SAVI: Web-based Multilayered Semantic Annotation Validation Interface

Sashank Tatavolu[*], Soma Paul[*], Pratibha Rani[*], Sukhada Sukhada[**]
[*]IIIT Hyderabad, [**]IIT (BHU), Varanasi
sashank.tatavolu@research.iiit.ac.in, soma@iiit.ac.in, ranipratibha@gmail.com,
sukhada.hss@iitbhu.ac.in

Abstract

This paper presents SAVI, a web-based interface for multilayer semantic annotation validation of Universal Semantic Representation (USR). USR encodes meaning across interdependent lexical, constructional, relational, discourse, and co-reference layers, making validation challenging using conventional annotation tools. SAVI addresses this limitation through structured tab-based layer separation, constraint-aware editing mechanisms, and role-based review workflows. The system integrates a multilingual concept dictionary to ensure sense-level consistency, along with a Hindi text-generation module and dependency-based visualization to support interpretation and correction. SAVI is implemented using a Flask backend, Flutter frontend, and PostgreSQL for structured data management. Evaluation results demonstrate effective governance of concept proposals and improved efficiency in multilayer USR correction, positioning SAVI as a structured validation framework for scalable semantic corpus development.

Keywords: NLP, Annotation Tool, Semantic Representation, Annotation, Validation, Universal Semantic Representation, Indian Grammatical Tradition

1. Introduction

Semantically rich representations are essential for advanced NLP applications such as inference-based question answering, summarization, and knowledge-driven systems. However, validating semantic annotations can be cognitively demanding when representations span multiple interdependent linguistic layers.

This paper presents **SAVI**, a web-based interface designed for validating **Universal Semantic Representation (USR)**. USR encodes meaning across five interconnected layers: lexico-conceptual, constructional, relational, discourse, and co-reference. Because these layers interact with each other, errors in one layer can propagate across others, making annotation validation structurally complex.

SAVI addresses this challenge through a structured validation framework that separates annotation layers into dedicated tabs while enforcing backend constraints to maintain cross-layer consistency. The system also integrates a multilingual concept dictionary for sense-level governance, along with visualization modules and Hindi text generation to support semantic interpretation during validation.

Unlike existing annotation tools that primarily focus on single-layer or span-based annotation schemes, SAVI is designed specifically for synchronized multilayer semantic validation with controlled concept management and role-based review workflows.

We evaluate SAVI through dictionary governance statistics, validation efficiency analysis, and usability studies conducted with annotators. The results demonstrate improved validation efficiency and consistent multilayer annotation control in practical annotation settings.

The remainder of the paper is organized as follows. Section 2 introduces USR. Section 3 reviews related annotation tools. Section 4 describes the architecture and features of SAVI. Section 5 presents evaluation results. Section 6 concludes, followed by limitations in Section 7.

2. Overview of USR

Universal Semantic Representation (**USR**) has been designed and developed in the last five years by a research team to apply the insights of Pāṇini and the Indian Grammatical Tradition (**IGT**) (Sukhada and Paul, 2023) to modern Indian language technology (Paul et al., 2025; Sukhada et al., 2023). USR is a multi-layered semantic representation system at the document level.

Unlike other semantic representations that completely abstract away from syntactic representations and preserves semantic relations alone (Copestake et al., 2005; Van Gysel et al., 2021; Abzianidze et al., 2017), USR uniquely captures the speaker's intent, how he/she wants to express a situation through what is called **syntactico-semantic** (kāraka in IGT (Sukhada and Paul, 2023; Garg et al., 2023)) relations. For example, the concept '*train_1*' in the sentence given in Figure 1 is in reality an argument of the event '*move_1*' and is so represented in all existing semantic representations. However, the sentence conveys that the actor (*Ram*) boarded a *train* that was already in motion at that particular time. Therefore, '*move_1*' is represented as a present-participial modifier of the concept '*train_1*' in the USR, as shown in Figure 1. The benefit of

Proceedings of The Seventh International Workshop on Designing Meaning Representations (DMR 2026) @ LREC 2026, pages 81–90

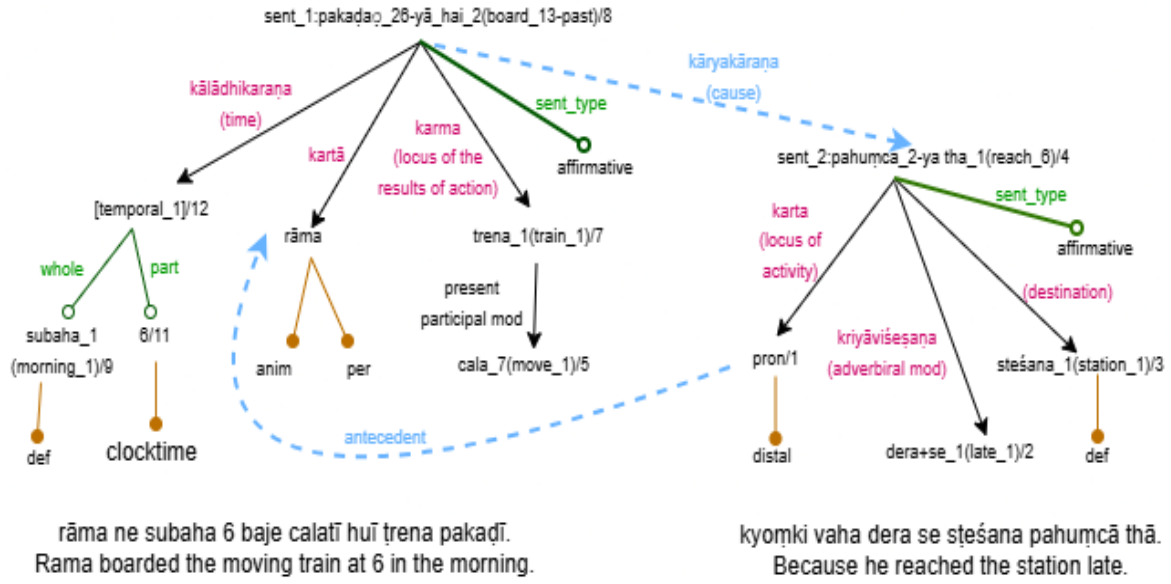

Figure 1: Universal Semantic Representation of a text snippet.

syntactico-semantic relations is that they capture how a speaker intends to delineate a situation. This representation is thus a quite suitable input to the Natural Language Generation system.

Figure 1 is an example USR for a short text snippet. The nodes in the graph represent unique concepts (*morning_1, move_1*) or Complex Concept (CC) labels (*[temporal_1]*). CCs specify complex expressions that have an internal structure. Each member of the CC is given a label, which is specified on the edge. In Figure 1, *[temporal]* has two components: *morning_1* and *6*, which are whole and part in a **part-whole** relationship. Different edge types signify different layers of information. For example, arrows (→) denote **dependency relations**, dotted arrows represent **discourse connective relations**, dotted lines specify **pronominal co-reference**, lines with square denote **CC components**, lines with circle denote **semantic categorical** information of a concept such as person, place, season, day-of-week, week-of-month, month-of-year, male/female and lines with diamond denote **pragmatic information** such as definiteness, respect, proximal/distal, emphasis, exclusiveness/inclusiveness, certainty - meaning that can be expressed by discourse particles.

3. Related Work

In this section, we review some of the annotation tools that have been developed to assist linguists and researchers in annotating text data. The latest web-based tool is EEVEE (Sorensen et al., 2024), which supports Named Entity Recognition (NER), POS tagging, and intent classification tasks.

UD Annotatrix (Tyers et al., 2018) is a language-independent browser-only client-side tool designed to edit dependency trees according to the guidelines of the Universal Dependencies (UD) project. UCCAApp (Abend et al., 2017) is a web-based tool for syntactic and semantic phrase-based annotation, particularly for the UCCA (Universal Conceptual Cognitive Annotation) framework.

BRAT Rapid Annotation Tool (BRAT) (Stenetorp et al., 2012) is a widely used web-based annotation tool that allows users to annotate textual data with entity relationships and dependencies. WebAnno (Yimam et al., 2013) provides annotation support for various linguistic layers, including syntax, co-reference, and discourse relations.

Prodigy (Montani and Honnibal, 2018) is an AI-assisted annotation tool that incorporates active learning to assist annotators.

UMR-Writer (Zhao et al., 2021) is a web-based application used to annotate Uniform Meaning Representation (UMR). UMR is a graph-based, cross-linguistically applicable semantic representation designed to support interpretable natural language applications that require deep semantic analysis.

TrEd (Mirovsky et al., 2010) is a customizable and programmable graphical editor and viewer of tree-like structures such as dependency trees. It is used as the primary annotation tool for syntactic and tectogrammatical annotations in the Prague Dependency Treebank (PDT).

While existing tools provide robust support for syntactic or frame-based annotation, they do not natively support multilayer semantic validation with enforced cross-layer constraints and sense-level dictionary governance. SAVI addresses this gap by integrating structured tab-based validation with

Feature	BRAT	INCEpTION	WebAnno	TrEd (PDT)	UMR-Writer	SAVI
Primary Focus	Entity & relation annotation	Multi-layer annotation + ML support	Multi-layer annotation	Syntax + semantic valency	Deep semantic representation	**Multilayer semantic validation (USR)**
Representation Type	Span + relation-based	Multi-layer structured	Multi-layer structured	Tree-based	Graph (sentence + document)	**Layered tab-based views**
Annotation Layers	Limited	Syntax, discourse, coref	Syntax, discourse, coref	Morphology, analytical, tectogrammatical	Concepts, relations, modality, co-ref	**Lexical, Constructional, Relational, Discourse, Co-ref**
Tool Interface	Web	Web	Web	Desktop	Web	**Web-based modular UI**
Multilingual Support	Yes	Yes	Yes	Czech-centric	Multilingual	**Multilingual (multi-script support)**
Co-reference Support	Limited	Yes	Yes	Yes	Yes	**Yes**
Language Generation Support	No	No	No	No	No	**Yes (Hindi generation)**
Constraint Enforcement	No	Partial	Partial	Limited	No	**Strong (tab-based constraints)**
Document-level Annotation	Limited	Yes	Yes	Limited	Yes	**Yes (speaker-level view)**
Unique Strength	Simplicity	ML-assisted annotation	Flexible annotation layers	Deep syntactic layering	Cross-lingual semantics	**Structured multi-layer validation + dictionary control**

Table 1: Comparison of **SAVI** with existing annotation tools and frameworks.

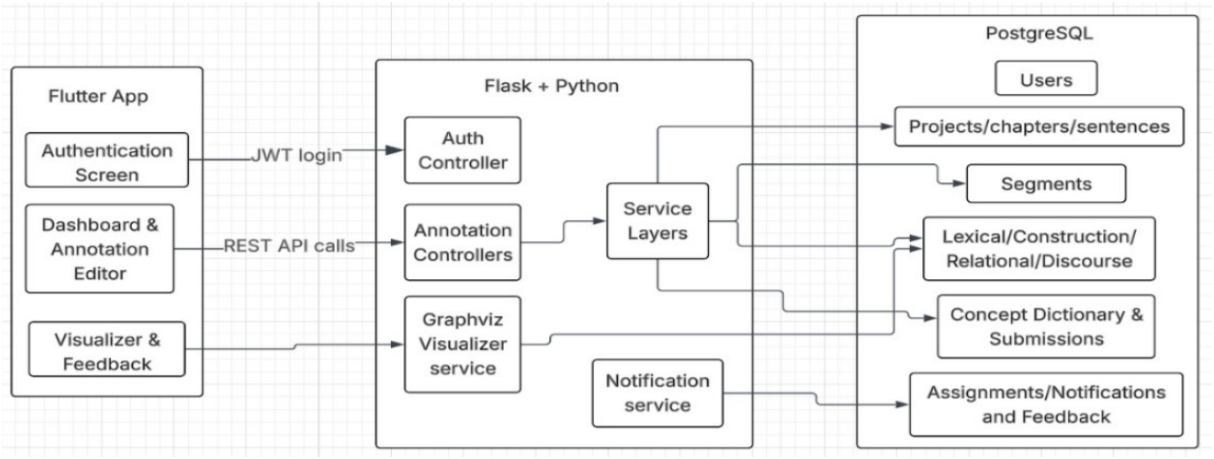

Figure 2: High-level system architecture of **SAVI** showing Backend (Flask), Database (PostgreSQL), and Frontend (Flutter) interaction.

role-based review workflows and concept-level consistency control.

USR is a multi-layer semantic representation that encodes lexical, relational, constructional, discourse, and co-reference information. Its interdependent structure makes annotation difficult using off-the-shelf tools, which primarily support single-layer or span-based schemes without cross-layer constraint enforcement or dictionary-governed validation. SAVI addresses this limitation by separating USR components into structured validation tabs, integrating sentence- and discourse-level visualization, and enforcing role-based consistency control.

Visualization tools such as Stanford NLP Dependency Visualizer tool (Manning et al., 2014), UDPipe (Straka et al., 2016), and spaCy (AI, 2020) present sentence-level graph structure. The SAVI USR-visualizer module provides graphical representations at both sentence and discourse levels using Graphviz (Ellson et al., 1991) and its component Digraph; thus making the tool a competent candidate for document-level semantic annotation. Table 1 presents the comparison of the features of **SAVI** with some of the existing annotation tools.

4. Our Proposed System Details

SAVI is designed for collaborative, distributed linguistic annotation and validation. This section presents the features of our tool, along with its architecture and implementation details.

4.1. Architecture and Implementation Details

SAVI follows a frontend–backend architecture (Figure 2) in which the frontend is implemented using Flutter (Google, 2017) and the backend is developed in Python (Python Software Foundation, 1991) using Flask (Ronacher and the Pallets team, 2010) framework, with modular blueprints for the projects, chapters, annotations, and dictionary management systems (explained in Section 4.3).

The PostgreSQL (PostgreSQL Global Development Group, 1996) database is used for persistent storage, accessed through the SQLAlchemy ORM (Bayer and contributors, 2006), ensuring ACID compliance and indexed lookups for annotation-heavy queries, and Gunicorn (Chesneau and Developers, 2025) for optimized performance. Graphviz is used to generate dependency trees, discourse and co-reference graphs, which are served to the frontend in SVG format.

SAVI provides a role-based user interface with dedicated dashboards for the **Admins**, **Annotators**, **Reviewers**, and **Dictionary Validators** roles.

4.2. User Roles

Our tool supports the following user roles with distinct responsibilities:

- **Admin**: This user role is responsible for uploading input data, assigning annotation tabs (explained in Section 4.3.2) to the validators, sending assignment notifications, and managing user roles.
- **Annotator**: This user role logs into the Flutter app, accesses the assigned segments, and annotates data tab by tab. Each tab enforces constraints (e.g., a head must be chosen in *Relational* **tab**, spans must be consistent in *Construction*). They can also submit new concept entries to the concept dictionary. Once satisfied, they finalize their work, which locks the tab until review.
- **Reviewer**: This user role receives annotated USR data for verification. They can either approve the finalized annotations or return them to the annotators with row-specific feedback. Notifications ensure that annotators are aware of required corrections. This iterative loop continues until the data gets final approval.
- **Dictionary Validator**: This user role reviews proposed new concepts, verifies linguistic accuracy, and approves/rejects them. Approved entries are added to the Concept Dictionary and linked back to the annotation layer.

4.3. Tool Features

This section presents the features available in our tool to help users have a smooth experience.

4.3.1. Chapter and Segment Tabs

The Chapter tab displays the running text for which semantic annotation is performed. The Segment tab displays segments and their segment IDs. Complex sentences are split into segments, with each segment having one finite verb. This tab allows the reviewer to edit segments, split a sentence into segments, if needed, and delete a wrong segment.

4.3.2. Semantic Annotation Tabs

The annotation editor is organized into five tab-based modules, each corresponding to a distinct USR layer:

- **Lexico-Conceptual Tab**(L): Displays concepts with unique ID, their semantic category, and also morpho-semantic and some pragmatic features such as definiteness, respect, honorificity, inclusiveness, exclusiveness, and emphasis associated with the concept under Speaker's View as shown in Figure 3.
- **Construction Tab**(C): Supports annotation of the components of Complex Concepts with enforced span consistency to maintain alignment with token boundaries as shown in Table 1 of Figure 4.
- **Relational Tab**(R): Enables dependency annotation between head and child concepts. Structural constraints enforce a single-head requirement, ensuring valid dependency structures. As shown in Table 2 of Figure 4, the dependency-annotation tab does not display the components of the Complex Concepts, thereby reducing annotators' cognitive load during validation. However, the information is still available in Table 1 of the same interface for annotators to consult if needed.
- **Discourse Connective Tab**(D): Captures discourse-level relations across segments (e.g., contrast, cause, elaboration) using predefined tagsets.
- **Co-reference Tab**(Coref): Supports annotation of pronoun and antecedent co-reference chains across segments via controlled selection mechanisms.

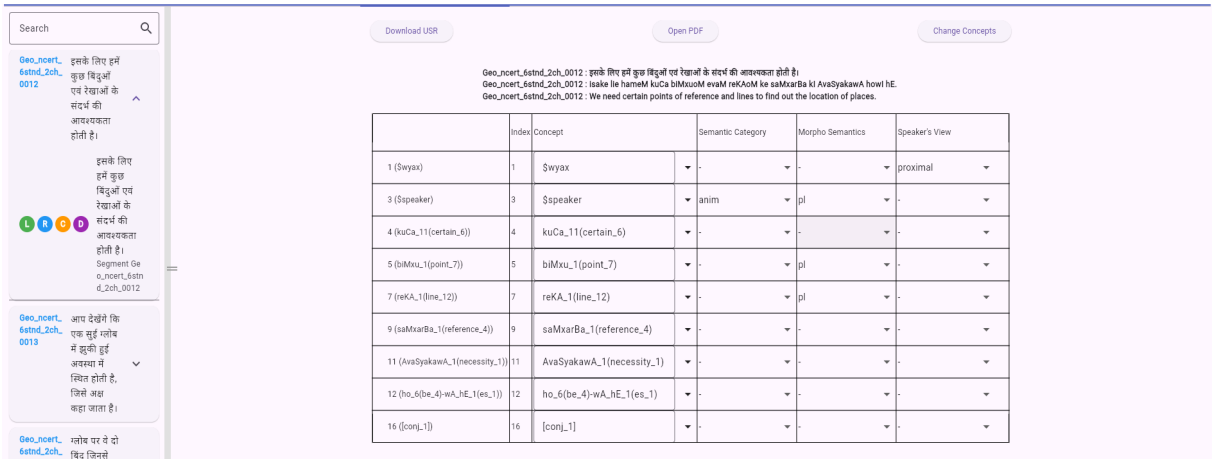

Figure 3: Lexico-conceptual tab of **SAVI**.

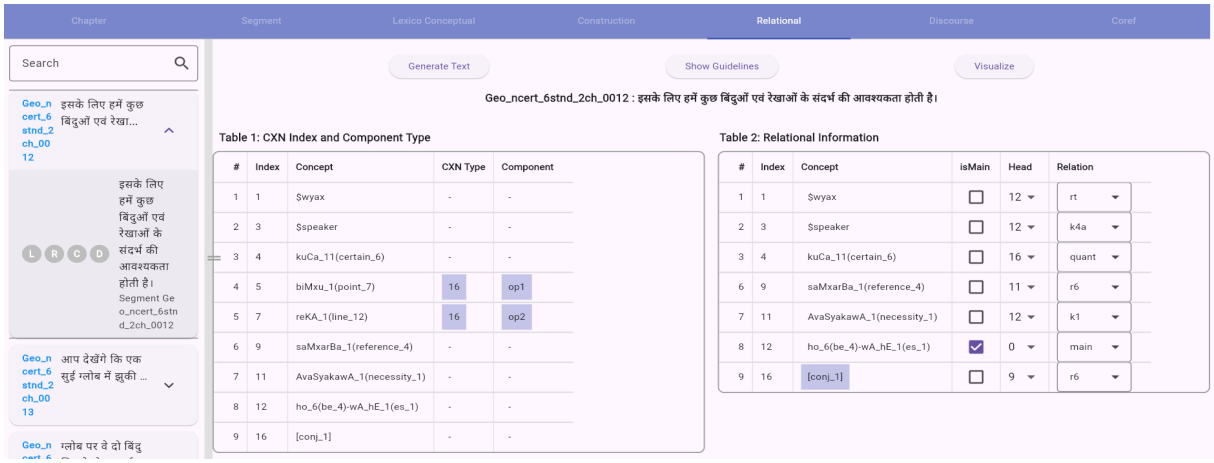

Figure 4: Relational tab of **SAVI**.

No tags are manually entered in these tabs. Instead, they are retrieved dynamically from the backend and displayed as drop-down menus, standardizing input across annotators and preventing invalid entries

4.3.3. Integrated Concept Dictionary

To ensure the postulation of unambiguous concepts and the cross-lingual alignment of concepts, SAVI integrates a **Concept Dictionary** in the Lexico-Conceptual Tab that stores sense representations of a concept across different languages. Each concept entry is assigned a unique `Concept_ID`, which is mapped to the sense representation across multiple languages. Currently, the dictionary includes entries for Hindi, English, Tamil, and Sanskrit. Operating at the sense level mitigates the ambiguity arising from homonymy and polysemy while enforcing stable cross-lingual equivalence.

Annotators select the correct Concept ID for a selected concept in the lexico-conceptual tab, and if no suitable ID exists, they can create a multilingual entry with an appropriate ID and submit it. After submission, the entry is stored with a *pending* flag.

Concept	Hindi	English	Tamil	Sanskrit
read_1	*paḍha_1*	*read_1*	*paṭi_1*	*paṭh_1*
study_1	*paḍha_2*	*study_1*	*paṭi_2*	*adhī_1*

Table 2: Illustrative sense-level multilingual mappings.

4.3.4. Review System

The Review System supports: (1) Concept review and (2) Validation of USR layers through various tabs.

For Concept review, the Dictionary reviewer can review the proposal with *pending* flag and *accept* or *reject* it. The reviewer can also edit the entry. This controlled mechanism prevents duplication and regulates the population of entries in the concept dictionary.

Notifications and email alerts are automatically

sent to annotators via SMTP (Klensin, 2008) when reviews are added. The system enforces a feedback loop that prevents annotators from finalizing unrelated segments until reviewer comments are resolved, ensuring data quality.

4.3.5. Visualization and Search Bar

SAVI provides a visualization module for USR data implemented using Graphviz. The visualizer helps annotators, reviewers, and validators interpret the hierarchical and relational structures encoded in annotations. It displays sentence-level dependency and construction structures as well as discourse-level links across segments. Examples of the visualization output are shown in Figure 5.

The interface also includes usability features to support efficient navigation during validation. A search bar allows users to locate annotation tags from predefined drop-down lists, and a quick-jump option enables direct navigation to a specific `segment_id`. In the segment sidebar, color coding indicates the completion status of annotation layers. Tabs corresponding to lexical (L), constructional (C), relational (R), and discourse (D) layers are highlighted with distinct colors when finalized, while unfinished layers appear in gray. This visual feedback enables annotators to quickly identify segments that require further validation.

4.4. System Workflows

The workflow shown in Figure 6 begins with the Admin uploading a chapter, its sentences, and the corresponding segments. Once uploaded, the system automatically indexes chapters, sentences, and segments, providing structured input for further processing.

Once uploaded, the Admin assigns segments to the validators for verification. Once verified, the segments are passed to the USR-Builder, which automatically creates multilayered USRs by applying a set of heuristics to the outputs of various NLP tools, including a Dependency Parser, a Morph Analyzer, a Named Entity Recognizer (NER), Discourse Connective markers, and Concept Identifiers.

This USR output is then parsed and stored in four database tables: lexico-conceptual (L), relational (R), construction (C), and Discourse (D). The information from these tables is displayed in separate tabs in the interface for manual validation and subsequent review.

Assignments for validation and review are role-specific. Annotators make corrections at L/R/C/D/Coref tabs (refer to Section 4.3.2) and finalize their work, and reviewers approve or return it with feedback. Role-based constraints ensure that annotators can only edit the segments and annotation types assigned to them.

Mode	Median (min)
SAVI-Assisted	2.84
Without Interface	3.73

Table 3: Median validation time per segment.

5. Evaluation, Results and Discussion

This section presents the statistics of the quantitative validation of the Concept Dictionary, the efficiency analysis, and the usability findings of SAVI in real annotation settings.

5.1. Validation Efficiency Analysis

To assess workflow efficiency, we compared median validation time per segment under two conditions: (i) SAVI-assisted validation using SAVI's structured multilayer interface and (ii) validation performed without structured interface support.

The experiment involved 10 annotators, each assigned 15 segments, resulting in 150 validated segments per condition. The same set of segments was used in both conditions to ensure comparability and to control for content complexity.

As shown in Table 3, SAVI-assisted validation reduces median correction time from 3.73 to 2.84 minutes per segment, corresponding to a **23.9% reduction in validation effort**. This improvement was observed consistently across annotators, suggesting that structured tab separation, constraint-aware editing, and controlled tag enforcement reduce annotator overhead related to formatting, structural alignment, and cross-layer consistency checking.

Even modest per-segment savings scale significantly for larger corpora, making the interface suitable for large-scale semantic corpus construction. Although the study is observational and limited in scale, the consistent improvement across annotators indicates that the efficiency gains are systematic and attributable to interface design.

5.2. Usability and Performance Evaluation

System usability was evaluated using the **System Usability Scale (SUS)** (Brooke, 1995, 2013), a widely used questionnaire consisting of ten statements rated on a five-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree," producing an overall usability score between 0 and 100.

Five annotators participated in the evaluation. Three had less than three months of experience with the tool, while two had more than one year of experience. SAVI obtained an **average SUS score of 66.5**, indicating acceptable usability. The dis-

Current Segment: Geo_ncert_7stnd_3ch_0002b: *"Ora ise jala ke AXe Bare blkara meM raKa xIjie."* Connected to Geo_ncert_7stnd_3ch_0002a: *"eka raMgIna kAgaja kI CotI-sI goll lIjie."*

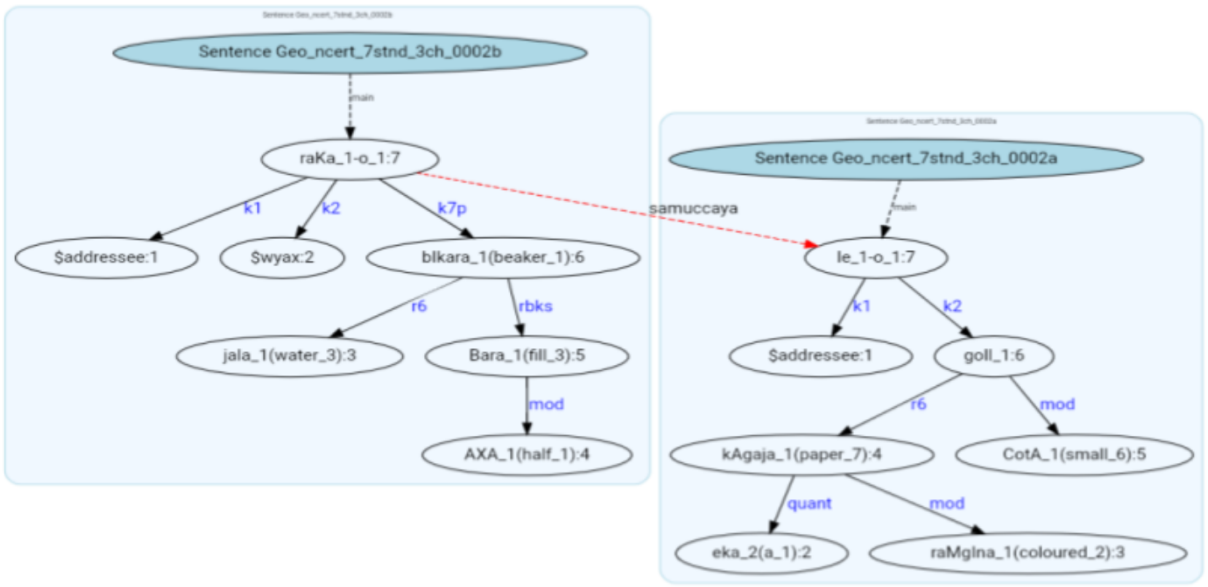

Figure 5: Sample of Visualization of Discourse relation provided by **SAVI**.

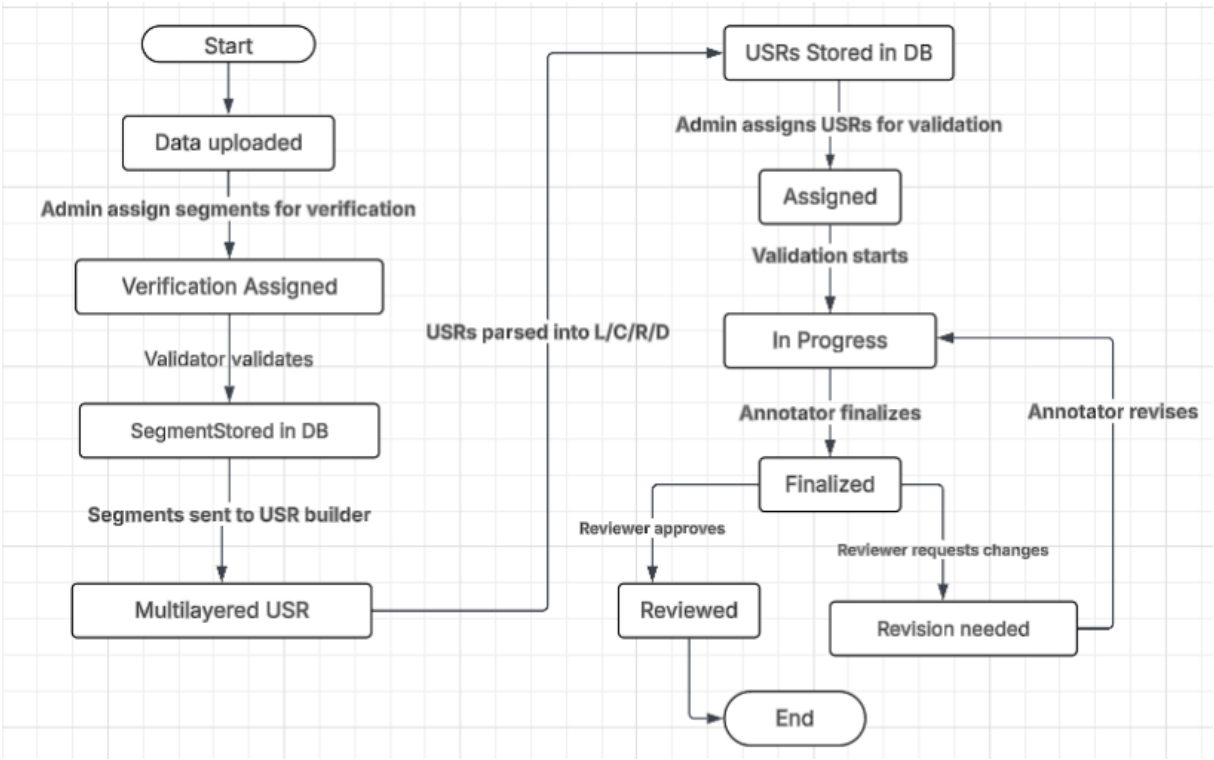

Figure 6: Overview of System Workflow of **SAVI**.

tribution of responses across SUS questionnaire items is shown in Figure 7. Most responses fall into the "Agree" or "Strongly Agree" categories, suggesting that users found the interface intuitive and well-integrated.

Additional usability feedback was collected through a short questionnaire assessing ease of use and interface features. Aggregated responses are summarized in Table 4. The majority of responses fall under the "Good" or "Excellent" categories, indicating that annotators found the interface easy to learn and effective for editing and vali-

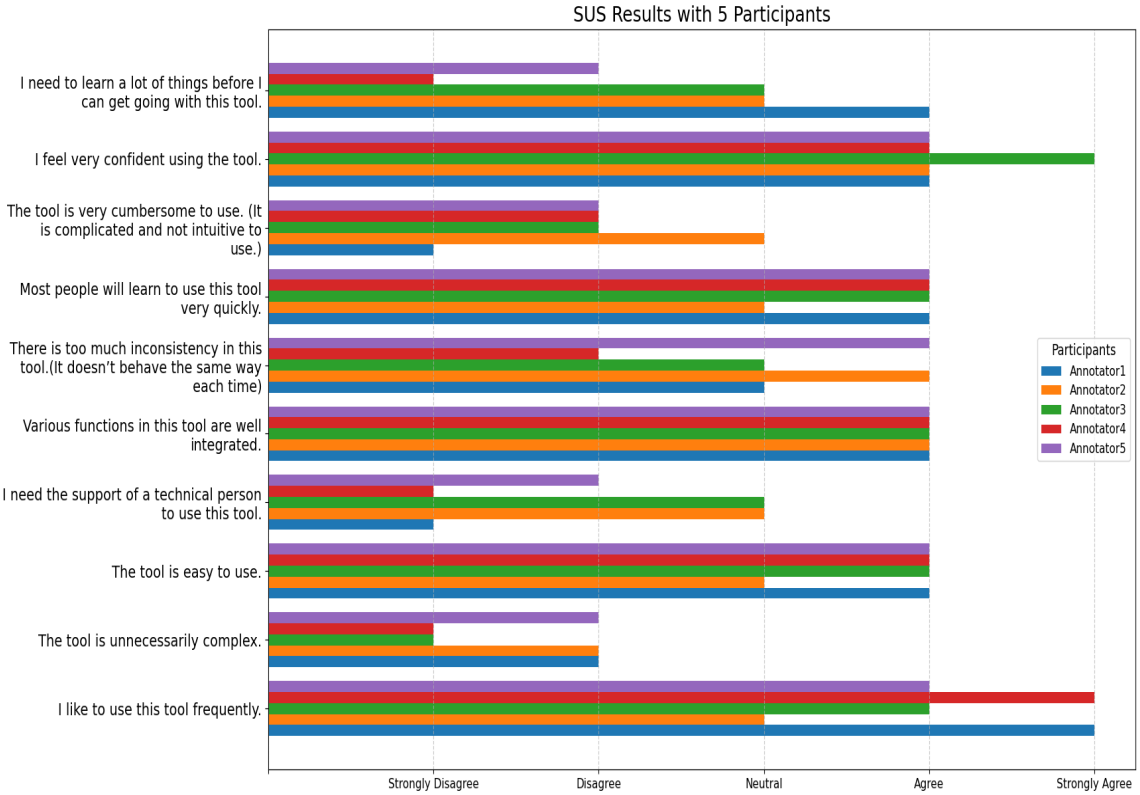

Figure 7: Distribution of System Usability Scale (SUS) responses across five annotators evaluating SAVI.

dation tasks.

Question	VP	P	Avg	G	Ex
Is the tool intuitive to use?	0	1	1	3	0
Is the tool easy to learn?	0	0	1	2	2
Is editing easy?	0	0	1	4	0
Is the tree/graph view helpful for editing?	0	0	1	3	1
Are features like search or shortcuts helpful?	0	0	1	3	1
Overall satisfaction with the tool	0	0	0	5	0

Table 4: User responses obtained from the usability survey on SAVI (VP=Very Poor, P=Poor, Avg=Average, G=Good, Ex=Excellent).

Technical performance responses indicated stable system behavior with predominantly instantaneous saving operations, as summarized in Table 5. Users reported moderate load times and fast response during editing operations.

Metric	Slow	Moderate	Fast
Load Time	0%	100%	0%
Saving Changes	0%	20%	80%

Table 5: User-reported technical performance of SAVI.

Early deployment showed minor stability issues, which were resolved through backend optimization and improved session handling. Overall, users reported that SAVI enables efficient multilayer validation while maintaining consistent annotation control.

6. Conclusion

This paper presented **SAVI**, a web-based interface for structured validation of Universal Semantic Representation (USR). Unlike conventional annotation tools, SAVI is specifically designed to support synchronized multilayer validation across lexico-conceptual, constructional, relational, discourse, and co-reference layers with enforced cross-layer constraints and dictionary-governed concept consistency.

Evaluation results demonstrate that SAVI supports controlled multilingual lexicon growth, reduces validation effort through a structured interface, and maintains multilayer structural consistency during annotation. Usability findings further indicate that tab-based separation and integrated visualization facilitate efficient and cognitively manageable validation workflows.

These results position SAVI as a structured validation framework suitable for scalable semantic corpus construction, particularly in low-resource and multilingual settings. Due to its modular and configurable architecture and a layered backend

design in which annotation types, tagsets, and validation constraints are defined independently of the interface logic, it can be adapted to other annotation schemes, also with minimal changes.

SAVI will be made publicly available as an open-source project under the GNU General Public License (GPL v3). The source code, along with detailed documentation and setup instructions, is available at: https://github.com/LC-Platform/Language_Communicator.git

Future work includes expanding AI-assisted validation mechanisms, improving automated structural suggestion modules, and extending support to additional languages and larger discourse-level corpora.

Limitations

The following are some of the limitations of our tool: (1) a preprocessing pass is required for the languages and scripts with complex segmentation, (2) specialized fonts or non-Unicode scripts may need normalization, (3) very large graphs (e.g., long-document discourse structures) may require chunked views and progressive rendering, and (4) for documents beyond the range of 5,000 tokens, increased memory consumption—primarily due to graph visualization and frontend rendering—can lead to slower response times.

Ethics Statement

SAVI is designed to support linguistic annotation and validation tasks and does not involve the collection of personal or sensitive user data. The system operates on textual datasets curated for research purposes, and no identifiable personal information is processed.

However, as with any language technology, biases present in the underlying data or annotation schemes may influence the outputs. Efforts should be made to ensure diversity and representativeness in annotated datasets. The system is intended for research use, and users are encouraged to critically assess outputs in downstream applications.

Acknowledgment

We are grateful to the annotators, especially Bidisha Bhattacharjee, Isma Anwar, Mohan, Rajni, Sabharaj, Satyaprakash, Sakshi, Muskan, Sanchari, Saumini, and Vandana, for their contributions to data preparation and experiments. This work forms part of the project titled "Sanskrit Knowledge Accessor" funded by MeitY, GoI, under the NLTM: BHASHINI scheme.

7. Bibliographical References

Omri Abend, Shai Yerushalmi, and Ari Rappoport. 2017. UCCAApp: Web-application for syntactic and semantic phrase-based annotation. In *Proceedings of ACL 2017, System Demonstrations*, pages 109–114, Vancouver, Canada. Association for Computational Linguistics.

Lasha Abzianidze, Johannes Bjerva, Kilian Evang, Hessel Haagsma, Rik Van Noord, Pierre Ludmann, Duc-Duy Nguyen, and Johan Bos. 2017. The parallel meaning bank: Towards a multilingual corpus of translations annotated with compositional meaning representations. *arXiv preprint arXiv:1702.03964*.

Explosion AI. 2020. spacy: Industrial-strength natural language processing in python. https://spacy.io/.

Mike Bayer and contributors. 2006. Sqlalchemy: The database toolkit for python. https://www.sqlalchemy.org/. Version used in project may vary.

John Brooke. 1995. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189.

John Brooke. 2013. Sus: a retrospective. *Journal of Usability Studies*, 8:29–40.

Benoît Chesneau and Gunicorn Developers. 2025. Gunicorn: Green unicorn {WSGI HTTP Server}. https://docs.gunicorn.org/. Accessed: 2025-08-28.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on language and computation*, 3(2):281–332.

John Ellson, Emden Gansner, Yifan Hu, Eleftherios Koutsofios, and Stephen North. 1991. Graphviz - graph visualization software. https://graphviz.org/.

Kirti Garg, Soma Paul, Sukhada Sukhada, Fatema Bawahir, and Riya Kumari. 2023. Evaluation of universal semantic representation (USR). In *Proceedings of the Fourth International Workshop on Designing Meaning Representations*, pages 13–22, Nancy, France. Association for Computational Linguistics.

Google. 2017. Flutter: Ui toolkit for building natively compiled applications. https://flutter.dev/.

J. Klensin. 2008. Simple mail transfer protocol (smtp). Technical Report RFC 5321, Internet Engineering Task Force.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Jivri Mirovsky, Lucie Mladova, and Zdenvek vZabokrtsky. 2010. Annotation tool for discourse in PDT. In *Coling 2010: Demonstrations*, pages 9–12, Beijing, China. Coling 2010 Organizing Committee.

Ines Montani and Matthew Honnibal. 2018. Prodigy: A new annotation tool for radically efficient machine teaching. https://prodi.gy. Accessed: 2025-04-10.

Soma Paul, Sukhada Sukhada, Bidisha Bhattacharjee, Kumari Riya, Sashank Tatavolu, Kamesh R, Isma Anwar, and Pratibha Rani. 2025. Indian grammatical tradition-inspired universal semantic representation bank (USR bank 1.0). In *Proceedings of the 1st Workshop on Benchmarks, Harmonization, Annotation, and Standardization for Human-Centric AI in Indian Languages (BHASHA 2025)*, pages 11–22, Mumbai, India. Association for Computational Linguistics.

PostgreSQL Global Development Group. 1996. Postgresql: The world's most advanced open source database. https://www.postgresql.org/.

Python Software Foundation. 1991. Python programming language. https://www.python.org/. Version 3.x.

Armin Ronacher and the Pallets team. 2010. Flask: A lightweight wsgi web application framework. https://flask.palletsprojects.com/. Version used in project may vary.

Axel Sorensen, Siyao Peng, Barbara Plank, and Rob Van Der Goot. 2024. EEVEE: An easy annotation tool for natural language processing. In *Proceedings of The 18th Linguistic Annotation Workshop (LAW-XVIII)*, pages 216–221, St. Julians, Malta. Association for Computational Linguistics.

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.

Milan Straka, Jan Hajič, and Jana Straková. 2016. UDPipe: Trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).

Sukhada Sukhada, Sirisipalli Veera Hymavathi, and Soma Paul. 2023. Generation of mrs abstract predicates from paninian usr. In *Proceedings of the 30th International Conference on Head-Driven Phrase Structure Grammar, University of Massachusetts Amherst*, pages 122–142, Frankfurt/Main. University Library.

Sukhada Sukhada and Soma Paul. 2023. Theory of sāmarthya in Indian Grammatical Tradition: The foundation of Universal Semantic Representation. In *Int J Sanskrit Res*, pages 17–22.

Francis M. Tyers, Maria Sheyanova, and Jonathan Washington. 2018. Ud annotatrix: An annotation tool for universal dependencies. In *Proceedings of the 16th International Workshop on Treebanks and Linguistic Theories (TLT16)*, pages 10–17, Prague, Czech Republic. Association for Computational Linguistics. Distributed under a CC-BY 4.0 licence.

Jens EL Van Gysel, Meagan Vigus, Jayeol Chun, Kenneth Lai, Sarah Moeller, Jiarui Yao, Tim O'Gorman, Andrew Cowell, William Croft, Chu-Ren Huang, et al. 2021. Designing a uniform meaning representation for natural language processing. *KI-Künstliche Intelligenz*, 35(3):343–360.

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.

Jin Zhao, Nianwen Xue, Jens Van Gysel, and Jinho D. Choi. 2021. UMR-writer: A web application for annotating uniform meaning representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 160–167, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.