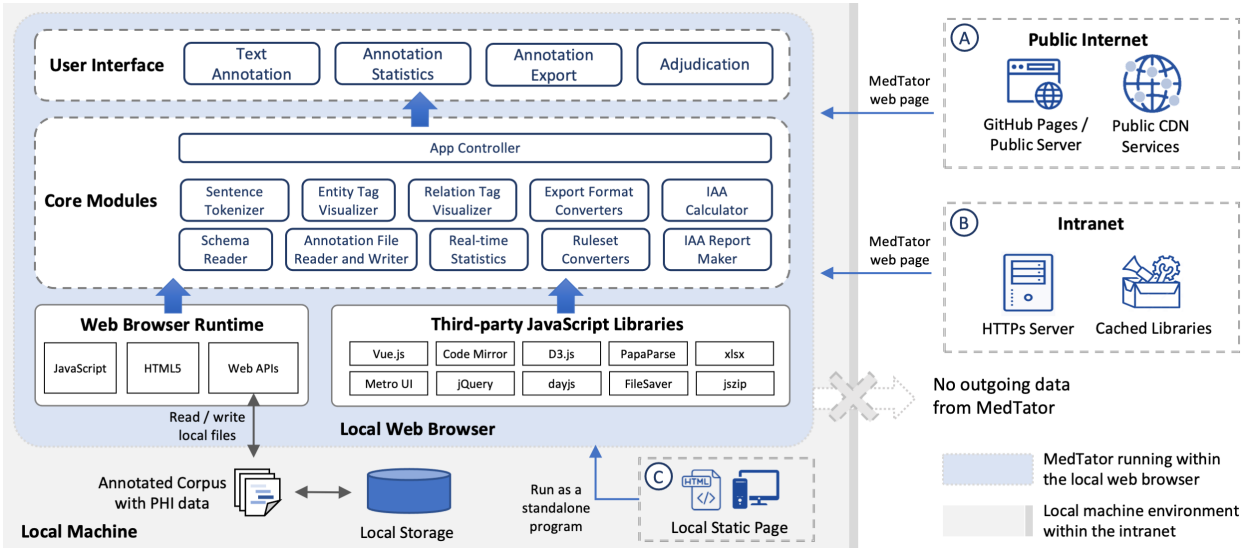

Figure 1. The MedTator architecture based on a serverless design. MedTator can be accessed as a web page from (A) public internet using GitHub pages or other public server, or (B) any static HTTPs server in the intranet. (C) It can also run as a standalone program locally without any network access. All functionalities are implemented based on web frontend techniques within the web browser environment. MedTator can only access local storage for read / write annotated corpus without sending / receiving out any data outside of local machine.

In addition to the above features for the functional requirements, we summarized the non-functional requirements, such as data security, maintainability, and easy-to-install. To meet these requirements, we adopted a serverless architecture to design MedTator. The details are described in the following sections.

Serverless architecture

Based on the above-mentioned requirements and our expertise in software engineering, we adopted a serverless architecture to design MedTator. Serverless is an emerging application design that removes much of the need for a traditional always-on server and incorporates third-party services[17]. Using this design, MedTator works like a single-page application, which is assembled as one HTML file with several external JavaScript libraries. Users can access it from any web server through HTTPs protocol or locally run it as a standalone program.

As shown in Figure 1, both the user interface and core modules of MedTator are implemented as a single-page web application within a web browser environment based on pure frontend web techniques, including JavaScript, HTML5, web APIs, and many JavaScript libraries such as Vue.js, D3.js, and CodeMirror. The user interface contains customized visualization and interactivity designs that collect user input and present the processed data. The core modules support the core functionalities described in the previous section. The data exchange between the user interface and the core modules is completely restricted to the user's local web browser, and all information of the annotated corpus is saved in the user's local storage to avoid any leakage of PHI.

As a serverless tool, this architecture allows users to use the tool without managing servers. The whole tool is packaged into a single HTML file, which can be accessed in three ways: 1) as a public webpage served by GitHub Pages or any other public server with the third-party libraries provided by public content delivery network (CDN) (Figure 1A); 2) as an internal webpage served by a HTTPs server on intranet website with cached third-party libraries (Figure 1B); and 3) as a static webpage that can be opened by local web browser directly (Figure 1C). All the dependent third-party libraries can be loaded through local disk, and all the traffics of the network can be monitored in the web browser console to ensure data security.

The serverless architecture brings the following advantages to MedTator. First, there is no need to manage a server to host the tool and install any programming runtime environment at all, such as Java, Python, or Node.js on the user's local machine. Secondly, end users can easily access the program in a secure way. Last and most important, no bit of data would be sent out from the user's local machine which is usually behind firewall to address security concerns.

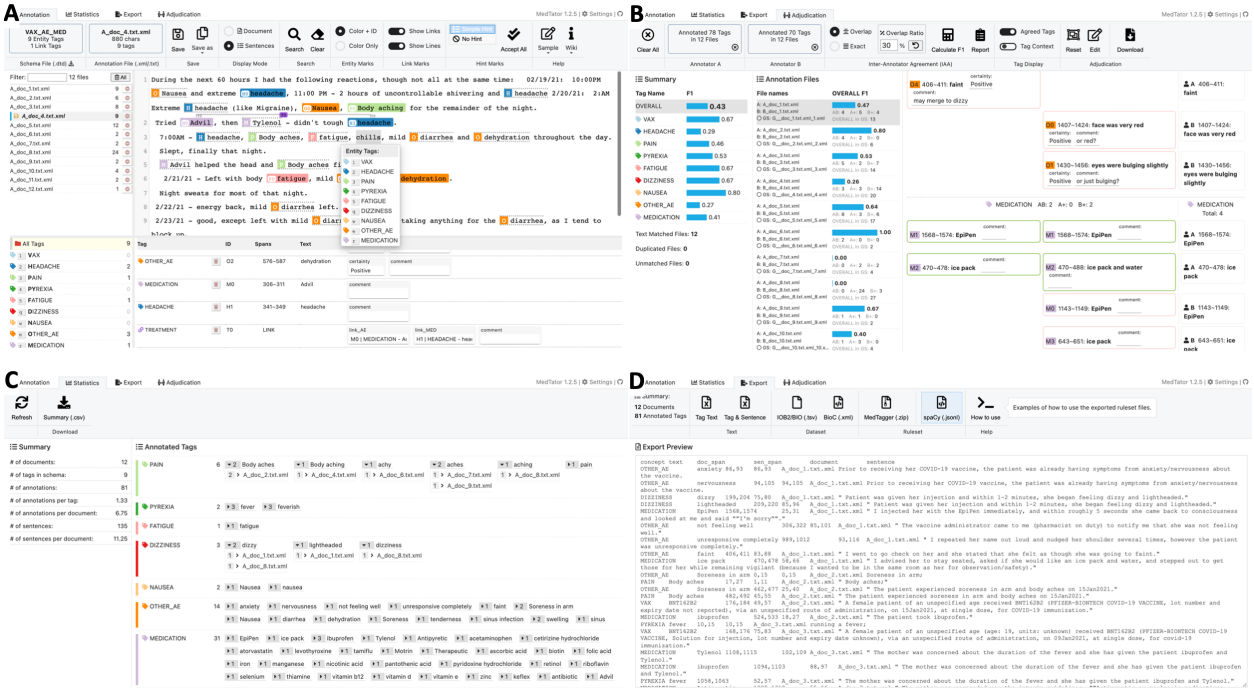

Figure 2. Screenshots of the four tabs in MedTator, including (A) the annotation tab showing multiple documents, visualized tags, and annotated tag list with details; (B) the adjudication tab showing the inter-annotator agreement in different levels with detailed tag context from two annotators; (C) the statistics tab showing the overall status, and detailed annotated tags of each concept; and (D) the export tab for converting the annotations in different formats.

User interface design

As shown in Figure 2, for the four core tasks identified in the annotation process, we designed four tabs to support the activities needed for each task. All four tabs follow the same design style and layout, which splits the interface into three parts, the top bar, ribbon menu, and workspace panel, from top to bottom, respectively (large figures and online demo can be found at our GitHub repo https://github.com/OHNLP/MedTator). Users can switch different tabs in the top bar, then the ribbon menu will be updated to show the related functionalities, and the workspace panel will be updated to show specific panels for the corresponding task. The details of each tab are as follows:

Annotation tab. This tab allows the user to annotate multiple documents according to a schema through four coordinated views (Figure 2A), including the file list view, tagging view, concept list view, and tag list view. The tagging view is the main interface for annotation, which allows users to add or delete tags on the selected document. At present, MedTator supports three types of text annotation in this tab: 1) span-based entity annotation, 2) entity-based relation annotation, and 3) document-level annotation. To reduce the repetitive workload, MedTator can optionally show hints on potential words of interest based on real-time statistics of the annotated tags. It takes a simple one-click on the hint box to automatically add a new tag or accept all hints as to new tags. In addition, the real-time statistical results will also be displayed on the file list view and concept list view for tracking annotation progress.

Adjudication tab. This tab allows users to compare annotations of two annotators to create a gold-standard corpus (Figure 2B). F1-score is used for IAA calculation to assess the reliability of annotations[18], and the detailed results are visualized at different levels for interactive exploration, which can be downloaded for further analysis. In addition, users could accept or reject each annotator's annotation to generate the gold-standard corpus for further tasks.

Statistics tab. The detailed information of the real-time statistics on the annotated tags is shown in this tab, allowing users to analyze both the overall status and the annotated tags from concept and document aspects (Figure 2C). Users can also trace back to the source document of each annotated tag in the annotation tab through hyperlinks.

Export tab. This tab allows users to convert the annotations to different formats for downstream tasks (Figure 2D). For example, the annotated tags and context sentences can be converted to a ruleset to build a rule-based system based on open-source tools (e.g., MedTagger[16] and spaCy), an inside-outside-beginning format file for training models for named-entity recognition, a tabular format file, or a BioC format[15] file for other NLP tasks.

Lessons learnt

During the process of design and development of MedTator, we have learnt several lessons. Those lessons would help the design and development of informatics solutions.

Lesson 1: Apply a user-centered design approach to meet the needs of various users

Our previous studies analyzed the activities in the annotation process and summarized the best practices[11,19]. To go through those activities and build a high-quality corpus, considerable time and expertise of the annotators are necessary. However, annotators are not the only stakeholder in the whole annotation process. In addition to annotators, we found that researchers and administrators are also involved in the annotation process with different goals and needs. Therefore, rather than developing a huge "all-in-one" platform to cover all activities for every stakeholder, developing a lightweight tool that focuses on the core activities is more feasible.

As a result, MedTator was designed to address the four core activities: text annotation, statistics, adjudication, and annotation export. For these four activities, a multidiscipline team with various skillsets is usually needed to meet the requirements of each activity, and their user requirements for the tool are also different.

The power user: researchers

Researchers are deeply involved in each task of entire corpus development and thus have a strong demand for an annotation tool to accelerate the whole process. They have advanced skills in informatics and rich experience in applying NLP in clinical studies. They make the decisions for each task to meet the research needs, as well as what tool to use for annotation. As a result, researchers of our team were the initial users since the early beginning of the development. We interviewed them and other researchers from the community to understand the needs in corpus annotation and the gap between those needs and existing tools as described in the System Design section.

In addition to the functional requirements which have been addressed in the core features, the following findings are also considered to inspire the tool design. First, project needs vary. We find that most of the needs in annotation projects are similar, but each project has a few special needs to meet its own research goals, such as ontology mapping or normalization. Those special needs can be addressed by developing new features, but since the goal of MedTator is a lightweight annotation tool, we need to evaluate whether the new feature can be applied to other projects and how it affects the user experience. Secondly, the resource is limited. Resources, such as computation infrastructure, budget, workforce, time, etc., seem to be always highly demanded but limited for every project. Therefore, when designing MedTator, we aim to offer a lightweight solution built upon public services (e.g., public web server and open-source libraries, etc.) to save the cost for both the team and users.

The end users: annotators

Annotators are the direct end user of an annotation tool. Unlike those annotation tasks of public corpora such as Twitter posts and news reports, the growing length of the clinical documents and the complexity of the contextual knowledge pose challenges to the annotators. Therefore, clinical document annotation heavily depends on experienced annotators with rich expertise in specific domains. They spend considerable efforts on the annotation tasks with the annotation tool; thus, any improvement on the user interface can thus have a significant impact on their experience and work efficiency.

According to the feedback from the annotators of our team and the community, we found that ease of use seems to be the only need for the annotation tool. Especially in the clinical corpus annotation, the annotators with clinical domain knowledge usually are not trained as technical experts. They stated that the annotation tasks would be challenging and very stressful if the tool was hard to use. Therefore, it is necessary to put them in the design cycles and take their comments into the design considerations for an easy-to-use user interface.

However, the ease of use is not just about the user interface or the usability of the annotation tool. In fact, it involves many aspects throughout the annotation process. For example, when new annotators join a project, they may not have any experience in the annotation tool and the corpus development. The first impression of the ease of use comes from the learning materials, which we provided in the training sessions, for helping them learn the basic operations. We prepared several types of user documentation for different purposes, such as a quick start to get a basic understanding of what annotation is, tutorial, and technical specifications. During the annotation, annotators may encounter various issues, such as program bugs, document management, version upgrade, questions about the tool, etc. These issues need to be quickly addressed and explained to annotators to help them. Therefore, we held training sessions, online forum, and email groups to collect feedback and discuss solutions.

The invisible users: system administrators

Because of the critical concerns of the sensitive PHI data in clinical research projects, system administrators are often involved in handling the management of computation environments and data access permissions. During the design and development cycles, we received comments from domain experts and community users about the server requirements and tool installation with the concerns of data privacy, security, and server permissions. Those comments indicated that the needs of system administrators should also be taken into consideration for tool development, especially in the clinical setting.

Unlike researchers and annotators, administrators usually won't directly use the functional features of the annotation tool, such as text annotation and adjudication, but they need to figure out how to install and maintain the tool for researchers and annotators. They thus need to have the specifications of the technical requirements, such as installation manual, runtime requirements, database access, and tool configurations. Moreover, they need to address the data security concerns and the system permissions requirements (e.g., machine, network, software installation, data access, etc.) with the IT department. Therefore, we adopted the serverless architecture to design MedTator to ease the burdens of tool installation on administrators and other users.

Lesson 2: Non-functional requirements are critical for adoption

As described in the previous section, in addition to the functional requirements, non-functional requirements, such as easy to install, easy to use, documentation, and data security, are frequently mentioned in users' comments. We found that those non-functional requirements can form the basis of user experience and the evaluation of the tool, which are critical for the tool adoption decision. The main non-functional requirements we learned are summarized as follows:

Ease of installation as the first positive impression

The comments from the domain experts and administrators indicate that the first barrier to using an annotation tool is often its installation. We investigated existing tools and found that the tool installation also depends on the tool architecture, which determines how the frontend (i.e., the user interface that collects user inputs) and the backend (i.e., the core modules that handle the data processing and calculation) communicate and how the data are saved. Therefore, we plan to address this ease of installation from the perspective of system architecture.

Existing tools' architectures can be divided into standalone tools and web-based tools. For those standalone tools, both the user interface and the core modules run on the user's machine (Figure 3A), and users need to install a runtime (e.g., Java or Python) to use the tool. However, the versions of the runtime and libraries are often confusing and sometimes even conflict with the existing environment in the user's machine. Managing multiple versions of the runtimes and avoiding conflict of those runtimes are not easy, even for experienced users. It can be more difficult for non-technical users to solve those installation issues on their own.

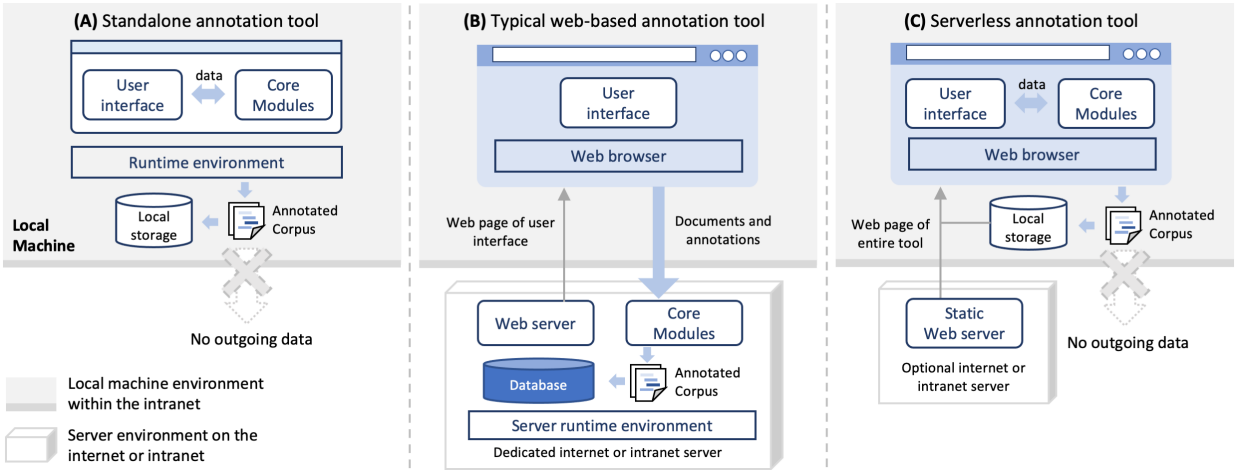

Figure 3. The architecture differences among MedTator and existing annotation tools. (A) The architecture of standalone annotation tool that requires local runtime environment and won't send out data. (B) The architecture of typical web-based annotation tool that requires server runtime environment, and the data need to be sent out to server for processing and storage. (C) The serverless architecture of MedTator that doesn't require any installation of runtime and won't send out any annotation data to outside just like a standalone tool.

For those web-based tools, the user interface (i.e., webpages) runs in the web browser, while the core modules run on the server (Figure 3B). It is much easier for the annotators as they do not need to install the runtime, but it will be more complex for administrators due to the server-side installation.

Therefore, to address the installation issue, we adopted a serverless architecture to design MedTator as introduced in the System Design section. As shown in Figure 3C, both the frontend and the backend run in the web browser; therefore, they do not require any extra installation of a runtime environment on a local machine or server. Consequently, the installation of MedTator is as simple as unzipping a downloaded zip package without any further steps. Users can just double-click the HTML file or visit the public web page to start annotation.

Usability

The usability of the annotation tool directly impacts on the user experience, which may affect the workload and efficiency of annotators. Therefore, the usability of MedTator is always considered when we are developing any feature and making design decisions. We adopted core human-computer interaction (HCI) design principles to guide the tool design according to the comments on existing annotation tools by our domain experts and experienced annotators. The core HCI design principles are as follows:

Reducing cognitive load: the cognitive load imposed by the user interface is the amount of mental resources that is required to perform a certain task with the user interface[20]. And the cognitive overload may be more likely to happen during annotation of the clinical documents because of the large number of clinical narratives and complex annotation guidelines. It is thus challenging for annotators to handle the annotation task comfortably without an easy-to-use annotation tool. We aim to apply this design principle in designing interfaces that decrease extraneous cognitive load so annotator's efforts can be devoted to the actual annotation tasks rather than other distracting tasks. For example, we decided to only include the core four features (e.g., text annotation, statistics, export, and adjudication) related to corpus annotation in MedTator to reduce the extra burdens on users of learning complex features. And for each feature included, we designed separated tabs to show the necessary information related to the annotation task with the same design language. Each tab only provides those necessary functionalities on one single webpage.

Low physical effort: users need to conduct several interaction steps to accomplish a task, and the tool should not involve making "unnecessary" interventions by the user. To minimize the number of steps in each task, we studied the existing workflow and optimized the interactive process for each task. Then, we use this design principle to guide the interactive design of specific operations. And sometimes, the first principle can also be applied together with the second principle to reduce the unnecessary interactions by reducing unnecessary cognitive load. For instance, new tags can be added by clicking shortcut key which is displayed with the tag name; when hovering any annotated tag, the detailed attributes will be displayed next to the hovered tag; the statistical results of the annotated tags are updated in real-time, so that users can track the annotation progress easily.

Flexibility in use: the tool should provide various functionalities to support different project preferences. Therefore, we designed and developed many features, including text span annotation at levels of characters, words, sentences, and documents; relationship annotation between multiple text spans for building semantic relationships; tag visualization; multiple document annotation; and customizable annotation schema for various tasks, etc. This design principle sometimes conflicts with the first principle, so we kept both principles in mind and tried to balance both of them when making design decisions. For example, according to the feedback on our prototypes, some users comment that the automatically generated annotation tags are useful because it helps the identification of all potential tags, while some users comment that these tags may lead to bias and confusion. From the perspective of the third principle, adding the automatic tag feature could be useful, but it also may sometimes confuse users from the perspective of the first principle. To balance the needs for this feature, we designed a group of radio buttons, which can turn on or off this feature in one click and update the tag visualization in real-time.

Documentation

As described in lesson 1, documentation is highly demanded by all users, but the creation and maintenance of documentation are often challenging for the team due to the massive workload of writing. To partially ease the workload, we summarized the comments from domain experts and our experience as follows.

First, identify the types of documentation and prioritize the most needed documents. A common consensus is that the documentation has great practical benefits in software development[21], and there are different types of documentation that have different usage at different stage[22]. For example, technical documentation such as code comments, design principles, and API references, are only helpful for developers in programming and maintenance. In contrast, user documentation can be helpful to a broader range of readers. With the limited resources, focusing on the user

documentation rather than the technical documentation can be a feasible way to reduce the burden of writing. In our case, we paid more attention to the quick start, user manual, and annotation task-related documents, whose target readers are annotators, researchers, and system administrators.

Secondly, user documentation should be designed to accommodate different users for different needs. For example, when we introduce MedTator to other researchers and annotators, they usually ask for some "learning materials" for different learning goals. Some users want to check the features of interest, such as what annotation schemas are supported and how to use the annotation file for their downstream tasks, meanwhile some users prefer a short document that goes through the basic annotation process. In response, we prepared different types of user documentation: a quick start with a step-by-step tutorial for showing the basic workflow and necessary features for corpus development; a well-structured instruction manual that focuses on explaining the detailed usage of all functionalities; and several annotated datasets for demonstrating different annotation tasks.

Lastly, keep the documentation up to date. When features and functionalities are introduced or updated in new releases, the affected sections in the document should be updated to avoid confusing users. We are using the GitHub wiki to organize all user documentation, and everyone in our team can contribute to the content.

Lesson 3: Community participation makes it thrive

During the development, we introduced MedTator to the American Medical Informatics Association (AMIA), Open Health Natural Language Processing (OHNLP), and National COVID Cohort Collaborative (N3C) communities through workshops, online meetings, and training sessions. The valuable feedback from various groups convinced us that community participation is the key to making an open-source tool thrive.

First, MedTator benefits through, and because of, community collaboration. Software development involves research, design, thinking, testing documentation as well as communication with end-users. As an open-source project initiated by a small team, our limited resources have to be allocated to the most needed development tasks, such as architecture design and code-writing of main features. Then, the rest tasks, such as testing, training, and documentation, are supported by our collaborators from the community. They tested MedTator in their corpora and reported issues, which greatly improved the quality of MedTator.

Secondly, there are many different user groups in communities, such as students, computer science researchers, clinicians, informaticians, software developers, NLP experts from the industry, etc. They commented MedTator from many aspects, such as user experience, user interface design, visualization and interactivity, system architecture, annotation schema design, annotation workflow, deployment, platform dependency, integration with other tools, data format, metadata extension, and annotation file management, etc. Although our goals and limited resources do not allow us to include all these valuable suggestions in the design and development plan, their insights help us dive into the practical needs of various users and inspire future development.

Discussion

Comparison with other annotation tools

Many text annotation tools have been developed to help users improve their annotation efficiency by providing many advanced features, such as pre-annotation and online collaboration, which are not available in MedTator. However, in terms of prerequisites for installation, MedTator shows a significant advantage.

Table 1. Open-source annotation tools and their prerequisites for installation on a server.

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

* Commercial tools (prodigy, tagtog, LightTag) and close-source tool (MyMiner) are excluded due to source code unavailability

In the latest review of annotation tools [3], only 15 out of 78 tools were selected for further evaluation, as other tools were excluded because they were either not installable or not workable. We further searched and summarized their prerequisites for installation on the server of those tools. As listed in Table 1, before the installation of those annotation tools, some specific software and libraries must be prepared, which usually requires:

- **A stable server**. A dedicated machine with a stable network connection is usually needed to host the server-side program, such as a physical computer or a virtual machine.
- **Technical expertise**. The installation, configuration, and maintenance of the server environment often require a series of technical expertise, such as the management of Linux server, network and firewall, virtual environment of programming language runtime, and databases administration.
- **Permissions**. Unlike annotating the public corpora (e.g., Wikipedia and PubMed metadata), the data privacy and security of clinical documents are critical concerns for clinical institutions. Administrators need to communicate with the IT department to get permissions such as software installation and network access.

By adopting the serverless architecture, MedTator can save the cost on the above items for researchers. The whole program is packaged into one HTML file, which can run locally or be served as a webpage through public web servers. Therefore, users do not need to configure any server and request extra permissions for using MedTator.

Usage scenarios

When a specific annotation tool is selected for a project, it usually would not be replaced during the annotation process to ensure consistent data quality and annotation experience. Therefore, the decision on the annotation tool should be made carefully based on project needs and tool features. We have been using MedTator in several annotation projects and we proposed a guideline of annotation best practice that aims to streamline the whole clinical document annotation process [19]. According to our experience and comments from community users, we think the features provided by MedTator can help to meet the needs in the following scenarios:

- **Restricted environment**. Due to privacy and data security concerns, if the annotation has to be conducted in a specific local machine or secured network with limited permissions of changing environment, MedTator can provide a lightweight annotation solution and address those concerns.
- **Prototype validation**. If researchers plan to implement an NLP system based on an annotated corpus, MedTator provides a quick start to validate the workflow of the prototype without extra costs.
- **Education**. If educators plan to teach students about clinical NLP, MedTator can be used as it does not require any user registration, installation, or server/client configuration. All the inputs are just plain text files that are easy to manage and distribute. And the tutorial and manuals are also provided for users to follow.

Limitations

While having the above advantages, MedTator still has some shortcomings: First, unlike Java and Python, there are fewer JavaScript libraries for scientific purposes, so advanced techniques such as machine learning models, knowledge graphs, and NLP toolkits, are not able to be integrated easily to provide more features such as active learning [6]. Secondly, due to the use of HTML5 APIs in the web browser, MedTator only supports specific modern web browsers. Lastly, the current user interface design is highly rated by our users, and it works well for annotating hundreds of documents, but a formal usability test and load test need to be conducted to evaluate the performance.

Conclusion

To facilitate the corpus development in clinical research, we developed a serverless web-based annotation tool, MedTator, to provide an intuitive and interactive user interface that focuses on the core annotation tasks. We summarized the lessons learned from designing, developing, and maintaining MedTator, which will be of value to the research community for future open-source tool development.

Acknowledgment

The authors thank Donna Ihrke and other annotators for their expertise and valuable feedback throughout the development, the members of the AMIA, OHNLP, and N3C community for their insights on the tool design and assistance on testing, and the anonymous reviewers for their valuable comments. This work was supported by the National Center for Advancing Translational Sciences of the National Institutes of Health under award number U01TR002062.

References

1. Wen A, Fu S, Moon S, El Wazir M, Rosenbaum A, Kaggal VC, et al. Desiderata for delivering NLP to accelerate healthcare AI advancement and a Mayo Clinic NLP-as-a-service implementation. Npj Digit Med. 2019 Dec 17;2(1):1–7.
2. Dreisbach C, Koleck TA, Bourne PE, Bakken S. A systematic review of natural language processing and text mining of symptoms from electronic patient-authored text data. Int J Med Inf. 2019 May;125:37.
3. Neves M, Ševa J. An extensive review of tools for manual annotation of documents. Brief Bioinform. 2021 Jan 1;22(1):146–63.
4. Stenetorp P, Pyysalo S, Topić G, Ohta T, Ananiadou S, Tsujii J. brat: a Web-based Tool for NLP-Assisted Text Annotation. In: Proceedings of the EACL 2012: Demonstrations. Avignon, France: ACL; 2012. p. 102–7.
5. Kwon D, Kim S, Wei CH, Leaman R, Lu Z. ezTag: tagging biomedical concepts via interactive learning. Nucleic Acids Res. 2018 Jul 2;46(W1):W523–9.
6. Klie JC, Bugert M, Boullosa B, Eckart de Castilho R, Gurevych I. The INCEpTION Platform: Machine-Assisted and Knowledge-Oriented Interactive Annotation. In: Proceedings of the 27th COLING : System Demonstrations. Santa Fe, New Mexico; 2018. p. 5–9.
7. Islamaj R, Kwon D, Kim S, Lu Z. TeamTat: a collaborative text annotation tool. Nucleic Acids Res. 2020 Jul 2;48(W1):W5–11.
8. Salgado D, Krallinger M, Depaule M, Drula E, Tendulkar AV, Leitner F, et al. MyMiner: a web application for computer-assisted biocuration and text annotation. Bioinformatics. 2012 Sep 1;28(17):2285–7.
9. Maxim Tkachenko, Mikhail Malyuk, Nikita Shevchenko, Andrey Holmanyuk, Nikolai Liubimov. Label Studio [Internet]. 2021 [cited 2022 Jan 31]. Available from: https://github.com/heartexlabs/label-studio
10. He H, Fu S, Wang L, Liu S, Wen A, Liu H. MedTator: a serverless annotation tool for corpus development. Bioinformatics. 2022 Jan 4;btab880.
11. Carlson LA, Jeffery MM, Fu S, He H, McCoy RG, Wang Y, et al. Characterizing Chronic Pain Episodes in Clinical Text at Two Health Care Systems: Comprehensive Annotation and Corpus Analysis. JMIR Med Inform. 2020 Nov 16;8(11):e18659.
12. Shen F, Wang L, Liu H. Phenotypic Analysis of Clinical Narratives Using Human Phenotype Ontology. Stud Health Technol Inform. 2017;245:581–5.
13. Peterson KJ, Jiang G, Liu H. A corpus-driven standardization framework for encoding clinical problems with HL7 FHIR. J Biomed Inform. 2020 Oct 1;110:103541.
14. Fu S, Leung LY, Wang Y, Raulli AO, Kallmes DF, Kinsman KA, et al. Natural Language Processing for the Identification of Silent Brain Infarcts From Neuroimaging Reports. JMIR Med Inform. 2019;7(2):e12109.
15. Comeau DC, Islamaj Doğan R, Ciccarese P, Cohen KB, Krallinger M, Leitner F, et al. BioC: a minimalist approach to interoperability for biomedical text processing. Database J Biol Databases Curation. 2013;2013:bat064.
16. Liu H, Bielinski SJ, Sohn S, Murphy S, Wagholikar KB, Jonnalagadda SR, et al. An Information Extraction Framework for Cohort Identification Using Electronic Health Records. AMIA Summits Transl Sci Proc. 2013 18;2013:149–53.
17. Castro P, Ishakian V, Muthusamy V, Slominski A. The rise of serverless computing. Commun ACM. 2019 Nov 21;62(12):44–54.
18. Hripcsak G, Rothschild AS. Agreement, the F-Measure, and Reliability in Information Retrieval. J Am Med Inform Assoc JAMIA. 2005;12(3):296–8.
19. Fu S, Leung LY, Raulli AO, Kallmes DF, Kinsman KA, Nelson KB, et al. Assessment of the impact of EHR heterogeneity for clinical research through a case study of silent brain infarction. BMC Med Inform Decis Mak. 2020 Mar 30;20(1):60.
20. Oviatt S. Human-centered design meets cognitive load theory: designing interfaces that help people think. In: Proceedings of the 14th ACM international conference on Multimedia. New York, NY, USA: Association for Computing Machinery; 2006. p. 871–80. (MM '06).
21. Aghajani E, Nagy C, Vega-Márquez OL, Linares-Vásquez M, Moreno L, Bavota G, et al. Software Documentation Issues Unveiled. In: 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). 2019. p. 1199–210.
22. Aghajani E, Nagy C, Linares-Vásquez M, Moreno L, Bavota G, Lanza M, et al. Software Documentation: The Practitioners' Perspective. In: 2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE). 2020. p. 590–601.