Integrating INCEpTION into larger annotation processes

Richard Eckart de Castilho and Jan-Christoph Klie and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP Lab)
Department of Computer Science and Hessian Center for AI (hessian.AI)
Technical University of Darmstadt
www.ukp.tu-darmstadt.de

Abstract

Annotation tools are increasingly only steps in a larger process into which they need to be integrated, for instance by calling out to web services for labeling support or importing documents from external sources. This requires certain capabilities that annotation tools need to support in order to keep up. Here, we define the respective requirements and how popular annotation tools support them. As a demonstration for how these can be implemented, we adapted INCEpTION, a semantic annotation platform offering intelligent assistance and knowledge management. For instance, support for a range of APIs has been added to INCEpTION through which it can be controlled and which allow it to interact with external services such as authorization services, crowdsourcing platforms, terminology services or machine learning services. Additionally, we introduce new capabilities that allow custom rendering of XML documents and even the ability to add new JavaScript-based editor plugins, thereby making INCEpTION usable in an even wider range of annotation tasks.

1 Introduction/Motivation

Annotated data is crucial for many branches of science and industry. It is used in supervised learning to train and evaluate machine learning models (Pustejovsky and Stubbs, 2013) and has been the catalyst as well as limiting factor for the deep learning revolution (Sun et al., 2017; Sambasivan et al., 2021). Large language models often require high-quality annotated data, be it for (instruction) fine-tuning or their evaluation (Chen et al., 2023; Zhang et al., 2023; Zhou et al., 2023).

As the processes producing and consuming annotations become more complex, annotation tools need to be able to act as a part in these larger processes. For instance, they need to be embedded in a crowdsourcing pipeline (Klie et al., 2023), integrate external knowledge bases (Bugert et al., 2021), or provide functionality to call machine learning models for annotation support (Schulz et al., 2019). They also need to be customizable in order to cope with the ever-demanding change in requirements. If the functionality for a task is not implemented yet, they need to be extensible so that these new features can be easily retrofitted.

We survey the required capabilities in five areas relevant to customization and integration into larger processes: annotator management, task design, process integration, machine learning services and external knowledge and discuss if and how popular annotation tools support them. In addition, we describe how INCEpTION has implemented them to serve as a role model for future implementations.

2 Related work

Over the years, many annotation tools have been developed that target different use cases and come with different capabilities. We discuss some older but popular as well as some more recently published text annotation tools (see Neves and Ševa (2019) for a more comprehensive overview). Most tools considered are free open source tools. LABEL STUDIO and POTATO are *freemium* tools that require a paid license for certain functionalities or use-cases. PRODIGY is commercial software.

Design choices tend to be made based on whether an annotation tool is mainly *instance-oriented* or *document-oriented*. Many recent tools are *instance-oriented* and focusing on high throughput (e.g., PRODIGY (Montani and Honnibal), LABEL STUDIO, GATE TEAMWARE 2 (Wilby et al., 2023), ALANNO (Jukić et al., 2023), LABEL SLEUTH (Shnarch et al., 2022), or POTATO (Pei et al., 2022)). They try to get annotators to label as many instances as possible in the shortest amount of time. Often, these tools only support labelling the entire instance. Some support span and relation annotation tasks, but tend to focus on very

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing
System Demonstrations, pages 110–121

	ALANNO	Doccano	GATE	Label Sleuth	Label Studio	POTATO	Prodigy	brat	MedTator	WebAnno	INCEpTION
Annotator management											
AM-1: Multi-user	✓	✓	✓	–	✓	✓	–	✓	–	✓	✓
AM-2: Workload mgmt.	dyn	stat	dyn	n/a	dyn*	dyn	n/a	stat	n/a	stat	dyn
AM-3: Reclaim abandoned	–	–	✓	–	–	–	–	–	–	–	✓
AM-4: Self-sign-up	L	IdP	L	n/a	L, IdP*	URL	n/a	–	n/a	–	URL, IdP
Task design											
TD-1: Customizable UI	tagset	tagset	templ.	tagset	templ.	templ.	templ.	schema	schema	schema	schema
TD-2: Document layout	–	–	–	–	✓	–	–	–	–	–	✓
Process integration											
PI-1: API	–	R	R	R	R	–	L	–	–	R	R
PI-2: Event notifications	–	–	–	–	✓	–	–	–	–	✓	✓
ML services											
ML-1: ML support	BI	P	P	BI	P, R	BI	P, L	P, R	P	P, BI	P, BI, R
ML-2: Active learning	✓	–	–	✓	✓*	✓	✓	–	–	–	✓
Knowledge bases											
KB-1: RDF/SPARQL	–	–	–	–	–	–	–	–	–	–	✓
KB-2: Generic lookup	–	–	–	–	–	–	–	✓	–	–	✓

Table 1: Integrability requirements and their support in selected annotation tools. Some features (*) are only available paid versions. **BI** - built-in; **L -** local, **R** - remote, **P** - pre-annotated, **IdP** - Identity Provider.

short documents, e.g. a single sentence a single turn in a conversation. Their user interface (UI) is streamlined to support this goal e.g. by showing only the instance to be annotated with little to no context. *Document-oriented* tools (brat (Stenetorp et al., 2012), DOCCANO (Nakayama et al., 2018), MEDTATOR (He et al., 2022), WEBANNO (Yimam et al., 2013), INCEpTION) on the other hand show an entire document to the annotator at a time. Here, a document usually consists of a longer text (e.g. an essay, article, speech, conversation, etc.). This allows the annotation of spans and relations in their intended context. While document-oriented tools impose a higher cognitive load on the annotator, context can be very important for areas where reading a statement in isolation can easily lead to misinterpretation. Areas prone to such problems include the analysis of misinformation, political speeches, or analysis of inconsistencies or incoherence in documents (cf. Chong et al. (2021)).

3 Requirements and Contributions

In the following subsections, we identify several requirements that annotation tools should meet in order to integrate well into a larger process and how INCEpTION meets these requirements. Table 1 compares the capabilities of INCEpTION to those of other annotation tools. A detailed discussion of this comparison can be found in the appendix. Our last publication on INCEpTION (Klie et al., 2020) was written around the time of INCEpTION 0.16.1 (Jun 2020). Most of the features touched upon in the present paper have been developed or significantly improved in the versions 0.17.0 (Oct 2020) to 34.0 (Oct 2024). In the contributions, we mention the approximate version introducing a particular feature, e.g. *AM-2* $\approx$ *v0.17.0* indicates that the features supporting the requirement *AM-2* was introduced around INCEpTION 0.17.0. A few features have always existed in INCEpTION and are mentioned just for the sake of completeness. These are noted as e.g. *AM-1 always*.

3.1 Annotator management (AM)

A central component of any annotation project is the team of annotators. In *traditional* annotation projects, teams tend to be small and all annotators end up annotating all the texts (Chamberlain et al., 2013). However, if an annotation project contains a larger number of documents, also a larger number of annotators is called for. Thus, annotation tools need to offer functionalities for dealing with a large and potentially dynamic group of annotators.

Requirements (AM-1) MULTI-USER [YES, NO] – Annotation tools should offer multi-user support and the ability to manage the annotation team. Single-user tools might still be integrable into a larger process where multi-user support is

provided through external systems, e.g. provisioning different instances of the tool to different users.

(AM-2) WORKLOAD MANAGEMENT [STATIC, DYNAMIC] – If the annotation team is known in advance and changes seldom (if ever) during the course of the project, project managers can manually distribute the workload (e.g. the texts to be annotated) to the team members. But if the team is dynamic, annotators frequently join or leave the project (Snow et al., 2008), or the productivity differs significantly among the team members, automatic methods of work distribution are necessary.

(AM-3) RECLAIM ABANDONED [YES, NO] – It can be necessary to detect when when annotators abandon a project so that unfinished work can be reclaimed and reassigned to other annotators. This is particularly important if workload distribution is based on larger units, e.g. batches of multiple instances or long documents.

(AM-4) SELF-SIGN-UP [LOCAL ACCOUNT, IDP, INVITE-URL] – A suitable sign-up and sign-in mechanism is required when team members should be able to join a project at any time. This can be useful e.g. in crowdsourcing or citizen science projects or if a project can otherwise call on a large pool of potential annotators. Self-sign-up can create a local account or operate in conjunction with an external identity provider (IdP). An invite URL that grants access to a particular annotation project can facilitate the process.

Contribution INCEpTION is a multi-user annotation tool (AM-1 always) that supports dynamic workload management (AM-2 $\approx$ v0.17.0). Its URL-based self-sign-up (AM-4 $\approx$ v0.18.0) can be used either with anonymous accounts or with permanent accounts in combination with an OAuth2 ($\approx$ v0.25.0) or SAML-compliant IdP ($\approx$ v0.27.0). A notable difference to other tools is the handling of abandoned work though (AM-3 $\approx$ v0.20.0).

Dynamic workload management in a document-oriented tool like INCEpTION needs to meet slightly different goals than in instance-oriented tools because an annotator usually spends a longer time per document. After an annotator as been offered a document, that annotator needs to be allowed some time to work on it. If the document has been offered to the maximum number of annotators allowed per document, it may not be offered again until one of these annotators has aborted or abandoned their work. Also, there is the possibility that a document is abandoned after a non-trivial amount of work went into it – or that the user simply forgets marking the document as finished. In such cases, it may be useful to reclaim the document and assign it to another user. However, it may also be sensible to not completely discard the annotations that may already have been created in the document.

In addition to setting a limit of annotators per document and configuring an optional timeout before a document is considered to be abandoned, INCEpTION offers three options of dealing with abandoned documents: *discard* the data from the annotator who abandoned the document; *lock* the document for the annotator who abandoned it so the annotator can no longer edit it (if the annotator re-joins the project, the annotator will be assigned a new document); mark the document as *finished* for the annotator even though the annotations in the document may be incomplete. With *discard* and *lock*, the abandoned document will not count against the annotator-per-document limit and will be reassigned to new annotators. With *lock* and *finished*, the work already invested by the annotator into the document will be preserved.

3.2 Task design (TD)

There are almost infinite possibilities how to design annotation tasks and what to annotate. For example, annotation tools may focus on specific tasks (e.g. entity linking) or classes of tasks (e.g. spans/relations or whole documents).

Requirements (TD-1) CUSTOMIZABLE ANNOTATION UI [TAGSET, SCHEMA, TEMPLATE] The annotation UI determines the efficiency of the annotators to a great degree. The better the UI is suited to the task at hand, the faster the annotators can work and the less cognitive load they have to bear. The structure of annotations can range from just allowing a single label to complex annotation schemes with multiple attributes. Specialized widgets should be offered depending on the type of attribute, e.g. to rank an instance on a Likert-scale, link it to a knowledge base, single- or multiple choice labels, etc. A flexible arrangement of widgets using a templating mechanism can further optimize annotation efficiency.

(TD-2) DOCUMENT LAYOUT [YES, NO] The documents to be annotated can come in many different formats from plain text files, PDF files, various XML dialects, up to complex pre-annotated files. The level to which an annotation can be customized and extended in these areas determines the range of

annotation tasks it can be used for. Web browsers can display formatted HTML documents. The ability to annotate formatted documents as opposed to plain text documents is important for many users.

Contribution When it comes to the customizability of the annotation UI, INCEpTION stays close to other document-oriented annotation tools. It supports a flexible schema definition with a range of different attribute types, each coming with specialized inputs (TD-1 ALWAYS). Some widgets are to a degree configurable (e.g. the size of an input field can be changed to accommodate large comments, choosing between a dropdown or a radio-box presentation for single-choice string labels, etc.). Recent additions to the available attribute types include multi-value string attributes (≈ v23.0) and multi-value concept attributes (≈ v24.0). Also, single-value string attributes with tagsets can be displayed as a radio group to allow single-click label selections (≈ v20.0).

INCEpTION offers two unique capabilities: the ability to switch between different views of a document (always) and the ability to add support for new XML-based formats through a plugin mechanism (TD-2 ≈ v30.0).

For example, if a PDF or HTML document is imported, the user can freely switch between layout-oriented annotation mode using PDF.js (2022) and a content-oriented annotation mode, e.g. using the brat-based one-sentence-per-line mode. The PDF support was updated to support a more robust anchoring of the annotations to the text (≈ v24.0).

There are certain annotation tasks that require particular UI arrangements, e.g. cross-document linking or word-alignment tasks. To support such cases, a plugin mechanism is introduced that allows implementing custom editors in JavaScript. The mechanism consists of a JavaScript API (≈ v23.0) that handles the communication between the editor running in the annotator's browser and the INCEpTION backend, a plugin descriptor, packaging specification, and an optional mechanism for styling and filtering XML to support documents in DocX, TEI, TMX, JATS or similar formats.

The JavaScript API allows the editor to send commands to the server, e.g. *create span annotation*, *delete annotation*, or *select annotation* or to request the annotated document from the server for rendering. It also allows the server to push updates to the editor. Annotated documents can be complex and contain a large amount of information. Instead of transferring the entire information, INCEpTION pre-renders the annotated document on the server side into a condensed *visual representation* containing only limited information such as span offsets, relation endpoints, annotation colors and labels. Rendering this visual representation in the browser is simpler and more efficient than working with the full server-side representation. To further reduce the size of the data sent to the browser, the editor request only data relevant to the part of the document that is visible in the browser. Additionally, when possible a differential update mode relying on JSONDiff/JSONPatch (Bryan and Nottingham, 2013) is used to send only minimal updates to the browser. The JavaScript API does not directly expose the wire format sent by the server but rather decodes the format into a JavaScript object model. This decoupling of the wire format from the requirements of convenient access to the data via the API provides further opportunity for choosing a compact wire representation.

To demonstrate its viability, we have integrated several editor front-ends using the plugin mechanism based on Annotator JS (2015) (INCEpTION AnnotatorJS plugin, 2023), Apache Annotator (2021) (INCEpTION Apache Annotator plugin, 2023), RecogitoJS (2023) (INCEpTION RecogitonJS plugin, 2023) and DOCCANO (INCEpTION Doccano plugin, 2023). The editor based on Apache Annotator is also now (≈ v29.0) built into INCEpTION and used as the default editor for HTML/XML-based files. Also, the updated PDF support makes use of the JavaScript API. The brat-based editors have been upgraded to use the JavaScript API to send commands to the server, but are still using their own document serialization format to receive annotation data from the back-end.

The actual document is usually not rendered by the editor plugin itself but rather provided directly by the back-end as text or XML/XHTML – depending on what the plugin requests. Browsers can not only render HTML documents, but they can actually render any XML documents and style them using cascading style sheets (CSS). This creates the opportunity for a generic XML document importer which analyzes and preserves the XML structure of the document during import. This structure can then be loaded into the browser. The plugin can then provide a CSS to visually style this XML structure. Additionally, the plugin has to provide a content policy file. This policy define which elements and attributes may safely be sent to the browser.

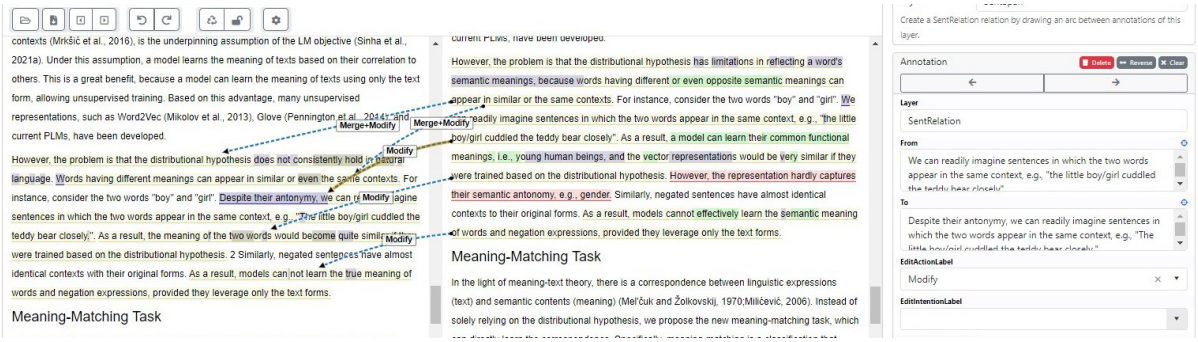

Figure 1: INCEpTION INTERTEXT plugin (2024) rendering a formatted XML document in a side-by-side view.

Anything not permitted by the policy is filtered out on the server side. In particular, such a policy should remove script tags or other potentially harmful content from the XML data. It can also be used to improve loading times by reducing the data being sent to the browser.

To avoid having to implement a new editor plugin for every XML dialect, there is also the option to define a custom XML format plugin (≈ v30.0) which includes only the CSS and policy file and which can be used in conjunction with any XML/HTML-based editor such as Apache Annotator or RecogitoJS. The generic XML document importer (≈ v23.0) in conjunction with the custom XML formats and/or the annotation editor API enable the support of many XML formats without having to change the INCEpTION code.

To demonstrate the viability of displaying formatted XML files, we have implemented partial support for the TEI P5 XML format to allow rendering plays from the Drama Corpora Project (Fischer et al., 2019) (≈ v31.0). Also, the INCEpTION custom XML format examples (2024) repository contains example custom format definitions for the *Timed Text Markup Language 2 (TTML)* and the *Translation Memory Exchange 1.4 format (TMX)*.

The INCEpTION INTERTEXT plugin (2024) uses the mechanisms described above to support a pairwise cross-document linking use-case (Ruan et al., 2024). For this use-case, we have defined a simple XML format that contains a *view-left* and *view-right* section which are display side-by-side in the browser using CSS styling (Fig. 1). Each of the two documents to be linked go into one of these sections. The RECOGITOJS-based editor plugin was then slightly modified to track the two views separately and to dynamically load annotations as the user scrolls.

3.3 Process integration (PI)

Annotation tools are used to create annotated data or to improve it, e.g. by correcting mistakes. Traditionally, there was a process of first compiling a corpus and then annotating it. The annotated gold standard corpus was then a final product to be published and shared. Annotation is increasingly becoming a step in a larger process where new data is automatically acquired, possibly pre-annotated before being rolled out to a dynamic group of annotators. Finally, the annotated data is fed back into a process to improve a model which is then used for pre-annotation in the next iteration.

Requirements (PI-1) API [NONE, LOCAL, REMOTE] To be integrable into such processes, annotation tools need to offer APIs through which data can be provisioned for annotation, the annotation process can be monitored, and the annotated data be retrieved again for further processing. The integration into a larger process works best if the tool offers an API for project management. Such an API should allow at least creating a project, deploying data to be annotated, monitoring the progress of the annotations, exporting the annotated data and finally deleting the project again.

(PI-2) EVENT-BASED NOTIFICATION [YES, NO] While an external process could poll the annotation tool for state changes, a event-based notification mechanism can more efficiently trigger external actions when specific events occur. Such events could include an annotator completing a document, or all documents in a batch being completed.

Contribution INCEpTION provides an AERO-compatible[1] remote API for project management needs (PI-1 ALWAYS). While AERO works well for setting up and wrapping up projects, it is lack-

[1] https://openminted.github.io/releases/aero-spec/1.0.0/omtd-aero/

ing functionality for dynamically updating certain aspects of running annotation projects such as managing user permissions. We therefore added new endpoints in INCEpTION (≈ v24.0) for listing, adding, and removing user permissions. Additionally, a new endpoint for setting the state of a document for a given user was added (≈ v0.19.0). This can be used for example to remotely re-open a document that an annotators has marked as finished or to lock certain documents for specific annotators. All new endpoints conform to conventions of the AERO API design. For integration into enterprise environments, we added support for OAuth authentication to the remote API (≈ v26.0).

Webhooks to trigger external processes when the state of individual users, documents or the entire project changes are supported as well (PI-2 always). Webhooks have been extended (≈ v24.0) to support a limited retry in case the recipient of the notification is temporarily unreachable, to allow header-based authentication to the recipient, as well as to include a timestamp and the user who triggered the event.

3.4 Machine learning services (ML)

There are several ways of using machine learning (ML) to support the annotation process. A common approach to improve annotation speed is using already pre-annotated data and just let annotators correct them (Fort and Sagot, 2010). There, potential annotations are shown inline in the annotation editor which can be accepted or rejected by the annotators. Tools that support loading pre-annotated data typically assume that any data not explicitly rejected by the user is correct. Instance-oriented tools usually require the user to accept the instance, but do not force the user to explicitly accept each span, relation or attribute value. Because in document-oriented tools there is typically large quantity of annotations per document, it can be easy to miss a wrong one. Thus, a mechanism that requires the annotator to verify each automatically generated annotation explicitly can be beneficial. One approach to achieve this are dynamic label suggestions in the form of recommenders (Schulz et al., 2019).

Requirements (ML-1) ML SUPPORT [PRE-ANNOTATION, BUILT-IN, LOCAL, REMOTE] Machine Learning (ML) is currently one of the fastest moving areas of science. Relying only on built-in ML capabilities limits the scope of an annotation tool. Being able to import pre-annotated data or to call out to a local library or a remote ML service gives users the opportunity to connect the latest and best available ML capabilities to a tool.

(ML-2) ACTIVE LEARNING [YES, NO] Active Learning (Settles, 2012) (AL) can be used to reduce the amount of training data needed to reach a certain performance level. It requires a tight integration of ML services with the annotation tool as the ML model determines the order in which instances are presented to the user for annotation and as the model is frequently updated or re-trained as part of the active-learning process.

Contribution INCEpTION follows the *predict/fit* paradigm for its ML service integration (ML-1 always). It comes with several built-in ML services as well as the ability to invoke remote ML services using a simple HTTP-based protocol. The INCEpTION external recommender (2024) repository contains a Python-based ML server implementation and provides examples based on scikit learn (Pedregosa et al., 2011), spaCy (Honnibal et al., 2020), SentenceTransformers (Reimers and Gurevych, 2019) and many more. In terms of interaction, INCEpTION opts for the recommender model where the annotator has to explicitly accept or reject annotation suggestions. If the ML services provide a score along with the labels, INCEpTION can apply an AL mode (ML-2 ≈ v0.13.0) that uses uncertainty sampling to guide the annotator through the annotation suggestions.

While the *predict* function typically generates only labels and potentially scores, we found it useful to also allow associating an explanatory description to each annotation which is presented to the user when the mouse hovers over the suggestion.

The ML service can set a flag on an auto-generated annotation to signal that it should be accepted immediately without user interaction (≈ v28.0). This can be used to avoid imposing work on the human annotator to explicitly verify annotation suggestions that have a very high probability of being correct. It also enables new usage scenarios which dynamically or conditionally create place-holder annotations that highlight spans an annotator should label, but without assigning the labels yet. This removes the need from the annotator to create the annotations themselves, so they can then focus on label assignment.

3.5 Knowledge bases (KB)

Some annotation tasks involve disambiguating concept mentions against a very large terminology or knowledge base. Such annotation tasks typically involve entity linking, concept disambiguation, or normalization (e.g. Ehrmann et al. (2020)).

Requirements (KB-1) SPARQL/RDF SUPPORT [YES, NO] The dominant data representation standard in this area is RDF and SPARQL as the query protocol. And even the different SPARQL server implementations each have their own proprietary full-text-search commands which are essential for efficiently querying large databases. Interoperability with SPARQL services gives an annotation tool access to many relevant resources.

(KB-2) GENERIC LOOKUP PROTOCOL [YES, NO]) There are other data formats such as OBO (Open Biomedical Ontologies) or TBX (TermBase eXchange) and other query standards such as the FHIR (Saripalle et al., 2020) terminology services API. Thus, tools that support a simpler protocol can offer a better integrability as users can adapt it for any kind of server back-end they may be using.

Contribution By supporting RDF and SPARQL, INCEpTION is able to use many terminology and knowledge-base resources (KB-1 always). Recently, in particular the support for large knowledge bases such as SNOMED-CT (SNOMED International, 2024) or the Human Phenotype Ontology (Robinson et al., 2008) has been improved by allowing to directly import files in OWL functional syntax and OBO formats (≈ v31.1), supporting synonyms (≈ v21.0), out-of-order matching of search terms to concept labels (≈ v33.0), as well as various performance improvements.

However, converting terminologies to RDF and querying them using SPARQL can still incur a significant overhead. Thus, we introduce support for a custom HTTP-based lightweight *lookup* protocol (LLP) into INCEpTION (KB-2 ≈ v27.0) to facilitate the integration with other resources. While RDF and SPARQL-support aims at supporting standard formats and protocols to be interoperable with exisiting technology, the LLP aims at facilitating the implementation of custom service proxies to be able to access arbitrary backends. It would be straightforward to index a terminology in an APACHE SOLR index, a FHIR server or even an SQL database and build a small LLP proxy service to access this index.

An LLP-compliant service responds to a GET request in one of two modes: *query* or *lookup*. The *query* mode is enabled by the presence of the query parameter *q* which contains the string entered by the user that is to be auto-completed. The context of the query may be included in the *qc* parameter. Typically, this is the text covered by the (span) annotation that is linked to the external resource. This allows generating auto-completion suggestions based on the annotated text even if the user did not type anything yet. Consider an entity-linking task where the user wants to disambiguate the name of a drug using a drug database. The user can simply annotate the drug name, press space in the label editor to trigger an auto-completion and the LLP service can return potential matches of the drug name from the database. When the user selects a match, the identifier of that match is stored in an annotation attribute. The *lookup* mode, is triggered by the presence of the *id* parameter. This is used during rendering to resolve the identifiers to their label and optional description. The INCEpTION lookup service examples (2024) repository offers example lookup service implementations supporting the EMBL-EBI Ontology lookup service and the Wikidata REST API.

4 Conclusion

We have discussed recent developments in the the free and open source annotation tool INCEpTION which allows it to be integrated as a step into larger processes and which allow it to be customized using format and editor plugins so the tool can be used with a wider range of document types and for a wider range of annotation tasks. We have compared the tool to the state-of-the-art and see that based on the capabilities discussed here, INCEpTION is one of the most versatile tools in its peer group. That said, we see further opportunities for innovative annotation user interfaces (e.g. to better accommodate annotation tasks related to large language models) as well as in for supporting a wider range of document types.

Acknowledgements

This project was supported by the German Research Foundation (DFG, EC 503/1-1 and GU 798/21-1, INCEpTION), by the Federal Ministry of Education and Research (BMBF, 01ZZ2314H, GeMTeX), and by the European Union (ERC, InterText, 101054961).

References

Annotator JS. 2015. Homepage. http://annotatorjs.org. Version 1.2.0.

Apache Annotator. 2021. Homepage. https://annotator.apache.org. Version 0.2.0.

Paul C. Bryan and Mark Nottingham. 2013. JavaScript Object Notation (JSON) Patch. RFC 6902.

Michael Bugert, Nils Reimers, and Iryna Gurevych. 2021. Generalizing Cross-Document Event Coreference Resolution Across Multiple Corpora. *Computational Linguistics*, 47(3):575–614.

Jon Chamberlain, Karën Fort, Udo Kruschwitz, Mathieu Lafourcade, and Massimo Poesio. 2013. Using Games to Create Language Resources: Successes and Limitations of the Approach. In *The People's Web Meets NLP*, pages 3–44. Springer, Berlin, Heidelberg.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2023. AlpaGasus: Training A Better Alpaca with Fewer Data. In *arXiv*.

David Chong, Erl Lee, Matthew Fan, Pavan Holur, Shadi Shahsavari, Timothy Tangherlini, and Vwani Roychowdhury. 2021. A real-time platform for contextualized conspiracy theory analysis. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 118–127.

Maud Ehrmann, Matteo Romanello, Simon Clematide, Phillip Benjamin Ströbel, and Raphaël Barman. 2020. Language resources for historical newspapers: The impresso collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 958–968, Marseille, France. European Language Resources Association.

Frank Fischer, Ingo Börner, Mathias Göbel, Angelika Hechtl, Christopher Kittel, Carsten Milling, and Peer Trilcke. 2019. Programmable Corpora: Introducing DraCor, an Infrastructure for the Research on European Drama. In *Proceedings of DH2019: "Complexities", Utrecht, July 912, 2019*. Utrecht University.

Karën Fort and Benoît Sagot. 2010. Influence of preannotation on POS-Tagged corpus development. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 56–63, Uppsala, Sweden. Association for Computational Linguistics.

Sian Gooding, Lucas Werner, and Victor Cărbune. 2023. A Study on Annotation Interfaces for Summary Comparison. In *Proceedings of the 17th Linguistic Annotation Workshop (LAW-XVII)*, pages 179–187, Toronto, Canada. Association for Computational Linguistics.

Huan He, Sunyang Fu, Liwei Wang, Andrew Wen, Sijia Liu, Sungrim Moon, Kurt Miller, and Hongfang Liu. 2022. Towards User-centered Corpus Development: Lessons Learnt from Designing and Developing MedTator. *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2022:532–541.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

INCEpTION AnnotatorJS plugin. 2023. GitHub Repository. https://github.com/inception-project/inception-annotatorjs-editor-plugin. Accessed: 2024-10-15.

INCEpTION Apache Annotator plugin. 2023. GitHub Repository. https://github.com/inception-project/inception-apache-annotator-editor-plugin. Accessed: 2024-10-15.

INCEpTION custom XML format examples. 2024. GitHub Repository. https://github.com/inception-project/inception-xml-formats-examples. Accessed: 2024-10-15.

INCEpTION Doccano plugin. 2023. GitHub Repository. https://github.com/inception-project/inception-doccano-sequence-editor-plugin. Accessed: 2024-10-15.

INCEpTION external recommender. 2024. GitHub Repository. https://github.com/inception-project/inception-external-recommender. Accessed: 2024-10-15.

INCEpTION INTERTEXT plugin. 2024. GitHub Repository. https://github.com/inception-project/inception-intertext-editor-plugin. Accessed: 2024-10-15.

INCEpTION lookup service examples. 2024. GitHub Repository. https://github.com/inception-project/inception-lookup-service-example. Accessed: 2024-10-15.

INCEpTION RecogitonJS plugin. 2023. GitHub Repository. https://github.com/inception-project/inception-recogito-editor-plugin. Accessed: 2024-10-15.

Josip Jukić, Fran Jelenić, Miroslav Bićanić, and Jan Snajder. 2023. ALANNO: An active learning annotation system for mortals. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 228–235, Dubrovnik, Croatia. Association for Computational Linguistics.

Jan-Christoph Klie, Richard Eckart de Castilho, and Iryna Gurevych. 2020. From Zero to Hero: Human-In-The-Loop Entity Linking in Low Resource Domains. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6982–6993, Online.

Jan-Christoph Klie, Ji-Ung Lee, Kevin Stowe, Gözde Şahin, Nafise Sadat Moosavi, Luke Bates, Dominic Petrak, Richard Eckart De Castilho, and Iryna Gurevych. 2023. Lessons Learned from a Citizen Science Project for Natural Language Processing. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3594–3608, Dubrovnik, Croatia. Association for Computational Linguistics.

Ines Montani and Matthew Honnibal. Prodigy: A modern and scriptable annotation tool for creating training data for machine learning models.

Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. 2018. doccano: Text annotation tool for human. Software available from https://github.com/doccano/doccano.

Mariana Neves and Jurica Ševa. 2019. An extensive review of tools for manual annotation of documents. *Briefings in Bioinformatics*, 22(1):146–163.

PDF.js. 2022. GitHub Repository. `https://github.com/mozilla/pdf.js`. Version 2.14.305.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jiaxin Pei, Aparna Ananthasubramaniam, Xingyao Wang, Naitian Zhou, Apostolos Dedeloudis, Jackson Sargent, and David Jurgens. 2022. POTATO: The portable text annotation tool. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 327–337, Abu Dhabi, UAE. Association for Computational Linguistics.

J. Pustejovsky and Amber Stubbs. 2013. *Natural Language Annotation for Machine Learning*. O'Reilly Media, Sebastopol, CA.

RecogitoJS. 2023. GitHub Repository. `https://github.com/recogito/recogito-js`. Version 1.2.8.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Peter N Robinson, Sebastian Köhler, Sebastian Bauer, Dominik Seelow, Denise Horn, and Stefan Mundlos. 2008. The human phenotype ontology: a tool for annotating and analyzing human hereditary disease. *Am J Hum Genet*, 83(5):610–615.

Qian Ruan, Ilia Kuznetsov, and Iryna Gurevych. 2024. Re3: A holistic framework and dataset for modeling collaborative document revision. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4635–4655, Bangkok, Thailand. Association for Computational Linguistics.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Kumar Paritosh, and Lora Mois Aroyo. 2021. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In *SIGCHI*, pages 1–21.

Rishi Saripalle, Mehdi Sookhak, and Mahboobeh Haghparast. 2020. An interoperable umls terminology service using fhir. *Future Internet*, 12(11).

Claudia Schulz, Christian M. Meyer, Jan Kiesewetter, Michael Sailer, Elisabeth Bauer, Martin R. Fischer, Frank Fischer, and Iryna Gurevych. 2019. Analysis of Automatic Annotation Suggestions for Hard Discourse-Level Tasks in Expert Domains. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Burr Settles. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Springer International Publishing, Cham.

Eyal Shnarch, Alon Halfon, Ariel Gera, Marina Danilevsky, Yannis Katsis, Leshem Choshen, Martin Santillan Cooper, Dina Epelboim, Zheng Zhang, and Dakuo Wang. 2022. Label sleuth: From unlabeled text to a classifier in a few hours. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 159–168, Abu Dhabi, UAE. Association for Computational Linguistics.

SNOMED International. 2024. SNOMED CT: Systematized Nomenclature of Medicine – Clinical Terms. `https://www.snomed.org/get-snomed`. Accessed: 2024-10-15.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: A web-based tool for NLP-Assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 843–852, Venice, Italy.

David Wilby, Twin Karmakharm, Ian Roberts, Xingyi Song, and Kalina Bontcheva. 2023. GATE teamware 2: An open-source tool for collaborative document classification annotation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 145–151, Dubrovnik, Croatia. Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A flexible, web-based and visually supported system for distributed annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria. Association for Computational Linguistics.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023. Instruction Tuning for Large Language Models: A Survey. *Preprint*, arxiv:2308.10792.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. LIMA: Less Is More for Alignment. *Preprint*, arxiv:2305.11206.

A Detailed comparison to state of the art tools

This appendix provides background information to the comparison presented in Table 1.

Annotator management Automatically distributing work across the available annotators is particularly relevant when there are many units of work to be distributed. In instance-oriented tools, every instance is a unit of work – typically very small one and there is a large number of them (e.g. several thousand). In document-oriented tools, the unit of work is typically larger and there are fewer of them. This is likely the reason that we find the most advanced annotator management features in the instance-oriented tools. For example, a recent update to PRODIGY (v1.12) introduced a programmable mechanism for routing work to annotators. However, that functionality seems only to be fully exploitable with the PRODIGY TEAMS offering that is unreleased at the time of writing, so we consider PRODIGY to be a single-user tool for the moment. POTATO and GATE TEAMWARE 2, LABEL STUDIO (paid) and ALANNO all offer configurable dynamic workload distribution mechanisms, typically allowing to set a target number of labels required for a given instance. GATE TEAMWARE 2 allows reclaiming instances abandoned by an annotator before assigning a label and distributing them to other annotators. DOCANNO has no workload distribution mechanism and expects all annotators to annotate all instances.

Dynamic workload management is most effective when paired with a self-sign-up mechanism or the ability to use a external identity provider (IdP) via OAuth or SAML protocols. POTATO offers a self-sign-up mechanism based on submitting an ID token via a special URL. LABEL STUDIO (paid) and DOCCANO offer IdP support e.g. via OAuth.

However, document-oriented tools mostly lack advanced workload management features. Neither BRAT nor MEDTATOR offer any workload management. WEBANNO allows the project manager to manually assign annotators to specific documents, but it is tedious and not suitable for scenarios where the composition of the annotation team is not known in advance or regularly subject to change.

Task design A highly customizable arrangement of the UI elements is mainly interesting for instance-oriented annotation tools in order to create a layout that minimizes cognitive load and maximizes annotation efficiency (He et al., 2022; Gooding et al., 2023). LABEL STUDIO, POTATO, PRODIGY and GATE TEAMWARE 2 are all relying on a templating mechanism to customize the layout of the annotation UI, typically intermixing predefined input elements with custom HTML code. DOCCANO offers different UIs for different kinds of pre-defined tasks, allows for custom tagsets, but is not flexibly configurable. LABEL SLEUTH and ALANNO allow for configurable tagsets, but no further customization of the annotation UI.

For document-oriented tools, usually, most of the screen is occupied by the document view, so there is less opportunity for custom arrangements. The flexibility of these tools tends to lie in the way the annotation schema is defined while leaving the UI layout to the tool. For example, WEBANNO, BRAT and INCEpTION offer a range of different attribute types of which one or more can be added to each annotation (string, number, rating, boolean, etc.); each coming with specialized inputs. These inputs are displayed to the user when editing an annotation, but their arrangement is not freely definable. MEDTATOR is least flexible in this area, allowing only for single-value or multi-value string attributes.

Support for annotating formatted text is scarce. Among the tools considered here, only LABEL STUDIO offers an input element that can display formatted text and allows creating span and relation annotations. It is also the only tool that supports displaying PDF documents, but only for document-classification tasks. Creating span and relation annotations inside the PDF are not supported.

Process integration While offering an API[2] is quite common for annotation tools today, there are still tools being published without one. ALANNO, POTATO, MEDTATOR and BRAT do all not offer an API. PRODIGY is essentially a programming library, so if offers a rich API. However, this API is not remotely accessible out-of-the-box. LABEL STUDIO, LABEL SLEUTH, GATE TEAMWARE 2 facilitating their integration into a larger process consisting of multiple interacting services.

The AERO remote API specification defines endpoints for remotely managing annotation projects,

[2]Note that some tools advertise the API used by their respective frontend layers as general purpose APIs. Frontend APIs are not management APIs and trying to coerce both use-cases into the same API is likely to create maintainability issues in the long run.

e.g. to create projects, import documents, monitor the progress of annotation and export the results. While annotation tools mostly implement proprietary APIs, the AERO specification was designed to be implementable by multiple tools, one of which is WEBANNO.

Event-based notifications allowing other services to react to state changes in the annotation tool are offered by LABEL STUDIO (paid) and WEBANNO.

Machine learning services GATE TEAMWARE 2 and MEDTATOR allow only importing and editing pre-annotated data. The document-oriented WEBANNO has a dedicated *correction* mode which requires the annotator to explicitly verify and merge each annotation from the pre-annotated document into the final document.

DOCCANO, BRAT and LABEL STUDIO support calling out to external ML services to annotate documents using generic HTTP-based protocols – the annotations can then be corrected by the annotator. WEBANNO offers only a single built-in ML algorithm with limited ability to customize its configuration. ALANNO comes with a range of built-in ML algorithms and automatically chooses the most applicable without the need or possibility for configuration. LABEL SLEUTH follows a similar approach but allows the configuration of model policies to decide which model is used for the next batch. PRODIGY defers to locally calling the spaCy library (Honnibal et al., 2020) from the same vendor for its ML backends.

The APIs to interact with ML services are very similar across tools. There is one *predict* method/endpoint which gets provided with data and returns data with annotations often in the same format. A second *fit* method/endpoint maybe be available if the tool also supports training models.

ALANNO, POTATO, LABEL STUDIO (paid), LABEL SLEUTH, and PRODIGY all offer Active Learning to efficiently source labels from the human annotator to improve the training efficiency of the model.

External knowledge Most annotation tools only offer limited support for controlled vocabularies in the form of tagsets – these were covered under *Task design*. Working with large terminologies or knowledge bases can put considerable cognitive load on the annotator, so it is not compatible with the throughput maximization objective of most instance-oriented annotation tools. The document-oriented tool BRAT is one of the few annotation tools that support linking annotations to knowledge bases using its *normalization* functionality. However, BRAT requires the manual generation of a local term index which is then used for auto-completion, so it is not really integrable with external services.

B Limitations

In this work, we discussed the importance of integrability for annotation tools based on a set of requirements and how state-of-the art tools implement them. While there are many annotation tools out there, they are too many to count or inspect. Therefore, we focused on a limited selection of some popular and some recent ones, trying to cover a reasonably representative portion of long term and recent trends. While we proceeded with utmost care when surveying the field, it is possible that we overlooked annotation tools that are highly relevant for this work.

When coming up with requirements concerning integrability, we derived them mainly from our own experience in developing annotation tools and integrating them with services and processes as well as our annotation tool survey. While mostly objective and generic, different annotation processes might need slightly different requirements and not 100% benefit from our suggestions. In particular, our perspective focuses more on document-oriented tools than on instance-oriented tools.

For each annotation tool, we read the papers, their documentation and at times had to look at their source code as well to assess how a tool works, if and how it supports a particular feature and how well it adhers in general to our set of requirements. Indeed, we were positively surprised how some of the tools we looked at have evolved in recent months. However, we did not actively use most of the tools. We still hope to have given a correct and fair assessments of their capabilities.