

Thresh A Unified, Customizable and Deployable Platform for Fine-Grained Text Evaluation

David Heineman, Yao Dou, Wei Xu

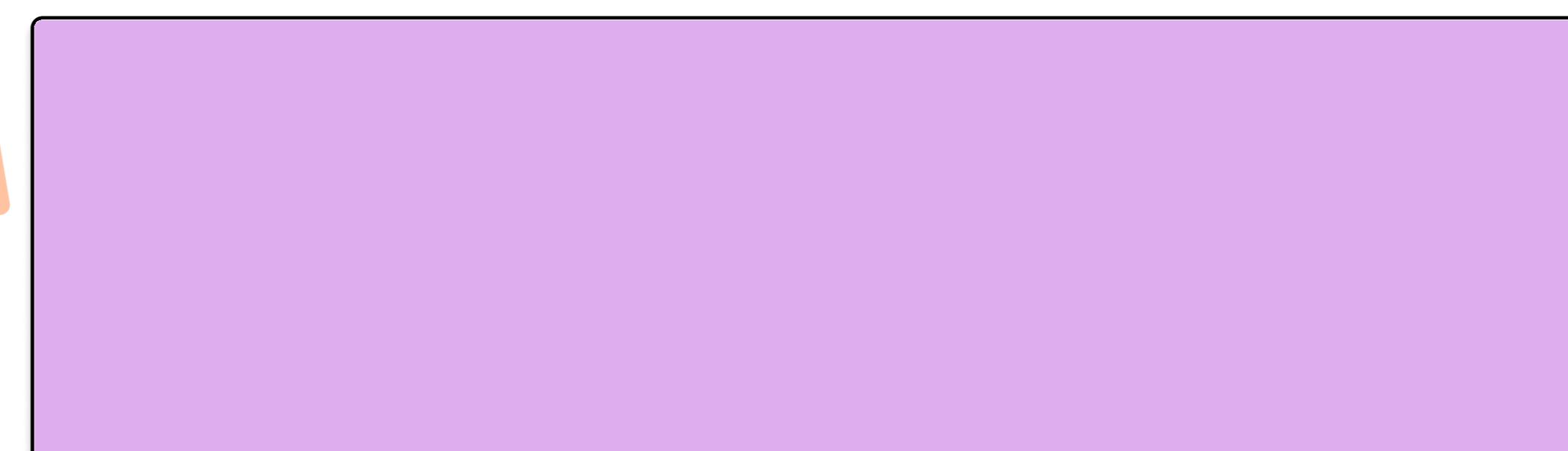


The screenshot shows a web-based annotation interface. At the top, there's a toolbar with various icons for file operations, zooming, and search. Below the toolbar is a main text area containing a multi-lingual document. Several spans of text are highlighted in different colors (blue, green, yellow) with corresponding annotations. A sidebar on the left provides context for the selected spans. At the bottom, there's a navigation bar with buttons for saving and exiting.

① Unified Text Span Selection & Annotation

- Fine-grained Evaluation — Text-to-text or open-ended evaluation consisting of **span selection** and **span annotation**.
- First tool built specifically for text generation evaluation, with tools tailored towards NLP tasks. Actively updated with new features + interfaces
- Build an interface **in-browser** and share with a **.YAML** file
- Open-source and portable to integrate with existing annotation management tools

Complex, overlapping span selection!



- 3 types: Single-, Multi-, Composite
- Specify span selection by token boundaries

② Highly Customizable

The interface includes two main views: 'Span Selection View' and 'Span Annotation View'. The Selection View shows a list of annotated spans with their start and end positions. The Annotation View shows a detailed view of a selected span, allowing users to edit its structure, grammar, and other properties. Both views have various configuration options and dropdown menus for customization.

- **Annotator Adjudication.** Deploy interfaces side-by-side for manual inspection
- **14 Languages Supported.** Fully custom interface text, supports zh, en, es, hi, pt, bn, ru, ja, vi, tr, ko, fr, ur
- **Paragraph Annotation.** Specify additional context and display options to reduce lengthy long-form annotation

③ Deploy to MTurk, Prolific + More!

The interface provides a 'Deployment' section where users can choose to deploy to MTurk, Prolific, or SALSA. It shows examples of how the interface looks on different platforms (web browser, mobile phone, and SALSA interface). It also includes a 'Report Data' section for tracking and analysis.

- Upload your interface directly, or host your own template in Github or Huggingface to share and visualize data collected from your project
- Deployment tutorials for **MTurk** and **Prolific**
- Integration with lightweight databases like **Google Firebase** to manage collected annotations

12 existing implementations across MT, open-ended generation, summarization!

Framework	Task	Released	Link
Evaluation			
MQM (Freitag et al., 2021)	Translation	✓	thresh.tools/mqm
FRANK (Pagnoni et al., 2021)	Summarization	✓	thresh.tools/frank
SnAC (Goyal et al., 2023b)	Narrative Summarization	✓	thresh.tools/snac
Scarcrow (Dou et al., 2022a)	Open-ended Generation	✓	thresh.tools/scarcrow
SALSA (Heineman et al., 2023)	Simplification	✓	thresh.tools/salsa
ERRANT (Bryant et al., 2017)	Grammar Error Correction	✗	thresh.tools/errant
FG-RLHF (Wu et al., 2023)	Fine-Grained RLHF	✓	thresh.tools/fghlf
Inspection			
MultiSIP (Dou et al., 2022b)	Paraphrase Generation	✗	thresh.tools/multisip
CWZCC (Himero and Pareja-Lora, 2020)	Zambomba Chavacano Spell Checking	✗	thresh.tools/cwzcc
Propaganda (Da San Martino et al., 2019)	Propaganda Analysis	✓	thresh.tools/propaganda
arXivEdits (Jiang et al., 2022)	Scientific Text Revision	✓	thresh.tools/arxivedits

④ End-to-End Integration with Python

```
pip install thresh
```

```
from thresh import load_interface, convert_dataset

# Load SALSA data using the SALSA typology
SALSA = load_interface("salsa.yml")
salsa_data = SALSA.load_annotations("salsa.json")

# Convert to the thresh.tools standardized format
thresh_data = convert_dataset(
    data_path=<path_to_original_data>,
    dataset=<dataset_name>
)
```

- Integrate your interface with the **thresh** library, installable from PyPi
- Convert to the **unified data model** from existing interfaces / data in one line
 - Supports type checking of collected annotations, parsing into Python functions, and recursive annotation!
- **Thresh** has already been used in large-scale data annotation, and we release demo notebooks for fine-grained data analysis tools

```
print(cwzcc_data[0])

Annotation(
    target = "Pirmi man iyo ta sinti ...",
    edits = [
        ...
        Edit(
            edit_id = 5,
            text = "Nusabe",
            output_idx = [31, 37],
            category = "unintentional",
            annotation = Annotation(
                error_type = ErrorType(
                    non_random = NonRandom(
                        regular_error = RegularError(
                            segmentation = Segmentation(
                                val = "space_omission"
                            )
                        )
                    )
                )
            )
        ),
        ...
    ]
)
```