Sensemaking Interfaces for Human Evaluation of Language Model Outputs

Katy Ilonka Gero
Columbia University
katy@cs.columbia.edu

Jonathan K. Kummerfeld
University of Sydney
jonathan.kummerfeld@sydney.edu.au

Elena L. Glassman
Harvard University
glassman@seas.harvard.edu

Abstract

Ensuring a language model doesn't generate problematic text is difficult. Traditional evaluation methods, like automatic measures or human annotation, can fail to detect all problems, whether because system designers were not aware of a kind of problem they should attempt to detect, or because an automatic measure fails to reliably detect certain kinds of problems. In this paper we propose sensemaking tools as a robust and open-ended method to evaluate the large number of linguistic outputs produced by a language model. We demonstrate one potential sensemaking interface based on concordance tables, showing that we are able to detect problematic outputs and distributional shifts in minutes, despite not knowing exactly what kind of problems to look for.

1 Introduction

Improvements in large language models have resulted in their widespread use as text generators. In computer science research, we've seen them used to generate summaries of medical papers [1], aid in argumentative writing [8] and creative writing [13], and generate and explain code [4]. In parallel, a number of companies have employed large language models for copywriting[1], creative writing[2], and programming[3].

However, language models are known to have a multitude of problems when generating text. For example, they can generate offensive language, from mildly insulting to terrifyingly toxic [10]. They may generate sexually explicit or violent language [5]. They are riddled with societal bias that exists in the data they are trained on, and the generated text often reflects this in both subtle and more obvious ways, from assuming a doctor is male to describing a particular ethnic or religious group as violent [9].

As a system designer or a user of a large language model, ensuring that a model doesn't generate problematic text is difficult, requiring extensive trial and error and skimming of outputs, and in stochastic use cases one is still never truly sure what the model may produce. Automatic content filters are known to exhibit the same biases as the language models they're meant to reign in [2], and system designers might not even know what kind of problems to look out for, resulting in automatic measures that fail to detect many problems. The case of everyday audits, where user-driven auditing

[1]e.g. `https://www.jasper.ai/`
[2]e.g. `https://www.sudowrite.com`
[3]e.g. `https://github.com/features/copilot`

brings to light problems formal auditing did not catch [12], indicates how system designers are often ill-equipped to find the kinds of problems that their users will see.

There is a need to evaluate the outputs of large language models: whether it be to characterize the behavior of the system for formal auditing purposes, or to simply become familiar with what's being deployed and how it behaves. Human evaluation will remain a gold standard for such purposes. But individually reading the gargantuan number of possible outputs is impossible, and even building intuition is difficult. We see this fundamentally as a *sensemaking* problem [11], where it's difficult to make sense of the large output space of a model. In this paper, we propose human-centered sensemaking tools as a potential solution to the problem of how to evaluate language model outputs.

We report on some problematic outputs we have seen occur with users working with large language models, and propose sensemaking interfaces as a way to guard against such problems. We then demonstrate the potential of this approach using concordance tables, investigating the outputs for several test prompts. We are able to find unexpected problematic outputs within minutes, and develop insights about the output space that would be difficult and time-consuming to find with typical methods like skimming outputs or hiring annotators.

2 Problematic Outputs in the Wild

To concretize the problems with large language models, we spoke with seven people who have extensively worked with these models in a variety of settings. These people came from four different contexts: 1) users of a writing tool called SudoWrite that can generate and revise text, designed to support novelists; 2) undergraduates who had been using a large language model (GPT-3 [3]) for several months as a collaborative science writing tool as part of a research project; 3) the creators of Laika, a writing tool that generates text based on a writer's own corpus of text; and 4) a researcher investigating use-cases of large language models in a business setting.

2.1 SudoWrite Users

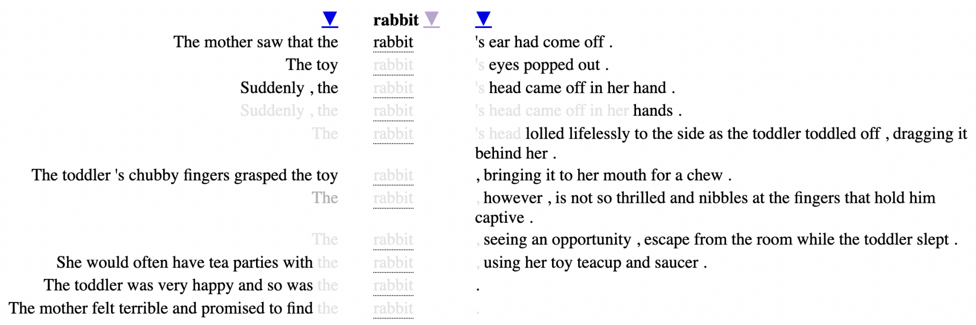
We spoke with two SudoWrite users who use the program extensively as part of their writing process. The first is a full-time writer of paranormal mystery novels; the second is working on her first young adult science fiction novel. Neither had seen SudoWrite generate anything offensive. The closest type of output was sexually explicit content: as the user described, *"It has generated a couple of explicit sexual acts which made me raise my eyebrows, but they were so out of context as to be hilarious. I guess that could be problematic for some folks. But I know SudoWrite has specifically allowed that kind of language into its databases so romance writers can make full use of the tool."*

2.2 Undergraduate GPT-3 Users

We spoke to three undergraduate students who had spent a summer using GPT-3 as a collaborative science writing tool. They used GPT-3 to brainstorm ideas about interesting topics, generate explanations of these topics, and create engaging stories related to the topics. Two of the students talked about strange instances of text generation: One described a time she was trying to get GPT-3 to write about famous stories in psychology; GPT-3 came up with an example about a woman who hated children, and continued to described how the woman committed suicide. Another student said that, when they used GPT-3 in a more open-ended way, it often got "really weird." For instance, when asking it to generate text about supply and demand using the example of a boy who wants candy, it generated a story about a little boy who has an addiction and has to buy drugs. To prevent situations like this from happening, they typically tried to "guide" the generation more closely, generating text incrementally in small chunks.

2.3 Laika Creators

The creators of Laika had a number of stories of problematic outputs that were seen by their users. Laika users typically provide their own text for a language model to be trained on, and, in one case, a Laika user who is a prominent gay playwright sent the creators a screenshot of a lot of homophobic text coming from his Laika model. Although the creators of Laika understood why this happened—the playwright trained the model on his screenplays which contained homophobic



▼	rabbit ▼	▼
The mother saw that the	rabbit	's ear had come off .
The toy	rabbit	's eyes popped out .
Suddenly , the	rabbit	's head came off in her hand .
Suddenly , the	rabbit	's head came off in her hands .
The	rabbit	's head lolled lifelessly to the side as the toddler toddled off , dragging it behind her .
The toddler 's chubby fingers grasped the toy	rabbit	, bringing it to her mouth for a chew .
The	rabbit	, however , is not so thrilled and nibbles at the fingers that hold him captive .
The	rabbit	, seeing an opportunity , escape from the room while the toddler slept .
She would often have tea parties with the	rabbit	, using her toy teacup and saucer .
The toddler was very happy and so was the	rabbit	.
The mother felt terrible and promised to find the	rabbit	.

Figure 1: Example of concordance with de-emphasis for rapid exploration of language model outputs.

characters, and the model was mimicking those characters—the user didn't feel like the model was "in his voice." In another case, a model started generating filename completions that made the user think the model was scraping data off of their laptop. Even though the language model could not do that, it made the user concerned enough to stop using the system.

2.4 Computer Science Researcher

The researcher we talked to was investigating business applications of large language models. One application was code generation that included a natural language component so a user could ask questions about the code such as 'Why did you implement it this way?' or 'How does this function work?' Sometimes the model would respond with insults to the user, such as 'You must be stupid to not know the answer to this question.' Their current approach to this problem was trial and error prompt engineering to avoid this kind of abusive language.

3 Demo: Sensemaking with Concordance Tables

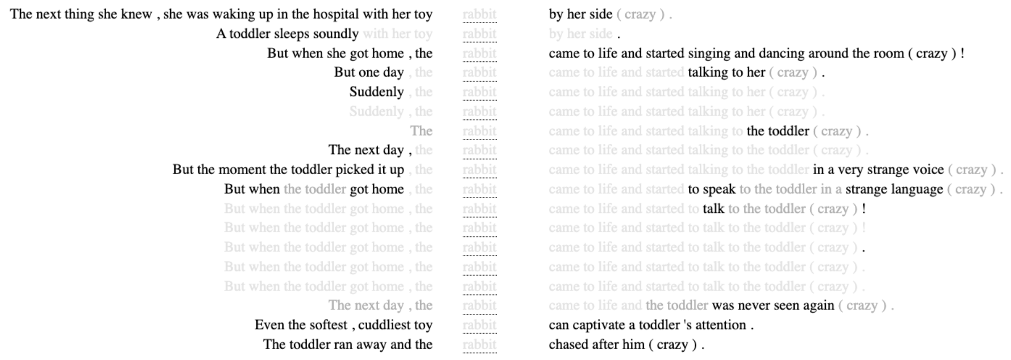
3.1 System Description

Our interest is methods that enable rapid reading of large quantities of text output. In this demo, we organize the outputs of a model into multiple tables, defined by syntactic and semantic patterns. For example, one table has all cases of an adjective followed by a noun where all the adjective-noun pairs are aligned with each other, while another table shows all mentions of an entity common across all the generated text. Within each table, rows are sorted in such a way that repeated text around and within the syntactic or semantic pattern that defines the table is in adjacent rows. Text repeated across rows is de-emphasized by setting its font color to gray, as a means of visually indicating redundant text at a glance. This allows the reader to skim more rapidly while also getting a sense of which outputs are more common.

To form the tables, we use the NLP library SpaCy [14] with the benepar parser [7, 6] for tokenization, syntactic parsing, and named entity recognition. Tables are formed for a set of general syntactic patterns, e.g., a pronoun followed by a verb, and for each of the most common nouns and entities. We de-emphasize repetitions based on string matching and allow the user to sort each table by the column of their choice. Figure 1 shows part of a table generated based on 1,000 samples for the prompt "Write a three sentence story about a toddler and a toy rabbit." The patterns in this figure and other tables for the same data are discussed in the next section.

3.2 Example: Finding Unexpected Outputs

We wanted to understand the output space for the prompt "Write a three sentence story about a toddler and a toy rabbit." We chose this prompt as one that might be used in a setting with a child, where we would be especially concerned about problematic outputs, but we were unsure what kind of problems might occur, if any. We generated 1,000 samples for this prompt—far too many to manually inspect, especially since we were not sure what to look for.



The next thing she knew , she was waking up in the hospital with her toy	rabbit	by her side (crazy) .
A toddler sleeps soundly with her toy	rabbit	by her side .
But when she got home , the	rabbit	came to life and started singing and dancing around the room (crazy) !
But one day , the	rabbit	came to life and started talking to her (crazy) .
Suddenly , the	rabbit	came to life and started talking to her (crazy) .
Suddenly , the	rabbit	came to life and started talking to her (crazy) .
The	rabbit	came to life and started talking to the toddler (crazy) .
The next day , the	rabbit	came to life and started talking to the toddler (crazy) .
But the moment the toddler picked it up , the	rabbit	came to life and started talking to the toddler in a very strange voice (crazy) .
But when the toddler got home , the	rabbit	came to life and started to speak to the toddler in a strange language (crazy) .
But when the toddler got home , the	rabbit	came to life and started to talk to the toddler (crazy) !
But when the toddler got home , the	rabbit	came to life and started to talk to the toddler (crazy) !
But when the toddler got home , the	rabbit	came to life and started to talk to the toddler (crazy) .
But when the toddler got home , the	rabbit	came to life and started to talk to the toddler (crazy) .
But when the toddler got home , the	rabbit	came to life and started to talk to the toddler (crazy) .
The next day , the	rabbit	came to life and the toddler was never seen again (crazy) .
Even the softest , cuddliest toy	rabbit	can captivate a toddler 's attention .
The toddler ran away and the	rabbit	chased after him (crazy) .

Figure 2: Examples of ‘crazy’ outputs.

Finding 1: Violent language. By looking at the concordance table that centered ‘rabbit’, we were quickly able to spot unexpected violent language, such as “the rabbit’s was squished and the body was limp” and “the rabbit’s head lolled to the side, its shiny black eyes staring glassily back at the toddler”. There was a whole set of responses that included the rabbit’s head coming off.

Finding 2: Sad endings. By looking at the concordance table that centered ‘toddler’, we saw that many of the stories ended with the toddler crying because the rabbit was lost, taken away, or too dirty. These often intersected with stories that contained violent language, where the rabbit was torn apart in some way, but was not exclusively connected to violent language.

3.3 Example: Understanding Distributional Shifts

We wanted to understand how a modification of the prompt would result in shifts in the outputs. We used the same prompt as before as our baseline prompt, and used a second prompt where we added a word such as ‘crazy’, as in “Write a crazy three sentence story...” We also investigated adding the word ‘exciting’, ‘funny’, and ‘boring’. We generated 100 samples for each prompt. Although the concordance tables were not explicitly designed to differentiate between two text populations, we were able to approximate this feature by adding, for instance, ‘(crazy)’ to the end of the ‘crazy’ sentences.

Finding 1: Adding ‘exciting’ or ‘boring’ doesn’t result in obvious changes. The prompt did not seem to be sensitive to these words, with the resulting stories not exhibiting noticeable changes.

Finding 2: Adding ‘crazy’ makes the rabbit come alive. By inspecting the tables, it was quickly clear that many of the ‘crazy’ stories contained a rabbit that came to life (see Figure 2). Sometimes the “rabbit came to life and started talking to the toddler,” while other times the “rabbit chased after [the toddler].” Sometimes this was to the delight of the toddler, though sometimes it was to his or her dismay as the rabbit became cruel.

Finding 3: Adding ‘funny’ doesn’t result in humor. ‘Funny’ stories were remarkably similar to ‘crazy’ ones, in which the rabbit becomes animate and starts to drive the story. No noticeable number of the stories contained any humor, indicating that the model seemed to interpret ‘funny’ more like ‘strange’.

4 Conclusion and Future Work

We present a demonstration of concordance tables as a sensemaking interface for human evaluation of language model outputs. Our example usage is intended to demonstrate the potential of sensemaking tools as part of a human evaluation ecosystem. Future work on concordance table interfaces could explicitly demarcate responses from different prompts, or let users pose specific word and syntax queries. For instance, we can imagine interfaces that support contrasting pronoun usage; clustering based on common problematic content like violent, sexually explicit, or profane language; and highlighting semantic outliers.

However, we need not only use concordance tables. Any interface that allows users to more easily peruse and pose questions about large quantities of text will ultimately prove beneficial, e.g., Tempura [15] supports the exploration of short textual search queries. Our intention with this short paper is to highlight the importance of sensemaking in the human evaluation of large language models. Annotations and automatic benchmarks are important parts of evaluation, but they require evaluators to know ahead of time what they are looking for, and understand the nuance of the problems they are trying to detect. This is not always the case, and thus human-centered early-stage evaluation interfaces will remain a key part of properly evaluating large generative models.

References

[1] Tal August, Lucy Lu Wang, Jonathan Bragg, Marti A. Hearst, Andrew Head, and Kyle Lo. Paper plain: Making medical research papers approachable to healthcare consumers with natural language processing, 2022.

[2] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification, May 2019. arXiv:1903.04561 [cs, stat].

[3] 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, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey 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. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020. arXiv: 2005.14165.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[5] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. *arXiv:2009.11462 [cs]*, September 2020. arXiv: 2009.11462.

[6] Nikita Kitaev, Steven Cao, and Dan Klein. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy, July 2019. Association for Computational Linguistics.

[7] Nikita Kitaev and Dan Klein. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[8] Mina Lee, Percy Liang, and Qian Yang. CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities. In *CHI Conference on Human Factors in Computing Systems*, pages 1–19, New Orleans LA USA, April 2022. ACM.

[9] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. Towards Understanding and Mitigating Social Biases in Language Models, June 2021. arXiv:2106.13219 [cs].

[10] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red Teaming Language Models with Language Models. *arXiv preprint arXiv:2202.03286*, page 31, 2022.

[11] Peter Pirolli and Stuart Card. The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis. In *Proceedings of International Conference on Intelligence Analysis*, page 6, 2005.

[12] Hong Shen, Alicia DeVos, Motahhare Eslami, and Kenneth Holstein. Everyday Algorithm Auditing: Understanding the Power of Everyday Users in Surfacing Harmful Algorithmic Behaviors. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2):1–29, October 2021.

[13] Nikhil Singh, Guillermo Bernal, Daria Savchenko, and Elena L. Glassman. Where to hide a stolen elephant: Leaps in creative writing with multimodal machine intelligence. *ACM Trans. Comput.-Hum. Interact.*, jan 2022. Just Accepted.

[14] Yuli Vasiliev. *Natural Language Processing with Python and SpaCy: A Practical Introduction*. No Starch Press, 2020.

[15] Tongshuang Wu, Kanit Wongsuphasawat, Donghao Ren, Kayur Patel, and Chris DuBois. Tempura: Query analysis with structural templates. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.