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

David Heineman, Yao Dou, Wei Xu
School of Interactive Computing, Georgia Institute of Technology
{david.heineman, douy}@gatech.edu; wei.xu@cc.gatech.edu

Abstract

Fine-grained, span-level human evaluation has emerged as a reliable and robust method for evaluating text generation tasks such as summarization, simplification, machine translation and news generation, and the derived annotations have been useful for training automatic metrics and improving language models. However, existing annotation tools implemented for these evaluation frameworks lack the adaptability to be extended to different domains or languages, or modify annotation settings according to user needs; and, the absence of a unified annotated data format inhibits the research in multi-task learning. In this paper, we introduce Thresh 🌾, a unified, customizable and deployable platform for fine-grained evaluation. With a single YAML configuration file, users can build and test an annotation interface for any framework within minutes – all in one web browser window. To facilitate collaboration and sharing, Thresh provides a community hub that hosts a collection of fine-grained frameworks and corresponding annotations made and collected by the community, covering a wide range of NLP tasks. For deployment, Thresh offers multiple options for any scale of annotation projects from small manual inspections to large crowdsourcing ones. Additionally, we introduce a Python library to streamline the entire process from typology design and deployment to annotation processing. Thresh is publicly accessible at https://thresh.tools.

1 Introduction

As modern large language models are able to generate human-level quality text (Brown et al., 2020; OpenAI, 2023), the evaluation of these models becomes increasingly challenging. Recent work has shown traditional surface-level evaluation methods such as pairwise comparison or Likert-scale ratings become less reliable (Clark et al., 2021; Maddela et al., 2023) due to the close performance of these LLMs. To address this, several fine-grained human evaluation frameworks have been proposed for various tasks such as open-ended generation (Dou et al., 2022a), text simplification (Heineman et al., 2023), and machine translation (Freitag et al., 2021). In these frameworks, annotators identify and annotate specific spans corresponding to quality or errors in the generated text.

Figure 1: Examples of fine-grained evaluation frameworks implemented on Thresh. In order: SALSA (Heineman et al., 2023), MQM (Freitag et al., 2021), Scarecrow (Dou et al., 2022a).

However, each of these evaluation frameworks releases its own dedicated annotation interface that is difficult to modify or adapt to different evaluation schemes, thus limiting the customizablility. For example, Scarecrow's typology (Dou et al., 2022a), which is designed for open-ended text generation

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 336–345

Framework	Task	Released
Evaluation		
MQM (Freitag et al., 2021)	Translation	✓
FRANK (Pagnoni et al., 2021)	Summarization	✓
SNaC (Goyal et al., 2022b)	Narrative Summarization	✓
Scarecrow (Dou et al., 2022a)	Open-ended Generation	✓
SALSA (Heineman et al., 2023)	Simplification	✓
ERRANT (Bryant et al., 2017)	Grammar Error Correction	✗
FG-RLHF (Wu et al., 2023)	Fine-Grained RLHF	✓
Inspection		
MultiPIT (Dou et al., 2022b)	Paraphrase Generation	✗
CWZCC (Himoro and Pareja-Lora, 2020)	Zamboanga Chavacano Spell Checking	✗
Propaganda (Da San Martino et al., 2019)	Propaganda Analysis	✓
arXivEdits (Jiang et al., 2022)	Scientific Text Revision	✓

Table 1: Existing typologies currently implemented on Thresh. *Released* indicates whether the annotated data is released. Corresponding links on Thresh for each framework can be found in Table 2 in the Appendix.

for news, may require modifications when applied to other domains such as story or scientific writing. Frameworks like MQM (Freitag et al., 2021) only allow selections of the spans in the target sentence, restricting the ability to select the associated source spans in error categories such as mistranslation. Furthermore, modern LLMs are ideally evaluated on multiple tasks (Hendrycks et al., 2021), but the lack of a unified annotation tool makes this process inconvenient. Considering the recent success of multi-task instruction fine-tuning (Wei et al., 2021; Sanh et al., 2021), a standardized annotation format would enable research in multi-task learning with fine-grained human feedback.

To this end, we present Thresh 🌾: a unified and customizable platform for building, distributing and orchestrating fine-grained human evaluation for text generation in an efficient and easy-to-use manner. Our platform allows users to create, test and deploy an evaluation framework within minutes, all in a single browser window and has already been used to orchestrate large-scale data annotation (Heineman et al., 2023). Thresh also serves as a *community hub* for fine-grained evaluation frameworks and annotation data, all presented in a unified format. Figure 1 displays three examples of evaluation frameworks built on Thresh. The following are the design principles of Thresh:

- **Unified:** Thresh standardizes fine-grained evaluation into two key components: span selection and span annotation. Users can easily implement any framework by writing a YAML template file (see Figure 5), and Thresh will build the corresponding annotation interface. All resulting annotations adhere to a consistent JSON format.
- **Customizable:** Thresh offers extensive customization to meet a wide range of user needs. This includes different span selection methods from subword to word-level, diverse annotation options including custom questions and text boxes to handle arbitrary typologies, as well as customized interface elements in any language.
- **Deployable:** Thresh supports a range of deployment options for annotation projects of various scales. Small-scale linguistic inspections (e.g., manual ablation studies) can be directly hosted on the platform. For larger projects, users can host their template in a GitHub repository and connect to Thresh. Thresh is also compatible with crowdsourcing platforms such as Prolific[1] and Amazon MTurk[2].
- **Contributive:** Thresh also operates as a community hub where users can contribute and access a wide variety of fine-grained evaluation frameworks and their annotation data. Currently, it includes 11 frameworks as displayed in Table 1.
- **End-to-End:** Beyond facilitating the creation and deployment of evaluation frameworks, Thresh streamlines every step of the annotation process. It offers functions for authors to publish their typologies as research artifacts and a supplementary Python library, released under the Apache 2.0 license, to help data collection.[3]

2 Related Work

Fine-grained Text Evaluation. Given the limitations of traditional human evaluation methods such as Likert-scale and pairwise comparison in the era of LLMs, many recent studies have proposed fine-grained human evaluation frameworks. Dou et al. (2022a) introduces Scarecrow to capture error spans in open-ended text generation for news, MQM (Freitag et al., 2021) identifies errors in machine translation, and FRANK (Pagnoni et al., 2021) captures factual errors in abstractive text summarization. We list other evaluation and inspection typologies in Table 1. However, these existing frameworks usually develop their own annotation tools which lack customizability and universality, making them difficult to adapt to other languages or domains, or to new annotation settings. Recently, Goyal et al. (2022a) proposes

[1] https://www.prolific.co
[2] https://www.mturk.com
[3] https://www.pypi.org/project/thresh

FALTE, customizable span-level error highlighting for long text evaluation, but it only includes a subset of features offered by Thresh, limiting its ability to implement complex typologies such as SALSA (Heineman et al., 2023). Specifically, FALTE only highlights errors without rating their severity or efficacy, does not support multi-span or composite selection, and cannot select overlapping spans. Moreover, its lack of a tree structure can make the interface cluttered if there are more than a handful of categories. Thresh instead builds unified and customizable support across task setups.

Annotation Tool. Accessible and replicable annotation tools have been a persistent goal for NLP tasks. Stenetorp et al. (2012) introduces BRAT, the first web browser-based annotation tool and Yimam et al. (2013) further improves BRAT on speed and label configuration. In recent years, a new generation of universal annotation tools have been introduced by academia and industry, including Prodigy (Montani and Honnibal, 2018), Doccano (Nakayama et al., 2018), LightTag (Perry, 2021), and POTATO (Pei et al., 2022). Focusing on universality, these tools allow authors to add custom UI elements such as multiple choice questions, text boxes or pairwise comparison. However, these surface-level annotation options are not sufficient to implement complex typology setups demanded by fine-grained evaluation, which are typically structured by decision trees (Heineman et al., 2023). Thresh addresses this gap by recursively building the interface, which allows for nested questions. Besides, Thresh encourages sharing and reproducibility by providing a community hub where users can upload their new or use existing fine-grained frameworks and annotated data.

Span-level Annotation. Span-level annotation has a long history across NLP tasks. In Named Entity Recognition (NER), spans are selected and labeled as names of persons, organizations, locations, or other entities (Tjong Kim Sang and De Meulder, 2003). Word alignment focuses on selecting aligned words or phrases between two parallel corpora across languages (Och and Ney, 2003), or within monolingual tasks (Lan et al., 2021). Span selection has also been used for question answering such as in SQuAD (Rajpurkar et al., 2016), where the answer is defined by a span within the document context. Furthermore, extractive text summarization (Hermann et al., 2015) highlights

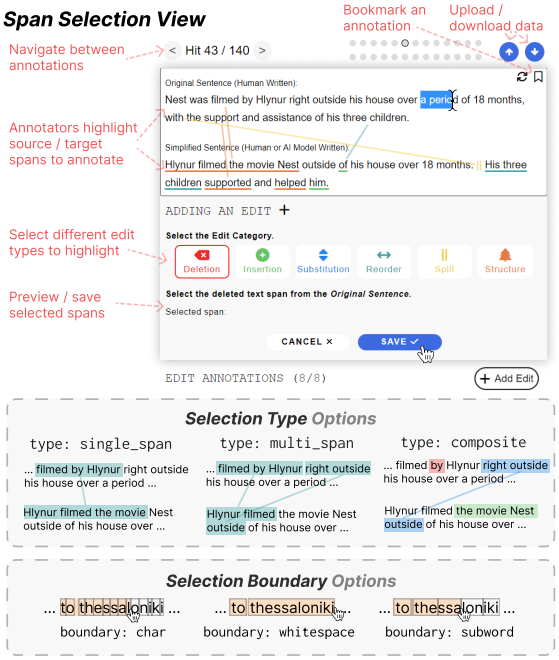

Figure 2: The span selection component of Thresh, customized with the SALSA (Heineman et al., 2023) typology as an example.

the spans that summarizes a given document. With a goal of understanding where and how text generation succeeds or fails, fine-grained text evaluation selects spans that are either quality or error in generated text. These selected spans are then annotated following a complex typology and rated on the severity of errors or efficacy of high-quality content (Freitag et al., 2021; Dou et al., 2022a; Heineman et al., 2023).

3 Fine-Grained Text Evaluation

Thresh formulates fine-grained text evaluation as two components: *span selection* and *span annotation*. During development, users define their annotation typology and interface features using a YAML template (see Sec 4 and Fig 5 for more details). Based on the configuration, Thresh then constructs an annotation interface that integrates both components, as illustrated in Figures 2 and 3.

3.1 Span Selection

Each annotation instance consists of the *source*, *target* and *context*. For example, in open-ended text generation (Zellers et al., 2019), the source is a starting sentence and the target is a model-generated continuation. In text simplification (Xu et al., 2016), the source would be a complex sentence or paragraph, and the target would be the generated simplification. The context holds additional relevant information, such as a prompt instruction,

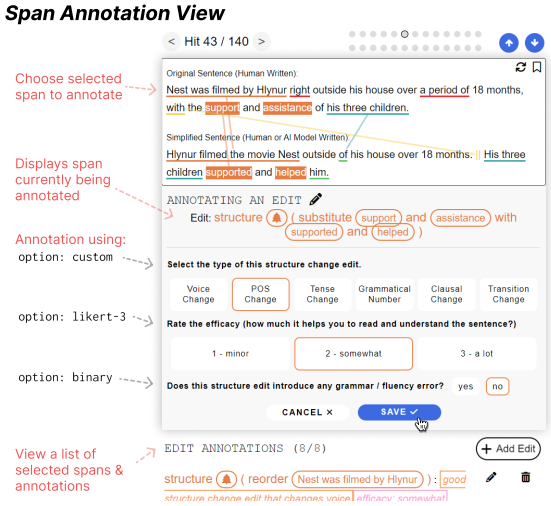

Figure 3: The span annotation component of Thresh, customized with the SALSA (Heineman et al., 2023) typology as an example.

a retrieved Wikipedia page, or a dialogue history. During the span selection stage, annotators select relevant spans, referred to as *Edits*, in the source and target, following the edit category definitions outlined in the typology, as illustrated in Figure 2.

Selection Type. For each edit category, users can specify one of three selection types: *single-span*, *multi-span*, or *composite* – the latter grouping together multiple single-span or multi-span selections. Multi-span selection is well-suited for edits that impact multiple parts of the source or target, e.g., the "Redundant" error in Scarecrow (Dou et al., 2022a), which requires selecting both the repetitive spans and their antecedents. Composite selections are ideal for high-level edits performed as a combination of several low-level edits, e.g., the "Structure" edit in SALSA (Heineman et al., 2023). Users can also customize each edit category to be selectable not only on the target, but also on the source (e.g., "Deletion" edit), or on both (e.g., "Substitution" edit), useful for text revision tasks.

Selection Boundary. Many span-selection interfaces define selection boundaries as each character, which can inadvertently lead to partial word selections and slow the annotation process. Dou et al. (2022a) proposes a solution that "snaps" the selection to the nearest whitespace, but this approach is limited in: (1) punctuation gets selected with adjacent words, even when this is not intended by annotators, (2) languages with no whitespace boundaries between words (e.g., Chinese) cannot be supported and (3) the annotation data cannot be perfectly translated to training data for token-level labeling tasks. We therefore introduce sub-word boundaries as a third option, in which users can use any LLMs tokenizer of their choice (such as `RobertaTokenizer` from Transformers[4]) to tokenize the data and specify a `boundary: subword` flag in the YAML configuration file.

3.2 Span Annotation

In the YAML file, users define the typology in a decision tree structure to further categorize the selected spans into fine-grained types. Unlike previous work which presents all fine-grained edit types to annotators simultaneously, `Thresh` recursively compiles the annotation interface. Annotators thus will answer a series of questions or follow-up questions under each edit type, as shown in Figure 3. This tree structure enables support for complex error typologies. An example of this can be seen in Figure 4, which shows a 35-category typology implementation for a grammar error correction task. `Thresh` supports binary, three and five-scale questions with customized label names, as well as text boxes for tasks that require human post-editing or explanations. With these features, our interface supports complex annotation schemes in a flexible and easily extensible way.

We also give users the option of only enabling one of the two above components. This allows annotation for word/span alignment tasks (Sultan et al., 2014) (where no annotation is needed) or two-stage annotation, where one set of annotators selects spans and then another set labels them.

3.3 Additional Features

Adjudication View. Using the `adjudication` flag, users can deploy two or three interfaces side-by-side, allowing adjudicators to inspect annotators' quality by comparing multiple candidate annotations simultaneously.

Multi-Language Support. Fine-grained evaluation has seen almost exclusive attention to English tasks (Huidrom and Belz, 2022). To smoothen the deployment barrier for multilingual fine-grained evaluation, all interface elements can be overridden to suit any language. For our default interface text, we support 14 translations which can be enabled out-of-the-box by adding a `language` flag: *zh*, *en*, *es*, *hi*, *pt*, *bn*, *ru*, *ja*, *vi*, *tr*, *ko*, *fr* and *ur*.

Instructions. Users may write interface instructions with Markdown formatting, which allows for

[4] https://www.github.com/huggingface/tokenizers

Figure 4: The left figure shows a grammar error typology with 35 categories for contemporary written Zamboangueño Chabacano, a variant of Philippine Creole Spanish (Himoro and Pareja-Lora, 2020). The center figure shows its annotation interface built on `Thresh`, highlighting the ability for `Thresh` to support complex, recursive annotation trees. The right figure shows the Python serialization for the annotation, generated by the `Thresh` library.

links, pictures and inline code. They have the option to display their instructions as a pop-up modal, or prepend the text above the interface.

Paragraph-level Annotation. By breaking evaluation down to individual sentences, authors can reduce the cognitive load required for lengthy annotation tasks such as identifying errors in long-form summarization (Goyal et al., 2022a). Users can specify an additional `context_before` or `context_after` field to add paragraph-level context or custom `display` options to view paragraphs text side-by-side with selected edits.

4 Interactive Interface Builder

To alleviate the time consuming process of customizing and hosting front-end code — even building custom databases in some cases — `Thresh` implements an in-browser interface builder, which allows users to create, test and deploy a fine-grained interface within a single web browser page, as depicted in Figure 5. Users write a YAML template to construct their interface and provide data with a JSON file. The *Compile* button allows users to preview their interface, and the *Deploy* button presents instructions for different deployment options, which are described in §5.

Template Hub. As `Thresh` aims to facilitate easy use and distribution of fine-grained evaluation frameworks, it provides a template hub that makes it simple for any NLP practitioner to access a framework with their own data. Alongside the 10 tutorial templates that explain each interface feature, the annotation builder currently includes 11 widely used inspection and evaluation typologies across major text generation tasks. Table 1 (on Page 2) lists each framework, its associated task and link to our implementation.

To upload a framework to `Thresh`, users can create a GitHub pull request with their typology's YAML file, which is merged publicly. We also include other features to facilitate sharing and replication. Users can add a `citation` flag along with a BibTex citation, which creates a *Cite this Typology* button in the annotation builder, a `paper_link` flag, which adds a link to their research paper in the builder and on deployment, and a `demo_data_link` flag which creates a *View Demo Data* button to allow viewers to use the interface with example data.

For testing, users can paste data into the interface builder interactively, and for deployment can link to data files. Data can be blank or come with existing annotations, in which case the annotations will be appropriately parsed, verified and rendered.

Unified Data Model. As shown in Table 1 on Page 2, many existing frameworks have released their annotated data, but in varied formats. To ensure compatibility, we create conversion scripts that adapt these annotations to our unified format. Our scripts are designed to be *bidirectional*, meaning data published for these typologies can be converted to our format and back without data loss. Our unified fine-grained data format allows smooth transfer of analysis, agreement calculation and modeling code between different projects. We believe this will support research in learning with multi-task fine-grained training setups or model feedback. Like framework templates, users can upload their annotated data to the hub via a GitHub pull request.

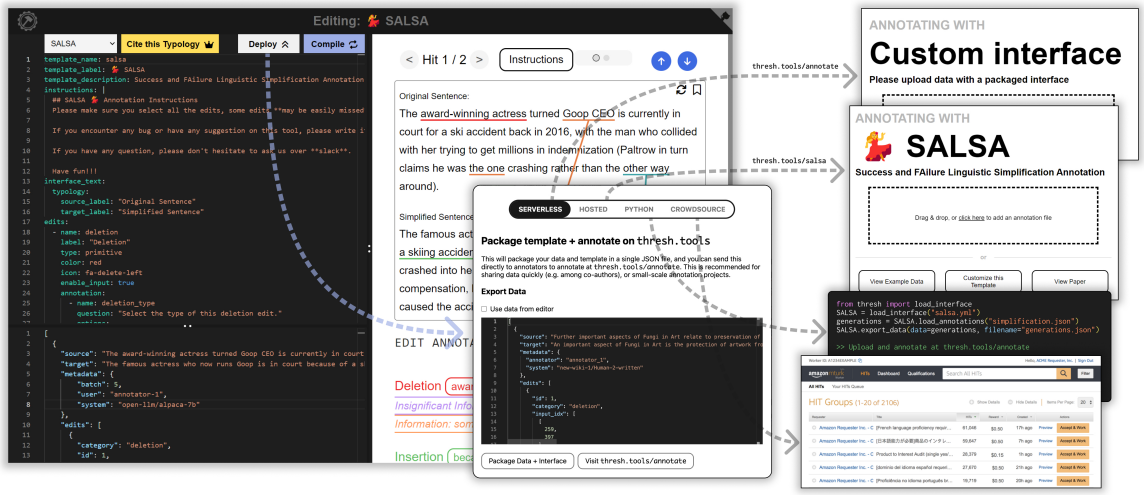

Figure 5: Thresh deployment workflow. Users build and test their template and then deploy with one of 4 options.

5 Deployment

Managing and collecting fine-grained annotations becomes bulky at scale, we thus release supplementary tools to deploy interfaces quickly or programmatically, and integrate loading annotations directly into Python. This includes the thresh library[5], which is useful for compiling interfaces and loading annotations. We support the following deployment types as shown in Figure 5:

- **Hosted:** Best for small-scale inspection or data exploration, users can download a file that bundles the data and template together. Then, users can upload this file to thresh.tools/annotate to begin annotation immediately.
- **Serverless:** Users upload their YAML template to a public repository such as GitHub or HuggingFace, and link their template to thresh.tools through a URL parameter: gh or hf respectively. Users can also link data via the d parameter. In addition, we release demo code for users to host their interface on their own domain without cloning the Thresh repository.
- **Python:** For large scale projects, users can programmatically generate and deploy templates using the create_template functionality provided in the thresh library. This helps for projects with a large number of templates, such as annotation in multiple languages. Additionally, integration with Python allows a direct connection from model generation to annotation processing, supporting the creation of workflows like fine-grained RLHF (Wu et al., 2023).
- **Crowdsource:** If the data collection process is mishandled, annotation by crowdworkers can lead to poorly standardized or noisy data (Karpinska et al., 2021; Veselovsky et al., 2023). To assist annotation quality control, we publish tools to encourage best practices when using crowdsource platforms. Our crowdsource deployment workflow includes example code for interactive, multi-stage tutorials to create qualification tasks and step-by-step tutorials for deployment on both Prolific and Amazon Mechanical Turk.

Additionally, we support lightweight database integration (such as with Google Firebase[6]) for all deployment types, allowing users to connect their own database to any annotation setup.

Python Serialization. Compared to previous work that simply exports JSON annotations, our supplementary thresh library includes functionality for loading and combining annotation files to simplify the data ingestion process. For example, load_annotations merges multiple data files, serializes the data into Python objects, and evaluates whether the data collected is consistent with the configuration used to load the data.

6 Conclusion

We present Thresh 🌾, a unified, customizable, and deployable platform for fine-grained text evaluation. Thresh offers extensive customization via a simple YAML configuration file, and facilitates a community hub for sharing frameworks and annotations. The platform also ensures seamless deployment for any scale of annotation projects and introduces a Python library to further ease the process from typology design to annotation processing.

[5] https://www.pypi.org/project/thresh

[6] https://firebase.google.com

Ethical Considerations

We do not anticipate any ethical issues pertaining to the topics of fine-grained evaluation supported by our interface. Nevertheless, as Thresh lowers the barrier to fine-grained evaluation, vast ethical responsibility falls upon practitioners using our platform to prevent the exploitation of crowdsource workers, through fair pay (Fort et al., 2011) and safeguards against exposure to harmful or unethical content (Shmueli et al., 2021). As task difficulty and complexity scales with the granularity of data collected, increasing care must be taken for training annotators adequately and to scale pay accordingly (Williams et al., 2019).

Acknowledgments

This research is supported in part by the NSF awards IIS-2144493 and IIS-2112633, ODNI and IARPA via the HIATUS program (contract 2022-22072200004). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of NSF, ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7282–7296, Online. Association for Computational Linguistics.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. 2022a. Is GPT-3 text indistinguishable from human text? scarecrow: A framework for scrutinizing machine text. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7250–7274, Dublin, Ireland. Association for Computational Linguistics.

Yao Dou, Chao Jiang, and Wei Xu. 2022b. Improving large-scale paraphrase acquisition and generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9301–9323, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Karën Fort, Gilles Adda, and K. Bretonnel Cohen. 2011. Last words: Amazon Mechanical Turk: Gold mine or coal mine? *Computational Linguistics*, 37(2):413–420.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022a. FALTE: A toolkit for fine-grained annotation for long text evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 351–358, Abu Dhabi, UAE. Association for Computational Linguistics.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022b. SNaC: Coherence error detection for narrative summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 444–463, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

David Heineman, Yao Dou, Mounica Maddela, and Wei Xu. 2023. Dancing between success and failure: Edit-level simplification evaluation using SALSA. *arXiv preprint arXiv:2305.14458*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt.

2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Marcelo Yuji Himoro and Antonio Pareja-Lora. 2020. Towards a spell checker for Zamboanga Chavacano orthography. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 2685–2697, Marseille, France. European Language Resources Association.

Rudali Huidrom and Anya Belz. 2022. A survey of recent error annotation schemes for automatically generated text. In *Proceedings of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 383–398, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Chao Jiang, Wei Xu, and Samuel Stevens. 2022. arXivEdits: Understanding the human revision process in scientific writing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9420–9435, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Marzena Karpinska, Nader Akoury, and Mohit Iyyer. 2021. The perils of using Mechanical Turk to evaluate open-ended text generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1265–1285, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wuwei Lan, Chao Jiang, and Wei Xu. 2021. Neural semi-Markov CRF for monolingual word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6815–6828, Online. Association for Computational Linguistics.

Mounica Maddela, Yao Dou, David Heineman, and Wei Xu. 2023. LENS: A learnable evaluation metric for text simplification. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Ines Montani and Matthew Honnibal. 2018. 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.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4812–4829, Online. Association for Computational Linguistics.

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.

Tal Perry. 2021. LightTag: Text annotation platform. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 20–27, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Boaz Shmueli, Jan Fell, Soumya Ray, and Lun-Wei Ku. 2021. Beyond fair pay: Ethical implications of NLP crowdsourcing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3758–3769, Online. 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.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment:

Exploiting word similarity and contextual evidence. *Transactions of the Association for Computational Linguistics*, 2:219–230.

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*, pages 142–147.

Veniamin Veselovsky, Manoel Horta Ribeiro, and Robert West. 2023. Artificial artificial artificial intelligence: Crowd workers widely use large language models for text production tasks. *arXiv preprint arXiv:2306.07899*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Alex C Williams, Gloria Mark, Kristy Milland, Edward Lank, and Edith Law. 2019. The perpetual work life of crowdworkers: How tooling practices increase fragmentation in crowdwork. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–28.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *arXiv preprint arXiv:2306.01693*.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

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.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

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., 2022b)	Narrative Summarization	✓	thresh.tools/snac
Scarecrow (Dou et al., 2022a)	Open-ended Generation	✓	thresh.tools/scarecrow
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/fg-rlhf
Inspection			
MultiPIT (Dou et al., 2022b)	Paraphrase Generation	✗	thresh.tools/multipit
CWZCC (Himoro and Pareja-Lora, 2020)	Zamboanga 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

Table 2: Existing typologies implemented on Thresh with their associated link. *Released* indicates whether the annotated data is released.