FALTE: A Toolkit for Fine-grained Annotation for Long Text Evaluation

Tanya Goyal[1] **Junyi Jessy Li**[2] **Greg Durrett**[1]
[1] Department of Computer Science [2] Department of Linguistics
The University of Texas at Austin
tanyagoyal@utexas.edu

Abstract

A growing swath of NLP research is tackling problems related to generating long text, including tasks such as open-ended story generation, summarization, dialogue, and more. However, we currently lack appropriate tools to evaluate these long outputs of generation models: classic automatic metrics such as ROUGE have been shown to perform poorly, and newer learned metrics do not necessarily work well for all tasks and domains of text. Human rating and error analysis remain a crucial component for any evaluation of long text generation. In this paper, we introduce FALTE, a web-based annotation toolkit designed to streamline such evaluations. Our tool allows researchers to collect fine-grained judgments of text quality from crowdworkers using an error taxonomy specific to the downstream task. Using the task interface, annotators can select and assign error labels to text span selections in an incremental paragraph-level annotation workflow. The latter functionality is designed to simplify the document-level task into smaller units and reduce cognitive load on the annotators. Our tool has previously been used to run a large-scale annotation study that evaluates the coherence of long generated summaries, demonstrating its utility.

1 Introduction

Recent years have seen a significant improvement in the generation capabilities of large language models (Lewis et al., 2020; Zhang et al., 2020; Brown et al., 2020), across tasks such as machine translation, open-ended story generation, summarization, and others. As these models generate extremely fluent and human-like text, their errors are more subtle than those of prior models and harder to detect (Clark et al., 2021). Overlap-based automatic metrics such as ROUGE (Lin, 2004), BLEU (Papineni et al., 2002), and BERTScore (Zhang et al., 2019) have historically been the most popular metrics used to evaluate the outputs of such generation models. However, recent work has shown that these are unreliable measures of quality (Fabbri et al., 2021), especially for longer text and more open-ended tasks that can have multiple reasonable answers (Wang et al., 2022).

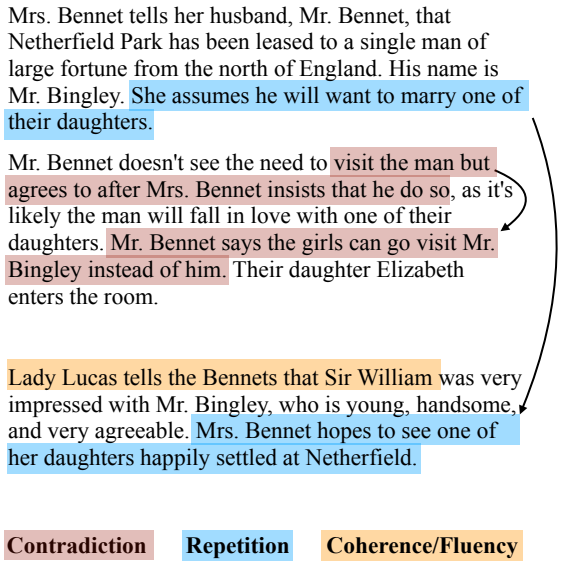

Figure 1: An example of fine-grained annotation using FALTE. A single annotation consists of a text span highlighted by a crowdworker and an error category attached to it. FALTE allows task designers to define their own error taxonomy. Error categories in this taxonomy, e.g. contradiction and repetition in the above example, can be modified to require two associated text spans.

For generation tasks, human evaluation of outputs is generally considered to be the gold standard. Such evaluations are primarily conducted using untrained annotators recruited through crowdsourcing platforms like Mechanical Turk[1] and Upwork.[2] However, even deploying human evaluators for a task is not a straightforward solution. Recent work (Karpinska et al., 2021; Clark et al., 2021) showed that untrained human crowdworkers fail to reliably

[1] https://www.mturk.com/
[2] https://www.upwork.com

distinguish human-written and model-written outputs for strong language models like GPT-3 (Brown et al., 2020), focusing on task designs where annotators are asked to evaluate the generated outputs holistically. To address these limitations, Dou et al. (2022) recommend **fine-grained evaluation** of text quality that is more successful at eliciting quality annotations from untrained crowdworkers. Instead of holistically rating the quality of the whole generated output, they instead ask annotators to identify text spans that correspond to errors from a predefined taxonomy. Moreover, human evaluation conducted using such fine-grained annotations also provides insights into error distributions of current models and reveals avenues for improvement. Figure 1 shows an example of fine-grained annotations at the span-level collected using our tool - FALTE; this is similar to the prior work.

In this paper, we are interested in the evaluation of long text. Human evaluation practices from short text evaluation studies (e.g. paragraph-level text from (Dou et al., 2022)), are not feasible for long text evaluation scenarios (Akoury et al., 2020). In fact, a majority of recent work on long text evaluation, e.g. evaluation of long generated summaries (Mao et al., 2022; Zhang et al., 2022), does not conduct any human evaluation, possibly due to the the difficulty in getting high-quality annotations from crowd workers for long texts. In this paper, we introduce FALTE (**F**ine-grained **A**nnotation for **L**ong **T**ext **E**valuation), a web-based annotation tool to address this gap. Our tool is designed to allow for fine-grained evaluation while also simplifying the long text annotation task through the design of the UI and task workflow. To achieve these goals, FALTE is centered around the following main functionality:

- **Fine-grained annotations:** Prior generation quality evaluation has primarily focused on collecting document-level labels along multiple dimensions, such as fluency, grammar, etc. In this work, we ask annotators to select specific text spans that exhibit errors in a particular error taxonomy, allowing for a more nuanced evaluation of model errors.

- **Decomposing the document-level task into smaller sub-tasks:** FALTE decomposes the overall document-level task into paragraph-level annotation tasks to simplify the user interface and reduce the cognitive load for the crowd workers. This is motivated by prior research in crowdsourcing (Mayer and Moreno, 2003; Kapelner and Chandler, 2010; Hauser et al., 2019) that shows that incrementally introducing texts motivate workers to pay more attention to all units of the task text. We allow task designers to set this granularity of annotation. FALTE also provides tools to crowd workers to easily navigate back and forth between paragraphs.

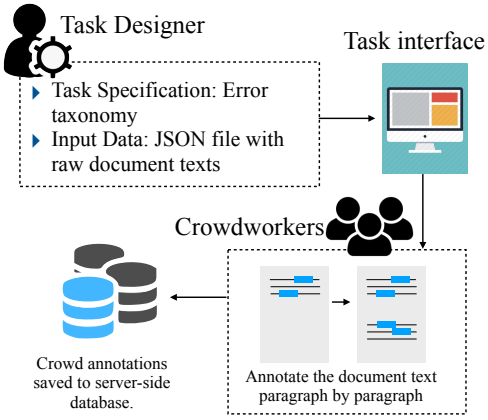

Figure 2: Overall Workflow

- **Flexibility over error definitions:** Finally, we refrain from setting a fixed error taxonomy to support annotation studies along different quality dimensions. Even within the same dimension, researchers may wish to opt for a different error taxonomy across different languages or datasets. Therefore, we allow them to define their own set of categories for annotation that aligns with their specific use case. We also allow them to pre-define if each error category must be associated with one or two text spans during annotation.

Figure 1 shows an example of fine-grained annotations that can be collected using FALTE. Each error in the collected annotations is associated with a corresponding text span; this helps in pinpointing exactly where the error occurred and support further downstream analysis. Note that each error in the error taxonomy can additionally be modified to require two associated text spans, to support error categories like repetition and contradiction.

2 FALTE Tool Description

FALTE is a web-based annotation toolkit designed to help researchers run large-scale annotation stud-

ies evaluating the quality of model generated text. The dataset collected through FALTE has the following form: annotators highlight errors $e \in E$ in a given document d where each annotation e consists of a text span $t \in d$ and corresponding error category c. Annotators incrementally proceed through the document and annotate as many errors as they can identify in the text. In this section, we will outline the workflow for the two stakeholders: 1) **task designers (researchers)** that employ the tool to run these studies, and 2) **crowd annotators** that interact with our web interface and provide annotations for the quality of long text. The overall workflow is outlined in Figure 2.

2.1 Task Designer Workflow

FALTE is designed to be flexible across different long document annotation tasks, for both data collection and evaluation use cases. Here, we describe the different action items for the task designers to create the annotation website and launch a study.

Define the error taxonomy In order to support annotation studies across a variety of quality dimensions and tasks, we allow task designers to define their own error taxonomy. FALTE supports an arbitrary number of error classes.

For each category defined, task designers additionally classify these as either *singleton* or *paired*. For the *singleton* errors, each crowd annotation corresponds to a pair of text span and an error category (as shown in Figure 1). For *paired*, each annotation is a tuple containing two text spans and the error category. This latter functionality was introduced to cover error types such as *repetition errors* which are naturally defined between pairs of text spans. Note that although the tool does not support error annotations with more than two text spans, these use cases can be tackled within FALTE's framework by creating chains of paired error annotations.

To specify their error categories and their specifications, researchers simply edit a configuration file. The contents of these configuration files are then reflected on the task interface when the web application is launched.

Define the annotation granularity Our annotation tool is aimed at collecting annotations for long machine generated text, say ~30-40 sentences or longer, although it can be deployed in shorter settings as well. To simplify this document level annotation task, we allow task designers to decompose the annotation workflow into iterative paragraph level annotation tasks. Note that paragraph here can be defined to correspond to segments of text of any length. In this paper and the demo, we use the terms segment and paragraph interchangeably.

Under this setting, initially the crowd annotators are only shown the first paragraph (or other specified unit of text) for annotation. Once annotation for that paragraph is completed, workers can proceed to subsequent paragraphs with the option to navigate back. We will discuss this in more detail in Section 2.2.

The FALTE tool expects a JSON file with containing all of the text to be annotated as input. Each document in this JSON is represented by a List of Strings; each list item represents a paragraph. Therefore, task designers can control the length of the annotation unit (paragraph) for each document individually through this input file.

Setting up and launching the crowd annotation study Our tool accepts the above task specifications and generates all relevant client and server side code. The task designer can then launch their annotation website using their preferred cloud platform; we host our example demo website using Heroku[3] at `https://coherence-annotation-summaries.herokuapp.com/id=oai1`. After the task annotation website is live, task designers can recruit crowd annotators and collect annotations for text quality.

2.2 Crowd Annotator Workflow

As previously mentioned, each worker starts the annotation process with the first paragraph of the text and incrementally annotates succeeding paragraphs.

Figure 3 shows the task interface and outlines the annotation steps for annotating a single text span in the *Current Paragraph* box, after annotations have already been completed for paragraphs in the *Context* box. The stepwise workflow is:

1. **Highlight span containing error:** First, the worker selects the text span in the *Current Paragraph* containing the error using the click-and-drag motion. The highlighted text will automatically populate in the relevant text box at the bottom.

2. **Choose error type:** Next, the worker chooses an error type or label for the highlighted text.

[3] `https://www.heroku.com/`

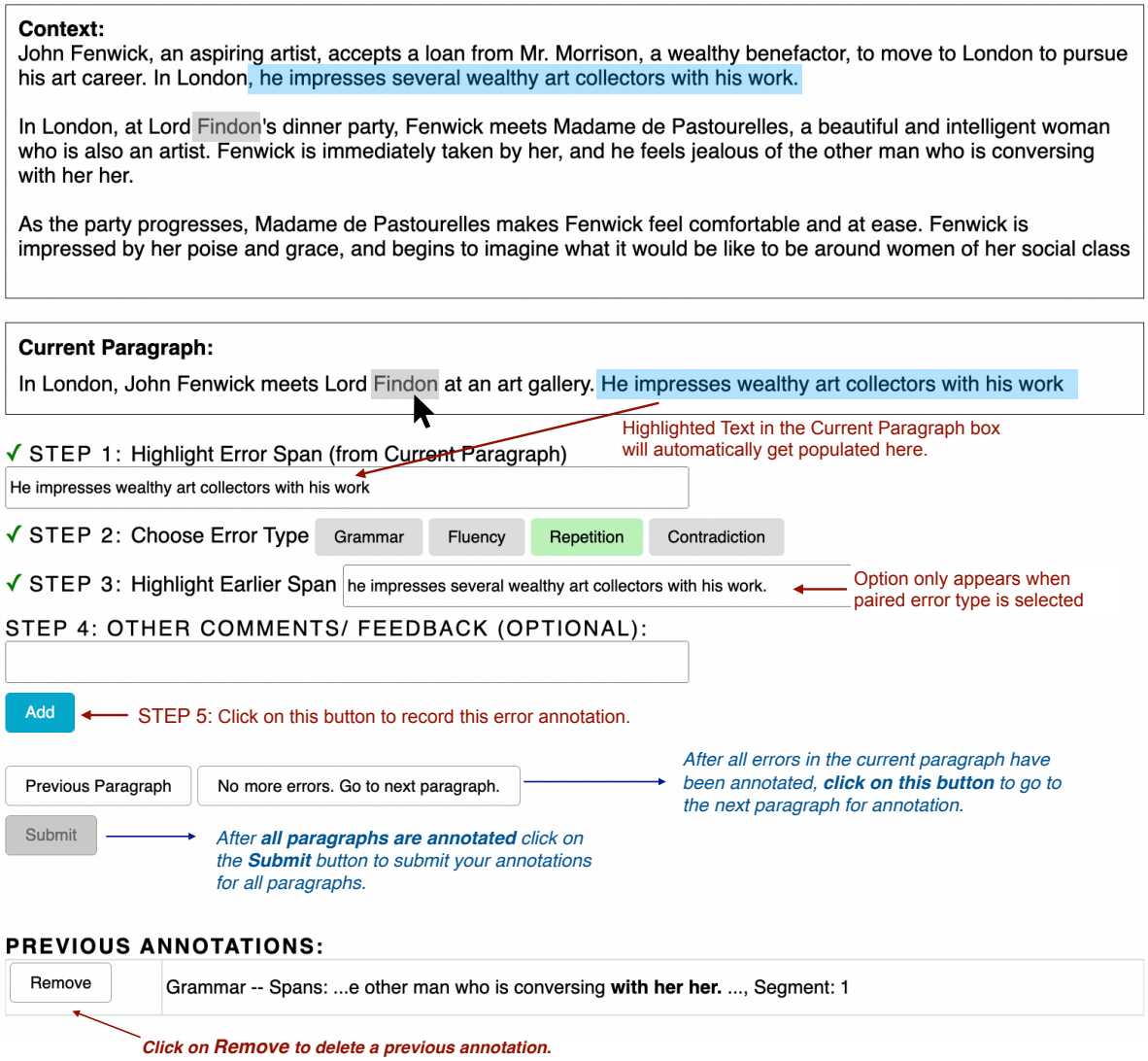

Figure 3: Annotation interface and workflow for a crowd annotator to annotate a single text span. The interface displays the previously annotated paragraphs in the *Context* box. The previous annotations are displayed at the bottom of the web page. At this stage of the task, the crowdworker annotates errors in the *Current Paragraph*. To aid their annotation, crowdworkers can hover over named entities to highlight other instances of the entity (in gray).

Note that Figure 3 currently shows placeholder error types; the actual task interface will list the error types defined by the task designer in the previous section.

3. **Highlight paired text:** If the error type selected in Step 2 is of type *paired*, the worker is further asked to select an additional text span. For our *repetition* error type, this additional span would be the first occurrence of the repeated information (as shown in Figure 3). The additional text span can be selected from either the *Context* box or the *Current Paragraph* box. This step is skipped if the worker selects a *singleton* error .

4. **Optional comments:** The annotation interface also provides the option of providing any additional free-text commentary for the annotated text span(s) and category tuple. In our experiments, we noticed that crowd workers tended to primarily use this option to convey their confidence about that particular span's annotation.

5. **Add annotation:** Finally, the worker clicks on the *Add* button to save error annotation. This will be instantly reflected in the *Previous Annotations* section at the bottom of the page.

The above procedure is repeated to annotate all errors in the *Current Paragraph*, after which the

worker clicks on the *No more errors. Go to next paragraph* button to update both the *Context* and *Current Paragraph* boxes; the *Current Paragraph* will now reflect the next paragraph of the document text. Finally, after errors in all paragraphs of the document have been annotated, the worker clicks on the *Submit* button that stores their error annotations for the whole document in the server-side database.

Coreference Cues Since the annotation interface is designed for long documents, we found that being able to quickly find instances of entities was very helpful during our pilot studies. The scope of this functionality can be controlled by the researchers: in our pilot study (discussed in Section 3), this was only enabled for named entities using string match on names. Specifically, if the user hovers their cursor over any named entity mention in the *Current Paragraph* section, the tool also highlighted all other instances of that entity. Putting the cursor over *Findon* in the *Current Paragraph* highlights the other instance of the same character in Figure 3. It is equivalent to searching (CTRL+F) for the entity, but saves keystrokes.

Making Revisions Workers may make mistakes during the annotation process or simply change their mind about previously annotated errors. We provide tools to address this in the FALTE interface. The bottom of the web page displays the *Previous Annotations* section that lists all prior error annotations by the crowd worker. The worker can remove erroneously annotated error tuples from this table using the *Remove* buttons corresponding to each annotation.

Navigation Flexibility Furthermore, the interface also provides flexibility in navigating between the different paragraphs of the document text; workers can use the *Previous Paragraph* button to go back and annotate any missed errors in text spans.

This final annotation workflow and these additional functionalities were designed based on worker feedback in pilot studies. For example, the pilot study asked annotators to select the error type (step 2) before highlighting the text span (step 1). This order was reversed based on the preference of multiple crowdworkers.

2.3 Output Data

All error annotations displayed in the *Previous Annotation* section get stored on a database when the crowdworker clicks on the *Submit* button.

Each row in the database table corresponds to a single error annotation. Our tool logs in the following fields: Document ID (string), Paragraph ID (int), Text Span (string), Span Start Index (int), Span End Index (int), Error Category (string), Paired Text (string), Optional Comments (string). The latter two are optional fields that may be empty for some errors. We log the span start and end indices in addition to the text to disambiguate between multiple instances of the same text. Additionally, FALTE generates a unique session ID for each (document, annotator) pair to distinguish between the annotations of different crowdworkers for the same document.

2.4 System Implementation

FALTE is a web-based annotation framework that has been tested with Google Chrome, Mozilla Firefox, and Safari browsers. Currently, the crowd annotation is only supported on desktop browsers and not on mobile or other touchscreen devices. The client side interface is made using HTML5, CSS, and JavaScript. The sever side uses the Python-based web framework Flask[4] and the PostgresSQL database[5] to manage and store user annotations, both of which are open-sourced. Our example demo website is hosted here: https://coherence-annotation-summaries.herokuapp.com/id=oai1.[6]

3 Use Case

FALTE has been used to collect crowd annotations for one long document evaluation task: coherence evaluation of long model-generated summaries (Goyal et al., 2022). The annotation study was conducted for narrative summaries of books and movies. The study defined 7 different errors across two categories (1) coherence, and (2) language and fluency errors.

Table 1 outlines the statistics of the annotated dataset in the coherence study. The study was run for 160 summary documents. Each summary was an average of 36 sentences long, which is approximately 12 times the length of the most common use case in summarization research, that of evaluating news summaries. For each document, annotations

[4] https://flask.palletsprojects.com/

[5] https://www.postgresql.org/

[6] See https://github.com/tagoyal/falte-tool for the implementation of the tool.

Data Statistic	Count
Documents Annotated	160
Average Length (words)	480
Average Length (sentences)	36
Error Annotations Collected	9.6K
Crowd Annotators	12
Error Categories	7
Singleton	5
Paired	2

Table 1: Statistics for texts evaluated in the coherence evaluation study conducted using FALTE.

were collected from 3 crowd annotators, totalling 9.6K error annotations across all summaries.

The study showed that: (1) The strategy of fine-grained annotation is better suited for long documents compared to document-level annotation. The study showed that the inter-annotator agreement of document-level labels for long texts is 0.19 compared to 0.48 reported for news summaries that are considerably shorter (Fabbri et al., 2021). (2) FALTE can be used to run a large-scale annotation study. In fact, the paper (Goyal et al., 2022) shows that the collected annotations are high quality and can be used to train a strong classifier for automatically identifying coherence errors in text.

The results also outline other benefits of fine-grained annotation. Different annotators have different criterion for judging the overall quality of text. The task design decision of explicitly breaking it down through a taxonomy and prompting for rationales, i.e. the text spans, provides insight into which errors types are more critical for each annotator. Note that due to the nature of the task design (identifying **all** error spans in a document), we saw that annotators tend to be high precision low recall (Dou et al., 2022; Goyal et al., 2022), i.e. they rarely highlight non-error spans, but tend to miss error spans. Devising techniques that can improve recall for such task designs is a promising research question that we leave for future work.

4 Related Work

Reference-based evaluation is the most popular evaluation paradigm for generation models. These include overlap-based metrics (Lin, 2004; Papineni et al., 2002; Banerjee and Lavie, 2005), or distributional similarity metrics (Zhang et al., 2019; Kusner et al., 2015), and others. However, recent work has shown that these do not correlate with human judgments of quality (Dhingra et al., 2019; Kryściński et al., 2019; Fabbri et al., 2021).

Human evaluation of generation quality is generally considered to be more reliable, although there do not exist any fixed protocols for conducting these studies (Celikyilmaz et al., 2020). In recent work, both Likert scale rating and A/B testing based evaluation frameworks have been widely used (Celikyilmaz et al., 2020; Clark et al., 2021). However, across both these frameworks, tasks are generally designed to elicit document-level quality judgments from crowdworkers that are insufficient to measure the quality of generated text (Clark et al., 2021; Karpinska et al., 2021; Gehrmann et al., 2022). Particularly, Clark et al. (2021) show that crowd annotators often conflate multiple dimensions of quality, and tend to primarily focus on surface properties like grammaticality while evaluating summaries. Therefore, in our task design, we focus on fine-grained error annotations that allow annotators to clearly distinguish between the different error categories and their occurrences.

The document-level task design of the prior work discussed above is quite straightforward to set up using the basic UI components provided by crowdsourcing platforms such as Mechanical Turk. However, creating a user-friendly interface for fine-grained annotation collection is much more challenging. Dou et al. (2022) create a task interface for fine-grained annotations of short generated text. In contrast, our iterative tool design is motivated by prior crowdsourcing research (Kapelner and Chandler, 2010; Hauser et al., 2019) that shows that worker performance and attention increases with an incremental task design for longer tasks. Moreover, decomposition into smaller paragraph-level annotations also reduces the cognitive load on the annotator (Mayer and Moreno, 2003; Brosnan et al., 2021).

5 Conclusion

We present FALTE, a web-based annotation tool to collect fine-grained error annotations for text. It provides an easy-to-use interface to annotate and submit fine-grained annotations and is equipped with capabilities such as navigational flexibility and coreference highlighting that are specifically designed for better user experience while annotating long documents. On the task designer side, our tool is highly customizable: task designers can define their own error taxonomy, error category specifications, and annotation granularity. Therefore,

it can accommodate a wide variety of evaluation objectives, e.g. different dimensions of quality like coherence or factuality, language or task-specific taxonomies, and more. We hope that FALTE can support the design and launch of fine-grained human evaluation studies in the future.

Acknowledgments

This project was partially supported by Good Systems,[7] a UT Austin Grand Challenge to develop responsible AI technologies, a grant from the UT Austin Office of the Vice President for Research through the "Creating Connections for National Security Research Grants" program, a grant from Open Philanthropy, NSF grants IIS-2107524, IIS-2145479, and gifts from Salesforce, Amazon, and Adobe.

References

Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. 2020. STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6470–6484.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Kylie Brosnan, Bettina Grün, and Sara Dolnicar. 2021. Cognitive load reduction strategies in questionnaire design. *International Journal of Market Research*, 63(2):125–133.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

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.

[7] https://goodsystems.utexas.edu/

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah Smith, and Yejin Choi. 2022. 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.

Alexander Richard Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2022. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *arXiv preprint arXiv:2202.06935*.

Tanya Goyal, Junyi Jessy Li, and Greg Durrett. 2022. SNaC: Coherence error detection for narrative summarization. *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

David Hauser, Gabriele Paolacci, and Jesse Chandler. 2019. Common concerns with mturk as a participant pool: Evidence and solutions. In *Handbook of research methods in consumer psychology*, pages 319–337. Routledge.

Adam Kapelner and Dana Chandler. 2010. Preventing satisficing in online surveys. *Proceedings of CrowdConf*.

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.

Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. 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 540–551.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer.

2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Ziming Mao, Chen Henry Wu, Ansong Ni, Yusen Zhang, Rui Zhang, Tao Yu, Budhaditya Deb, Chenguang Zhu, Ahmed Awadallah, and Dragomir Radev. 2022. DYLE: Dynamic Latent Extraction for Abstractive Long-Input Summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1687–1698.

Richard E Mayer and Roxana Moreno. 2003. Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist*, 38(1):43–52.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alex Wang, Richard Yuanzhe Pang, Angelica Chen, Jason Phang, and Samuel R Bowman. 2022. SQuALITY: Building a Long-Document Summarization Dataset the Hard Way. *arXiv preprint arXiv:2205.11465*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. 2022. Summ^n: A multi-stage summarization framework for long input dialogues and documents. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604, Dublin, Ireland. Association for Computational Linguistics.