Antarlekhaka: A Comprehensive Tool for Multi-task Natural Language Annotation

Hrishikesh Terdalkar and **Arnab Bhattacharya**
{hrishirt, arnabb} @cse.iitk.ac.in
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur
Kanpur, India

Abstract

One of the primary obstacles in the advancement of Natural Language Processing (NLP) technologies for low-resource languages is the lack of annotated datasets for training and testing machine learning models. In this paper, we present *Antarlekhaka*, a tool for manual annotation of a comprehensive set of tasks relevant to NLP. The tool is Unicode-compatible, language-agnostic, Web-deployable and supports distributed annotation by multiple simultaneous annotators. The system sports user-friendly interfaces for 8 categories of annotation tasks. These, in turn, enable the annotation of a considerably larger set of NLP tasks. The task categories include two linguistic tasks not handled by any other tool, namely, sentence boundary detection and deciding canonical word order, which are important tasks for text that is in the form of poetry. We propose the idea of *sequential annotation* based on small text units, where an annotator performs several tasks related to a single text unit before proceeding to the next unit. The research applications of the proposed mode of multi-task annotation are also discussed. Antarlekhaka outperforms other annotation tools in objective evaluation. It has been also used for two real-life annotation tasks on two different languages, namely, Sanskrit and Bengali. The tool is available at https://github.com/Antarlekhaka/code.

1 Introduction and Motivation

Manual annotation plays an important role in natural language processing (NLP). It is particularly important in the context of low-resource languages for the creation of datasets.

There are a number of syntactic and semantic tasks in NLP which can utilize annotation by domain experts. Lemmatization, morphological analysis, parts-of-speech tagging, named entity recognition, dependency parsing, constituency parsing, co-reference resolution, sentiment detection, discourse analysis and so on are some examples of such common NLP tasks.

There's a need of considering historical context and respecting the perspectives of Indigenous language speaking communities when conducing NLP research involving these languages (Schwartz, 2022). Sanskrit, a classical language, has a large amount of text available digitally; however, it still suffers from poor performance in standard NLP tasks. Hence, manual annotation of text in Sanskrit is of prime necessity. Further, most of the classical Sanskrit literature is in poetry form following mostly free word order (Kulkarni et al., 2015), without any punctuation marks. Therefore, certain specialized tasks, such as sentence boundary detection and canonical word ordering, are needed for Sanskrit text processing

Consider an example[1] from a Sanskrit text, *Valmiki Ramayana*, (Dutt et al., 1891) shown in Fig. 1. The sentence boundaries are denoted using square brackets ([and]). The half-verse boundaries are marked by single forward slashes (/) and the verse boundaries by two forward slashes (//). It can be observed that the sentence boundaries do not coincide with the verse boundaries. In particular, there may be multiple sentences present in a single verse, or a sentence may extend across multiple verses. Further, the right side of the arrow shows that the canonical word order is different from the order in which words appear in the original text.

For such languages that either do not use punctuations or use them in a limited amount, sentence boundary detection is an important task. Additionally, in languages with relatively free word order, decision of a canonical word order is also relevant. These two tasks also play a vital role when dealing with the corpora in the form of poetry, making them potentially relevant for all languages.

Performing multiple annotation tasks on the

[1] Using IAST transliteration scheme: https://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration

[na rocate mama-api-etad-ārye]$_1$ [yad-rāghavo vanam / tyaktvā rājyaśriyaṃ gacchet]$_2$ [striyā vākyavaśaṃ gataḥ // 2 viparītaś ca vṛddhaś ca viṣayaiś ca pradharṣitaḥ / nṛpaḥ kim iva na brūyāc codyamānaḥ samanmathaḥ // 3]$_3$ [...]

→

[ārye etad mama api na rocate]$_1$ [yad rāghavo rājyaśriyaṃ tyaktvā vanam gacchet]$_2$ [viparītaḥ vṛddhaḥ ca viṣayaiḥ pradharṣitaḥ ca codyamānaḥ samanmathaḥ ca striyā vākyavaśaṃ gataḥ nṛpaḥ kim iva na brūyāt]$_3$ [...]

Figure 1: Sanskrit verses from Valmiki Ramayana. Original text appears on the left with sentence boundary markers added. The canonical word order is shown on the right.

same corpus is common, and the order of these tasks can be important due to their interdependence. Specifically, in cases[2] where sentence boundary detection is relevant, it must be performed before any other annotation task. For instance, determining the word order of a sentence requires finalizing the constituent words first. The same holds true for tasks such as dependency parsing, sentence classification, and discourse analysis.

In this paper, we describe *Antarlekhaka*[3], a tool for distributed annotation that provides user-friendly interfaces to facilitate the annotation process of various common NLP tasks in a straightforward and efficient way. We propose a sequential annotation model, where an annotator carries out multiple annotation tasks relevant for a small text unit, such as a verse, before proceeding to the next. The tool has full Unicode support and is designed to be language-agnostic, meaning it can be used with corpora from any language, making it highly versatile. The tool sports eight task-specific user-friendly annotation interfaces corresponding to eight general categories of NLP tasks: sentence boundary detection, canonical word ordering, free-form text annotation of tokens, token classification, token graph construction, token connection, sentence classification and sentence graph construction. The goal of the tool is to streamline the annotation process, making it easier and more efficient for annotators to complete multiple NLP tasks on the same corpus. Annotators can participate in the annotation without any programming knowledge. Additionally, the tool's easy setup and intuitive administrator interface make it accessible to administrators with minimal technical expertise.

2 Background

An annotation tool is crucial for the successful completion of any annotation task, and its success relies heavily on its usability for the annotators. Apart from this, the tool should be easily installable and should support web deployment for distributed annotation, allowing multiple annotators to work on the same task from different locations. The administration interface of the tool should also be intuitive and should provide convenient access to common administrative tasks such as corpus upload, ontology creation, and user access management. Additionally, often there is a need for multiple annotation tasks to be completed on the same corpus. A well-designed annotation tool should encompass these features to ensure a smooth, efficient, and accurate annotation process.

Numerous text annotation tools are available that target specific annotation tasks, such as WebAnno (Yimam et al., 2013), INCePTION (Klie et al., 2018), GATE Teamware (Bontcheva et al., 2013), FLAT (van Gompel, 2014), BRAT (Stenetorp et al., 2012), doccano (Nakayama et al., 2018) and more. However, each of these tools falls short in fulfilling all the requirements of an ideal annotation tool. For instance, WebAnno is rich in features but becomes complex to use and experiences performance issues as the number of lines displayed on the screen increases. Both WebAnno and BRAT lack full support for Firefox (Fort, 2016), an issue that was rectified in INCEpTION. GATE Teamware suffers from shortcomings such as inadequate support for relation and co-reference annotation (Herik et al., 2018), installation issues (Neves and Ševa, 2021) and complexity for average users (Yimam et al., 2013). FLAT uses a non-standard FoLiA XML data format and the system is not intuitive (Neves and Ševa, 2021). BRAT has not been actively[4] developed and exhibits issues such as slowness, limited scope for configuration and limitations regarding file formats (Yimam et al., 2013). The tool doccano, although simple to set up and intuitive, only supports labeling tasks. Sangrahaka (Terdalkar and Bhattacharya, 2021), while being easy to set up and use, focuses only on the annotation towards creation of knowledge graphs and lacks support towards general-purpose NLP annotation tasks.

Some annotation tools including INCEpTION

[2]For corpora without clear sentence boundaries, like languages with limited punctuation or poetic corpora

[3]*Antarlekhaka* is a Sanskrit word meaning 'annotator'.

[4]The latest version was published in 2012

Table 1: Comparison of NLP annotation tools based on primary features and supported tasks

	INCEpTION	GATE	BRAT	FLAT	doccano	Sangrahaka	**Antarlekhaka**
Distributed Annotation	✓	✓	✓	✓	✓	✓	✓
Easy Installation			✓	✓	✓	✓	✓
Sequential Annotation							✓
Querying Interface						✓	
Token Text Annotation	✓	✓	✓	✓			✓
Token Classification	✓	✓	✓	✓	✓		✓
Token Graph	✓	✓	✓	✓		✓	✓
Token Connection	✓	✓	✓	✓		✓	✓
Sentence Boundary	✓						✓
Word Order							✓
Sentence Classification	✓						✓
Sentence Graph							✓

use spans for marking most annotations, which a user by selecting and dragging mouse cursor over the corpus text. This method, while versatile, has a trade-off that the annotation process is slower and more tedious. Importantly, none of these tools address crucial tasks like canonical word ordering. Hence, there is a need for an annotation tool that is user-friendly, easy to install and deploy, and encompasses all the necessary tasks for NLP annotation.

Thus, for the general purpose multi-task annotation of NLP tasks, we present *Antarlekhaka*. The annotation is performed in a sequential manner on small units of text (e.g., verses in poetry). The application is language and corpus agnostic. The tool is able to process data in two different formats: the standard CoNLL-U[5] format and plain text format. Regular-expressions based tokenizer is applied when using the data in plain text format.

Tab. 1 shows a comparison of the prominent annotation tools. We also conduct an objective evaluation of *Antarlekhaka* using the scoring methodology proposed by (Neves and Ševa, 2021). We modify the criteria suitable to the domain of NLP annotation. Details of the evaluation are described in Sec. 4. It is important to note that while some of the existing tools, in theory, have the capability to support certain NLP tasks, they may not be designed with user-friendly interfaces.

3 Architecture

Antarlekhaka is a language-agnostic, multi-task, distributed annotation tool presented as a Web-deployable software. The tool makes use of several technologies, including *Python 3.8* , *Flask 2.2.2* , and *SQLite 3.38.3* for the backend, and *HTML5*, *JavaScript*, and *Bootstrap 4.6* for the frontend.

Flask web framework powers the backend of *Antarlekhaka* providing a robust and scalable infrastructure. A web framework is responsible for a range of backend tasks, including routing, templating, managing user sessions, connecting to databases and others. The recommended way to run the tool in a production environment is using a *Web Server Gateway Interface* (WSGI) HTTP server, such as *Gunicorn* , which can operate behind a reverse proxy server such as *NGINX* or *Apache HTTP Server* . However, any WSGI server, including the built-in server of Flask, can be utilized to run the application.

SQLite is used as the database management system to store and manage the data and metadata associated with the annotation tasks. An Object Relational Mapper (ORM) *SQLAlchemy*[6] is used to interact with the relational database. This allows the user to choose any supported dialect of traditional SQL, such as *SQLite*, *MySQL* , *PostgreSQL* , *Oracle* , *MS-SQL* , *Firebird* , *Sybase* and others[7].

The frontend of the tool, built using *HTML5*, *JavaScript*, and *Bootstrap*, provides user-friendly interfaces for annotators and administrators. The tool provides a feature-rich administrative interface to manage user access, corpus, tasks and ontology. The tool also includes eight types of intuitive annotation interfaces, explained in detail in Sec. 3.3.

The tool simplifies setup with a single configuration file that controls various customizable aspects. Overall, by combining the state-of-the-art technologies, *Antarlekhaka* offers a powerful and flexible solution for large-scale annotation projects.

3.1 Workflow

The workflow of the system is demonstrated in Fig. 2. The application is presented as a full-stack web-based software. It follows a single-file configuration system. An administrator may configure the tool and deploy it to web, making it immediately

[5] https://universaldependencies.org/format.html

[6] https://www.sqlalchemy.org/

[7] https://docs.sqlalchemy.org/en/20/dialects/

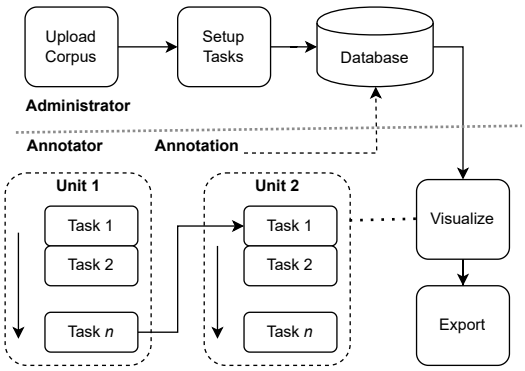

Figure 2: Workflow of *Antarlekhaka*

available for use. User registration is supported. User access is controlled by a 4-tier permission system, namely User, Annotator, Curator and Admin.

The tool has eight annotation interface templates corresponding to eight generic categories of NLP annotation tasks: sentence boundary detection, canonical word order, free-form token annotation, token classification, token graph construction, token connection, sentence classification, and sentence graph construction. Various NLP tasks can be modelled using each of these categories. More than one tasks of same category may be required for a specific annotation project. For example, named entity recognition (NER) and parts-of-speech (POS) tagging are both examples of token classification. To facilitate this, the administrative interface allows an administrator to create multiple tasks of each category. Additionally, an administrator can also control the set of active tasks, order of the tasks, ontology for the relevant tasks, corpus management and user access management.

We propose a streamlined sequential mode of annotation where an annotator completes multiple tasks for a single unit of text before moving on to the next unit. The set and order of tasks is customizable through an administrative interface. We consider a small logical block of text as a unit for the annotation, e.g., a verse from the poetry corpus.

3.2 Data

The data for corpus can be in either of two formats: *CoNLL-U* format or *plain text* format and can contain Unicode text. *CoNLL-U* is a widely used format for linguistic annotation, and it is based on the column format for treebank data. The format is designed to store a variety of linguistic annotations, including part-of-speech tags, lemmas, morphological features, and dependencies between words in a sentence. Data in *CoNLL-U* format can be obtained from treebanks such as Universal Dependencies (De Marneffe et al., 2021), which is a project that aims to develop cross-linguistically consistent treebank annotation. In addition, NLP tools such as Stanza (Qi et al., 2020) are capable of processing a general corpus of text and producing data in *CoNLL-U* format.

Plain text data is processed using a regular-expression based tokenizer, which is a process that splits the text into individual units of meaning, such as verses, lines and tokens using patterns defined in the form of regular expressions to identify the respective separators. The plain text processor module is a pluggable component. An administrator may reimplement it using any language specific features or tools as long as the data output by the module meets the current format specifications.

After the data has been imported, it is organized in a five-level hierarchy structure consisting of: Corpus, Chapter, Verse, Line, and Token. The hierarchical structure of the data provides a clear and organized framework for annotating and analyzing the data, making it easier to capture the relationships between different elements of the data.

3.3 Task Categories and Interfaces

The annotation supports annotation towards eight categories of annotation tasks and offers intuitive interfaces for each category. Annotators view the corpus in the form of text units (e.g., verses) on the left, and an annotation area on the right. After submitting annotations for a task, the interface automatically advances to the next task. Annotators are expected to complete all the tasks associated with a text unit before moving on to the next unit. This, however, is not strictly enforced, allowing annotator to still go back to modify annotations. Fig. 3 showcases the overall annotation interface.

The administrator can configure task-related information, including task titles, instructions, active tasks, and their order, through the administrative interface. This interface is illustrated in Fig. 5 (Appx. A). Tasks such as user access management, corpus creation, and ontology management also have intuitive administrative interfaces. Next, we provide a detailed description of each task category and its corresponding interface.

3.3.1 Sentence Boundary Detection

The importance of the sentence boundary task is not limited to languages without distinct sentence markers; it also pertains to poetry text, making it

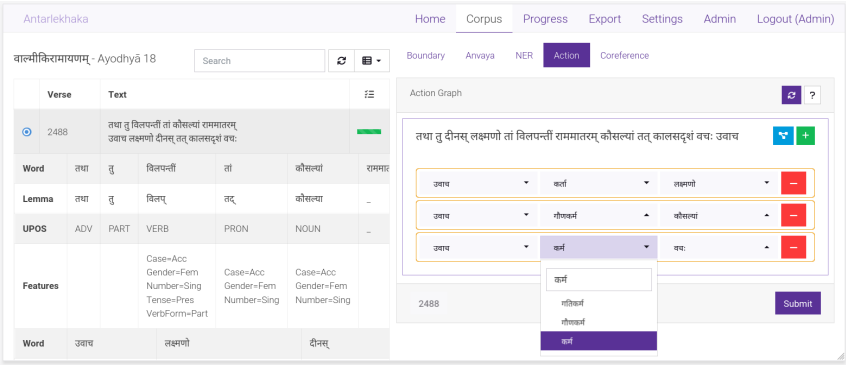

Figure 3: Annotation Interface: a Sanskrit corpus split into small units, and annotation area with task tabs

relevant to all languages.

The annotator's task is to identify and mark sentence boundaries by placing the delimiter '##' (two 'hash' symbols) at the end of each sentence in the provided editable text area prefilled with the original text. If the sentence does not end in the displayed unit, the user does not add any delimiters. After marking sentence boundaries, the user can proceed to the next annotation task. An illustration of this annotation task is shown in Fig. 6 (Appx. A).

It is worth mentioning that although the sentence boundary task is given primary citizen treatment, it can still be turned off for languages where it is not applicable. In such instances, the boundaries of annotation text units (e.g., verses) are treated as sentence boundaries.

3.3.2 Canonical Word Order

All sentences that end in the current unit of text are displayed to the annotator as a list of sortable tokens. The annotator can rearrange these tokens into the correct canonical word order by dragging them into place. Additionally, if any tokens are missing, the annotator can add them as well. A visual representation of this task is shown in Fig. 7 (Appx. A). The sorting capability is made possible through the use of the *jQuery UI (Sortable plugin)*[8].

3.3.3 Token Annotation

The token annotation interface allows an annotator to add free-form text associated with every token. This free-form text can have different purposes, such as to identify the root word of a word (lemmatization), to separate multi-word expressions into individual words (compound splitting), to analyze the morphological structure of a word (morphological analysis), etc. The token annotation interface is shown in Fig. 8 (Appx. A).

3.3.4 Token Classification

Token classification is a process of assigning predefined categories to individual tokens in text data. It is a special case of free-form token annotation, wherein the annotations are guided by an ontology. For such a task, an administrator must create an ontology. During the annotation process, an annotator is provided with a list of tokens, each accompanied by a dropdown menu, from which they can select the appropriate category for relevant tokens. Some common examples of token classification tasks include NER, dependency tagging, POS tagging, and compound classification. Fig. 9 (Appx. A) illustrates the token classification interface.

3.3.5 Token Graph

A token graph is a graph representation of the sentence, where the nodes are tokens belonging to a single sentence and the relations are based on an ontology. Tasks such as dependency parse tree, constituency graph, action graph are examples of tasks belonging to this category.

Semantic triple[9] is a standard format to represent and store graph-structured information in a relational database in a systematic manner. The interface allows an annotator to add multiple relations per sentence in the form of subject-predicate-object triples, where subject and object are tokens from the sentence and the predicate is a relation from the task specific ontology. The valid values of subject, object and predicate appear in individual dropdown menu elements for the annotator to choose from. Erroneous triples may also be removed. During the annotation process, an annotator can view the current status of the token graph at any time using the 'Show Graph' button. Fig. 10 (Appx. A) shows the token graph interface with graph visualization.

[8] https://api.jqueryui.com/sortable/

[9] https://en.wikipedia.org/wiki/Semantic_triple

3.3.6 Token Connection

Token connection is similar to token graph, however, there is a single type of relation to be captured. For example, when marking co-references, only connecting the two tokens to each other is sufficient, while the relationship 'is-coreference-of' is implicit. The tool provides a special simplified interface for this scenario. In addition to implicit relations, token connections can also be established across sentences. The annotator is presented with a list of clickable tokens from the current sentence as well as tokens from a context window of previous n (a configuration parameter with default as 5) sentences. The annotator can add a connection by clicking on the source token and the target token one after the other and confirming the connection. If a connection is added in error, it can be removed as well. In some cases, a connection might extend beyond the default context window. To address this, we have incorporated a button that an annotator can click to load additional context when needed. The token connection interface is shown in Fig. 11 (Appx. A).

3.3.7 Sentence Classification

Sentence classification is a task where sentences are classified into different categories, e.g., sentiment classification and sarcasm detection. This task is similar to ontology-driven token classification, with the difference being that classes are associated with sentences rather than tokens. The ontology is predefined by the administrator while setting up the task. The annotator can select the category for a sentence from a dropdown menu. Fig. 12 (Appx. A) illustrates the sentence classification interface.

3.3.8 Sentence Graph

Sentence graph is a graph representation of relationships between sentences, captured as subject-predicate-object triples. The connections can be between tokens or complete sentences. Tokens from the previous n (a configuration parameter with default as 5) sentences are presented as buttons arranged in the annotated word order. An annotator creates connections by clicking on the source and target tokens and selecting the relationship from a dropdown menu based on an ontology. A special token is provided to denote the entire sentence as an object. Tasks such as timeline annotation and discourse graphs are examples of tasks belonging to this category. Fig. 13 (Appx. A) shows the interface for creating sentence graph connections. Similar to the token graph task, an annotator can also visualize the sentence graph.

3.4 Clone Annotations

The administrative interface offers the capability to replicate annotations from one user to another. This feature proves valuable in cases where certain annotators possess expertise in specific tasks or if an annotator leaves a task incomplete, requiring another annotator to resume the task from their account. The cloned annotations are displayed just like regular annotations. However, they maintain the source information, including the annotator's ID and a reference to the original annotation.

3.5 Pluggable Heuristics

The tool supports the use of heuristics as 'pre-annotations' to assist annotators. Heuristics are custom functions that generate suggestions for the annotators to use or ignore. These heuristics are often specific to the language and corpus, and thus, must be implemented by the administrator when setting up the tool. The codebase of the tool outlines the format and type specifications of the heuristics, making them a pluggable component.

3.6 Progress Report

Detailed progress tracking serves multiple essential functions. Firstly, it provides project managers with the capability to allocate resources effectively and oversee the distribution of tasks among chapters. This facilitates the early detection of potential bottlenecks or areas needing extra focus. Moreover, the breakdown of progress contributes to streamlining the annotation process, guaranteeing the steady advancement of all chapters and tasks at an optimal rate. To facilitate this, we've developed an interface that offers a comprehensive overview of annotators' progress. This interface provides a detailed breakdown of advancements on a per-chapter and per-task basis, enabling thorough tracking and evaluation of their contributions.

3.7 Export

The export interface enables the access, retrieval and visualization of the annotated data for each task in a clear and straightforward manner. Annotator can easily view and export the data in two formats (1) a human-readable format for easy inspection and (2) a machine-readable format compatible with

the standard NLP tools. The specifics of the standard format depend on the task. For example, a standard format for NER datasets is the BIO format (Tjong Kim Sang and De Meulder, 2003), which stands for *begin*, *inside*, and *outside*. The B-tag marks the beginning of a named entity, while the I-tag indicates the continuation of a named entity. The O-tag signifies that a word is not part of a named entity. Fig. 14 (Appx. A) illustrates the export interface showcasing the capability to export NER data in the standard BIO format. The interface facilitates the visualization and export of graph representations for tasks related to graphs. The export interface is accessible not only to annotators but also to curators, allowing them to review annotations made by other users. This feature serves as a mechanism for quality control.

3.8 Language Independence

Unicode is a widely used standard for encoding, representing, and handling text in a uniform and consistent way across various languages, computing platforms and applications. The standard assigns unique numerical codes to each character in a large number of scripts. Thus, full Unicode support allows users to work with text data in their preferred languages.

4 Evaluation

The tool is being used for annotation of a large corpus in Sanskrit, namely, *Valmiki Ramayana*. The details of this project are described in Sec. 4.3. Additionally, the tool is also being used for the annotation of plain text corpus in Bengali language.

Following (Terdalkar and Bhattacharya, 2021), we have evaluated our tool using a two-fold evaluation method of subjective and objective evaluation. We have not used the time taken for annotation as an evaluation metric since annotators often spend more time processing the text to identify the relevant information than physically annotating. Hence, it may not be a reliable measure.

4.1 Subjective Evaluation

For the subjective evaluation, an online survey was conducted, with participation from 16 annotators. They were asked to rate the tool on a scale of 1 to 5 across various categories including ease of use, annotation interface, and overall performance. The feedback from annotators was predominantly positive. The tool received a score of 4.3 for ease

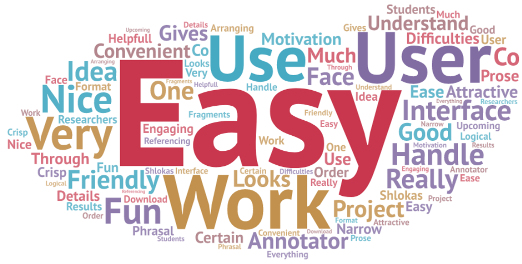

Figure 4: Word cloud of comments received in a survey

of use, 4.4 for annotation interface, and an overall score of 4.1. Furthermore, we gathered comments from users, and a word cloud over these comments can be found in Fig. 4.

4.2 Objective Evaluation

The objective evaluation utilized a scoring mechanism similar to that employed in previous studies (Neves and Ševa, 2021; Terdalkar and Bhattacharya, 2021). We retained the additional categories introduced by (Terdalkar and Bhattacharya, 2021) while incorporating supplementary categories pertinent to the comprehensive assessment of a general-purpose NLP tool. These supplementary categories were designed to gauge the tool's support for a diverse range of NLP tasks. Consequently, a total of 29 categories were employed for the evaluation process. Each tool was assigned a rating: 1 for full support of a feature, 0.5 for partial support, and 0 for the absence of a feature.

In this evaluation, *Antarlekhaka* achieved a score of 0.79, outperforming other tools like *Sangrahaka* (0.74), *INCEpTION* (0.74) and *FLAT* (0.71). The detailed list of the 29 categories used for objectively scoring the annotation tools can be found in Tab. 2.

4.3 Case Study: Large Sanskrit Text

The *Valmiki Ramayana*, an ancient Sanskrit text, presents a rich and intricate narrative with a diverse array of characters, events, emotions, and settings, making it an ideal choice for annotation purposes. We employ Antarlekhaka, for this large-scale annotation task, as a case study. The text, sourced from the Digital Corpus of Sanskrit (DCS) (Hellwig, 2010–2021), comprises a total of 18754 verses distributed across 606 chapters.

The annotation is being performed with the help of 119 annotators to annotate 5 tasks per verse. These annotations have resulted in the following *task-specific datasets*:

- *Sentence Boundary*: 1972 sentence markers across 1499 verses;
- *Canonical Word Ordering*: 1886 sentences;

Table 2: Objective evaluation criteria for annotation tools. Each feature is evaluated on a ternary scale of 0, 0.5 and 1, where 0 indicates absence of the feature, 0.5 indicates partial support and 1 indicates full support for the feature.

Criteria			Tools					
ID	**Description**	**Weight**	**INCEpTION**	**doccano**	**FLAT**	**BRAT**	**Sangrahaka**	**Antarlekhaka**
P1	Year of the last publication	0	1	0	0	1	1	1
P2	Citations on Google Scholar	0	1	0	0	1	0	0
P3	Citations for Corpus Development	0	1	0	0	1	0	0
T1	Date of the last version	1	1	1	1	0.5	1	1
T2	Availability of the source code	1	1	1	1	1	1	1
T3	Online availability for use	1	0	0	1	0	0	0
T4	Easiness of Installation	1	0	1	1	0.5	1	1
T5	Quality of the documentation	1	1	1	1	1	0.5	0.5
T6	Type of license	1	1	1	1	1	1	1
T7	Free of charge	1	1	1	1	1	1	1
D1	Format of the schema	1	1	1	1	0.5	1	1
D2	Input format for documents	1	1	0.5	1	1	1	1
D3	Output format for annotations	1	1	1	1	0.5	0	0
F1	Allowance of multi-label annotations	1	1	0	1	1	1	1
F2	Allowance of document level annotations	0	0	0	0	0	0	0
F3	Support for annotation of relationships	1	1	0	0	1	1	1
F4	Support for ontologies and terminologies	1	1	0	1	1	1	1
F5	Support for pre-annotations	1	0.5	0	0.5	0.5	0	0
F6	Integration with PubMed	0	0	0	0	0	0	0
F7	Suitability for full texts	1	0.5	0.5	1	1	1	1
F8	Allowance for saving documents partially	1	1	1	1	1	1	1
F9	Ability to highlight parts of the text	1	1	1	1	1	1	1
F10	Support for users and teams	1	0.5	0.5	1	0.5	0.5	0.5
F11	Support for inter-annotator agreement	1	1	0.5	0	0.5	0.5	0.5
F12	Data privacy	1	1	1	1	1	1	1
F13	Support for various languages	1	1	1	1	1	1	1
A1	Support for querying	1	0	0	0	0	1	0
A2	Crash tolerance	1	0	0	0	0	1	0.5
A3	Web-based / Distributed annotation	1	1	1	1	1	1	1
A4	Sequential Annotation	1	0	0	0	0	1	1
A5	Support for Sentence Boundary Annotation	1	1	0	0	0	0	1
A6	Support for Word Order Annotation	1	0	0	0	0	0	1
A7	Support for Token Classification Tasks	1	1	1	1	1	1	1
A8	Support for Sentence Classification Tasks	1	1	0	0	0	0	1
	Total	29	21.5	16.0	20.5	18.5	21.5	**23.0**
	Score		0.74	0.55	0.71	0.64	0.74	**0.79**

- *Named Entity Recognition*: 1717 entities in 947 verses based on custom ontology with 89 labels;
- *Co-reference Resolution*: 2271 co-reference connections across 950 verses;
- *Action Graph*: 90 action graphs with 720 relations from 71 verses, where an *action graph* captures action-related words, encompassing verbs, participles, and other action-denoting words and their relationships with other words.

We continue to enhance these datasets through the ongoing annotation endeavour.

5 Potential for NLP Research

A natural language annotation tool simplifies the process of creating datasets for machine learning models, which is useful for NLP tasks such as lemmatization, NER, POS tagging, co-reference resolution, text classification, sentence classification, and relation extraction. Annotator-friendly and intuitive interfaces can simplify the otherwise tedious process of manual annotation process to a great extent. The effectiveness of higher-level tasks such as question-answering, grammatical error correction and machine translation often relies on the success of several low-level tasks, which can be handled by a multiple task annotation tool. The tool's ability to handle large amounts of data and multiple users simultaneously can contribute to faster completion of these tasks.

6 Conclusions

We have developed a web-based multi-task annotation tool called *Antarlekhaka* for sequential annotation of various NLP tasks. It is available at https://github.com/Antarlekhaka/code. The tool is language-agnostic and has full Unicode support. The tool sports eight categories of annotation tasks and an annotator-friendly interface for each category. Multiple annotation tasks from each category are supported. The tool enables creation of datasets for computational linguistics tasks without expecting any programming knowledge from the annotators or administrators. It is actively being used for a large-scale annotation project, involving a large Sanskrit corpus and a significant number of annotators, as well as another annotation task in Bengali language. *Antarlekhaka* has a potential to propel research opportunities in NLP by simplifying the conduction of large-scale annotation projects.

7 Limitations

While *Antarlekhaka* is a powerful tool for annotation, it does have some limitations. These include:

- *Subjectivity in annotations*: Manual annotation can introduce subjective biases and inconsistencies among annotators. Currently, there is no in-built automated mechanism to ensure inter-annotator agreement and quality check is performed manually with a small set of curators. Nevertheless, we plan to implement inter-annotator agreement metrics in future.
- *Language-specific challenges*: Antarlekhaka may encounter challenges specific to certain languages or linguistic phenomena, including variations in syntax, morphology, or semantic nuances that may require additional customization or fine-tuning. Additionally, languages with complex orthographic systems or unique writing conventions may pose difficulties.
- *Dependency on sequential annotation*: Sequential annotation is one of the strong features of the tool. However, if the sentence boundary detection task is enabled, it imposes a dependency in the sense that other tasks can be performed only after marking the sentence boundaries.
- *Constraints on AI-guided annotation support*: While Antarlekhaka supports incorporation of heuristics as pluggable components, the in-built support for AI-guided annotation is still limited in nature. The inherent language-specific characteristics and limitations of various NLP tools present challenges in delivering a fully language-agnostic guided annotation system. However, our future plans involve the integration of established NLP tools to offer more comprehensive guidance for annotators during the annotation process.

8 Ethics Statement

The research conducted in the development and use of *Antarlekhaka* adheres to ethical considerations and guidelines. The annotation tasks performed using the tool involve the analysis and processing of language data. We ensure the following ethical principles:

- *Informed Consent*: Prior to engaging in annotation tasks, all annotators participating in the research are informed about the nature of the tasks, their purpose, and the potential use of the annotated data. Annotators provide their voluntary consent to participate.
- *Anonymity and Privacy*: All personal information of annotators is kept confidential and handled securely. The data collected during the annotation process is anonymized to protect the privacy of individuals involved.
- *Data Usage*: The annotated data is solely used for research purposes and in compliance with relevant data protection regulations. It is not shared with any third parties without explicit consent or legal requirements.
- *Bias Mitigation*: We strive to minimize any biases that may arise during the annotation process. Annotators are provided with guidelines and training to ensure consistency and fairness in their annotations. Regular quality checks are being performed to address any potential bias issues for the Sanskrit text corpus.
- *Annotator Pool*: Annotators for the Sanskrit text corpus are under-graduate and post-graduate level Sanskrit students from various institutes and colleges. This ensured that annotations were of accepted quality. Participation in the annotation task was voluntary, and everybody who wanted to annotate was allowed to do so.

By adhering to these ethical principles, we aim to contribute to the responsible advancement of Natural Language Processing technologies and promote ethical practices in language annotation research.

References

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. 2013. Gate teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029.

Marie-Catherine De Marneffe, Christopher D Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal dependencies. *Computational linguistics*, 47(2):255–308.

Manmatha Nath Dutt et al. 1891. *The Ramayana*, volume 1. Girish Chandra Chackravarti.

Karën Fort. 2016. *Collaborative annotation for reliable natural language processing: Technical and sociological aspects*. John Wiley & Sons.

Oliver Hellwig. 2010–2021. *The Digital Corpus of Sanskrit (DCS)*.

Hendrik Jacob Herik, Ana Paula Rocha, and Joaquim Filipe. 2018. *Agents and Artificial Intelligence: 9th International Conference, ICAART 2017, Porto, Portugal, February 24 {u2013} 26, 2017, Revised Selected Papers*. Springer International Publishing.

Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. 2018. The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In *proceedings of the 27th international conference on computational linguistics: system demonstrations*, pages 5–9.

Amba Kulkarni, Preethi Shukla, Pavankumar Satuluri, and Devanand Shukl. 2015. How free is free word order in sanskrit. *The Sanskrit Library, USA*, pages 269–304.

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. 2021. An extensive review of tools for manual annotation of documents. *Briefings in bioinformatics*, 22(1):146–163.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. *arXiv preprint arXiv:2003.07082*.

Lane Schwartz. 2022. Primum non nocere: Before working with indigenous data, the acl must confront ongoing colonialism. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, volume 2.

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.

Hrishikesh Terdalkar and Arnab Bhattacharya. 2021. Sangrahaka: A tool for annotating and querying knowledge graphs. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2021, page 1520–1524, New York, NY, USA. Association for Computing Machinery.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147.

Maarten van Gompel. 2014. Folia linguistic annotation tool. https://github.com/proycon/flat.

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.

A Screenshots of Various Interfaces

We showcase various interfaces of *Antarlekhaka* in this section. An administrative interface for managing tasks is shown in Fig. 5. Figs. 6 to 13 illustrate annotation interfaces for each task category. Fig. 14 highlights the export interface with the capability to export the data in the standard format.

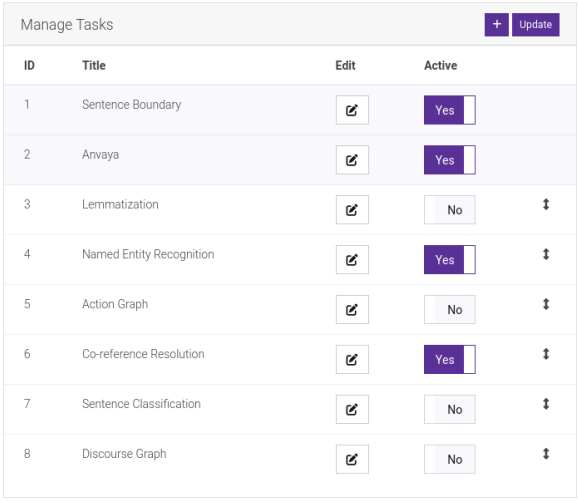

Figure 5: Task Management Interface: Add, Edit, Activate, Deactivate, Reorder Tasks

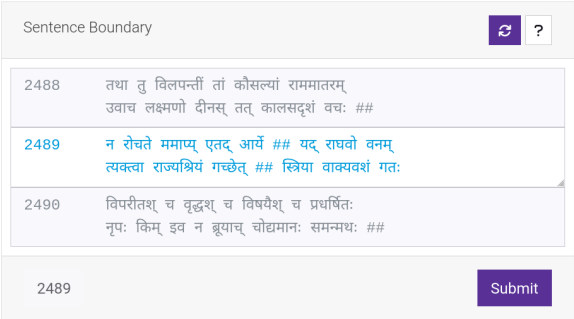

Figure 6: Sentence Boundary Annotation Interface

Figure 7: Word Order Annotation Interface

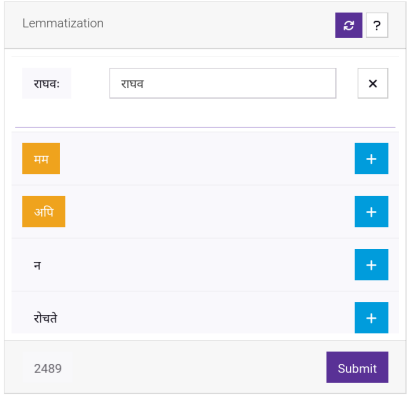

Figure 8: Token Annotation Interface: Lemmatization

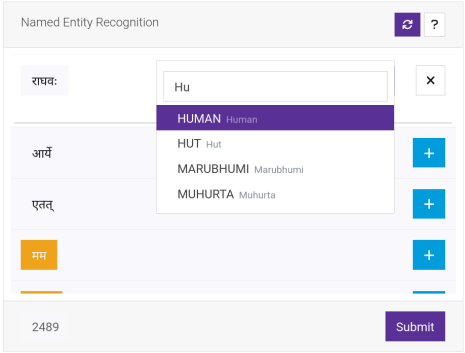

Figure 9: Token Classification Interface: Named Entity Recognition

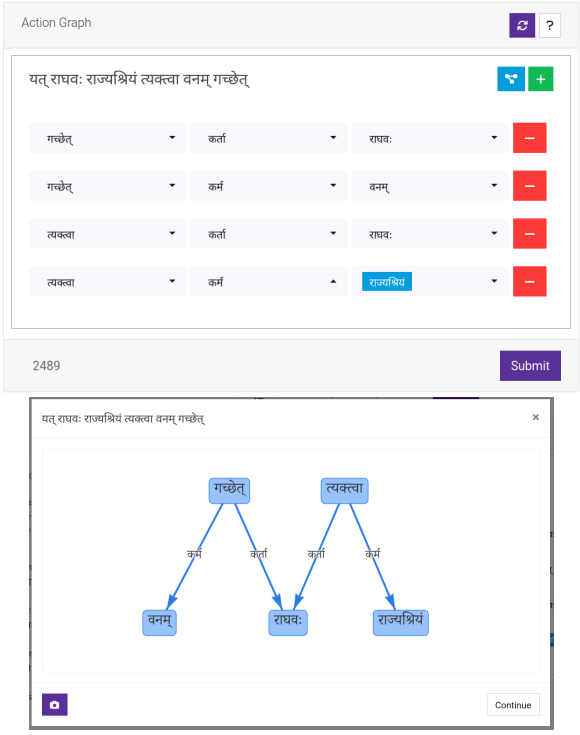

Figure 10: Token Graph Interface with Graph Visualization: Action Graph

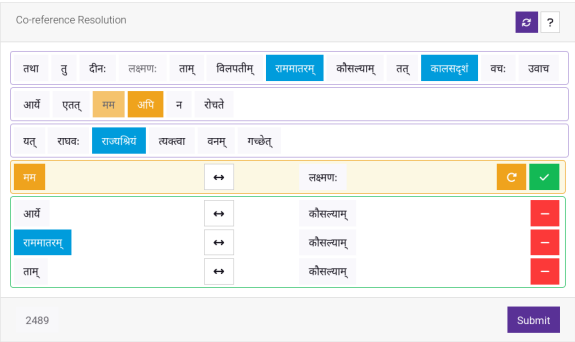

Figure 11: Token Connection Interface: Co-reference Resolution

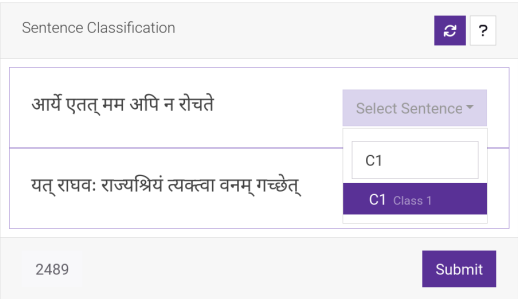

Figure 12: Sentence Classification Interface

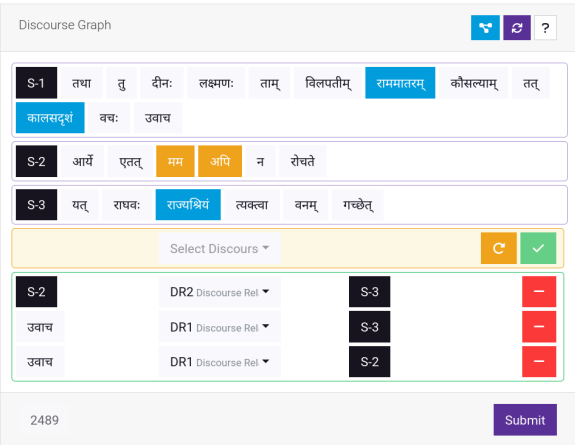

Figure 13: Sentence Graph Interface

B Schema

Antarlekhaka utilizes a relational database to store information such as, corpus data, user data, task data and annotations. A relational database allows for efficient storage and retrieval of functional data, as well as the ability to establish relationships between different pieces of data. For example, annotations of specific verses by specific users can be linked allowing the system to quickly locate and display relevant annotations when needed.

B.1 Tasks

The information regarding tasks is stored in a single table within the relational database. This table serves as a centralized repository for information related to each task, including its title, category, and instructions for annotators. Each task is assigned a unique identifier known as a 'task id', which serves as a means of easily referring or linking to a specific task.

B.2 Ontology

Ontology is required for four task categories: token classification, token graph, sentence classification, and sentence graph. The ontology information is stored as a flat list of labels in four separate tables, each specific to a particular task category. There may be multiple tasks corresponding to each of these categories. Therefore, every ontology table also has a 'task id' column which associates the ontology entries with the corresponding tasks. This setup allows for clear organization and linking of the ontology information with the relevant tasks.

B.3 Annotations

There are eight annotation tables, each corresponding to a different category of annotation tasks. Annotations of all tasks belonging to each category are stored in the corresponding table. The annotations are linked to the semantic units of text, specifically, the sentences marked in the sentence boundary task. The other seven annotation tables include a reference to the 'boundary id'. In cases where the sentence boundary task is not necessary, the boundaries of the annotation text units (e.g., verse) are considered as sentence boundaries and annotated automatically in the background using a special annotation user. Additionally, to facilitate multiple instances of tasks from each task category, every annotation table contains a reference to the 'task id'. Finally, each annotation table sports a tailored schema to support the recording of task specific annotations. An 'annotator id' associated with every task annotation table, allows for proper organization and tracking of the annotations.

Fig. 15 shows the Entity Relationship (ER) diagram on a subset of tables from the relational database of *Antarlekhaka*.

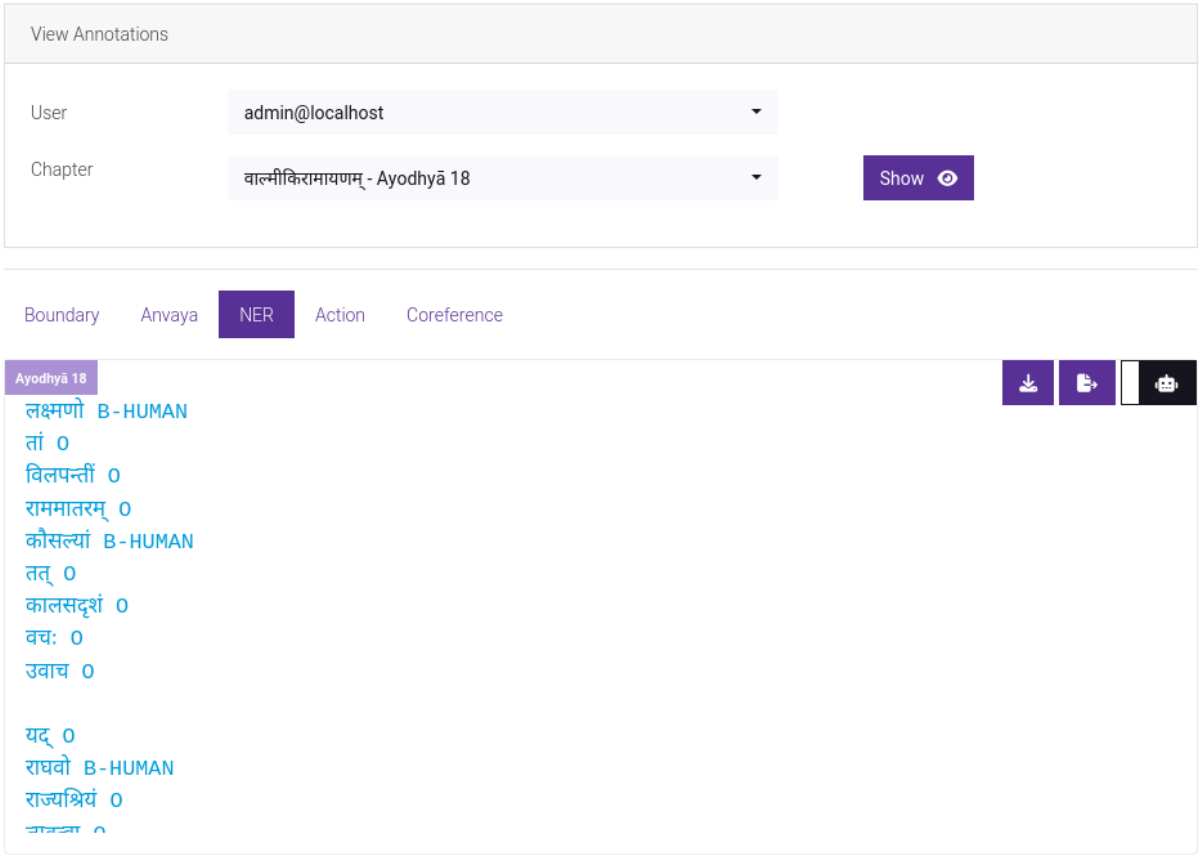

Figure 14: Export Interface: NER data in the standard BIO format

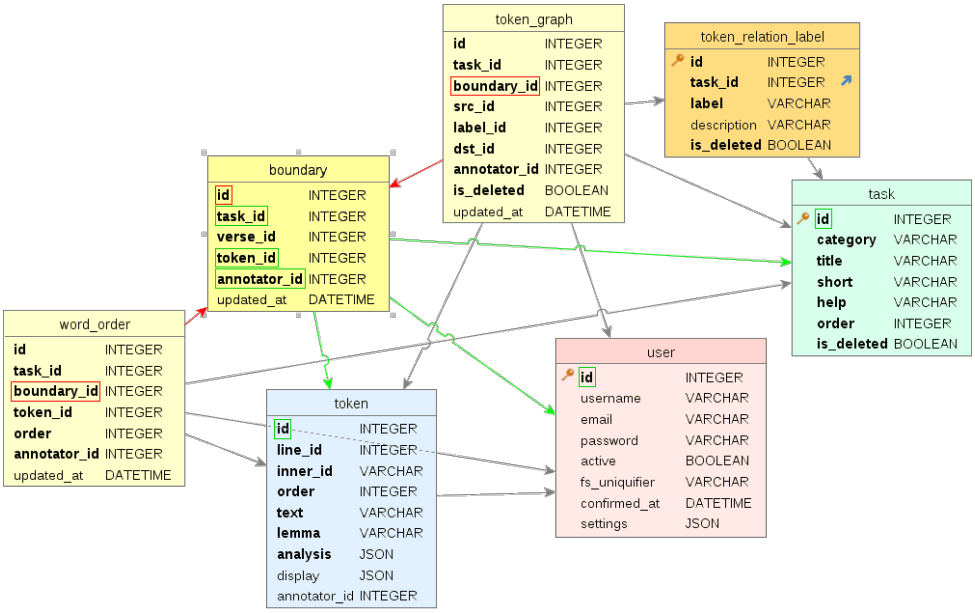

Figure 15: Entity Relationship Diagram illustrating some relevant links. Tables are color coded. Yellow: Annotation Tables, Orange: Ontology Tables, Blue: Corpus Tables, Pink: User Tables, Green: Task Information Table. The annotation table for 'Sentence Boundary' task is highlighted, showing the references incoming (red) and outgoing (green) references to other tables.