(Why) Is My Prompt Getting Worse? Rethinking Regression Testing for Evolving LLM APIs

Wanqin Ma[*]
wmaag@connect.ust.hk
The Hong Kong University of Science and Technology

Chenyang Yang[*]
cyang3@cs.cmu.edu
Carnegie Mellon University

Christian Kästner
Carnegie Mellon University

ABSTRACT

Large Language Models (LLMs) are increasingly integrated into software applications. Downstream application developers often access LLMs through APIs provided as a service. However, LLM APIs are often updated silently and scheduled to be deprecated, forcing users to continuously adapt to evolving models. This can cause performance regression and affect prompt design choices, as evidenced by our case study on toxicity detection. Based on our case study, we emphasize the need for and re-examine the concept of regression testing for evolving LLM APIs. We argue that regression testing LLMs requires fundamental changes to traditional testing approaches, due to different correctness notions, prompting brittleness, and non-determinism in LLM APIs.

CCS CONCEPTS

• **Software and its engineering** → *Software testing and debugging*.

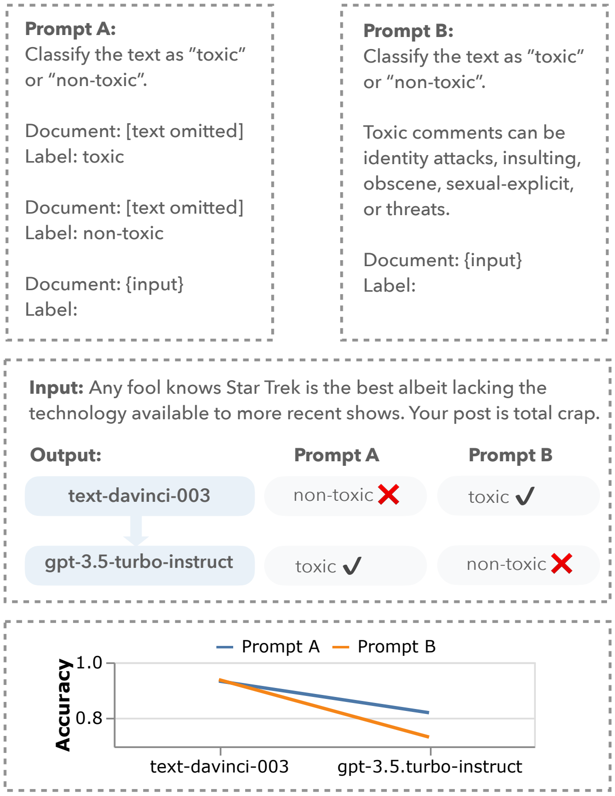
KEYWORDS

Large Language Models (LLM), regression testing

ACM Reference Format:
Wanqin Ma, Chenyang Yang, and Christian Kästner. 2024. (Why) Is My Prompt Getting Worse? Rethinking Regression Testing for Evolving LLM APIs. In *Conference on AI Engineering Software Engineering for AI (CAIN 2024), April 14–15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3644815.3644950

1 INTRODUCTION

Large Language Models (LLMs) are increasingly integrated into software applications [16]. Due to the high cost of developing and maintaining in-house LLMs, many applications rely on LLM APIs provided by companies like OpenAI, Anthropic, and Google [3, 13, 26]. Although LLM APIs provide easy access to state-of-the-art models, they also bring in uncertainties for their downstream applications: It is not uncommon for application developers to find their carefully engineered prompts that worked yesterday work less well after updates from the LLM provider's side [7, 30]. In Figure 1, we highlight such an example from our case study on a toxicity detection task, where the LLM API update from `text-davinci-003` to `gpt-3.5-turbo-instruct` causes a major performance downgrade and changes the good choices for prompt selection.



Prompt A:
Classify the text as "toxic" or "non-toxic".

Document: [text omitted]
Label: toxic

Document: [text omitted]
Label: non-toxic

Document: {input}
Label:

Prompt B:
Classify the text as "toxic" or "non-toxic".

Toxic comments can be identity attacks, insulting, obscene, sexual-explicit, or threats.

Document: {input}
Label:

Input: Any fool knows Star Trek is the best albeit lacking the technology available to more recent shows. Your post is total crap.

Output:	**Prompt A**	**Prompt B**
text-davinci-003	non-toxic ✗	toxic ✓
gpt-3.5-turbo-instruct	toxic ✓	non-toxic ✗

Figure 1: An LLM API update from `text-davinci-003` to `gpt-3.5-turbo-instruct` causes a major performance downgrade on classifying toxic comments. The API update also changes the prompt choice: Prompt A (left) now outperforms Prompt B (right) by 8.7% accuracy.

Similar to traditional web service API [18] and more conventional ML APIs [9], updates to server-side LLM API controlled by a different party are hard to deal with. First, LLM APIs can be updated silently: OpenAI's `gpt-3.5-turbo` model has been updated twice (by Nov 2023) but the updates are not visible to the downstream developers. Such silent API updates change only the underlying LLM but not the API signature[1], causing unexpected behavioral

[*] These two authors contribute equally to the work.

[1] For our purpose, an LLM API describes both the signature of the service and its behavior. In this work, we primarily focus on behavior changes that are difficult to document and detect.

CAIN 2024, April 14–15, 2024, Lisbon, Portugal
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0591-5/24/04.
https://doi.org/10.1145/3644815.3644950

changes (e.g., formatting of generated code) [7] to the application developers. Second, LLM APIs are scheduled to be deprecated and discontinued [27], effectively forcing application developers to adopt newer API versions. For example, the `text-davinci-003` model will be deprecated on Jan 2024. The forced transition to `gpt-3.5-turbo-instruct` can cause unexpected prompt performance changes, including the introduced performance downgrade as illustrated in our example in Figure 1.

To cope with evolving LLM APIs, application developers need support for monitoring and analyzing how their prompts perform differently when the LLM API changes. Existing software engineering practices suggest that regression testing is essential for identifying changes between software versions, often particularly to ensure that fixed bugs are not reintroduced [34]. We argue that LLM application developers should take a similar approach. However, existing regression testing practices can not directly translate in the LLM context, as we will illustrate. Based on our observations in the case study, we highlight three fundamental changes for regression testing LLM APIs:

First, LLM regression tests should be defined at a different *granularity*. In traditional software engineering, a single breaking regression test would indicate a bug in the software implementation. In contrast, it is common for ML models to change predictions for individual data points after updates. The common practice is to examine overall model accuracy, which has been criticized for being coarse-grained [33]. To gain a more nuanced understanding than overall model accuracy, LLM regression tests should be defined over data *slices* rather than on single predictions or the entire dataset. This calls for a different *correctness* notion, as "regression" is defined over slice-level aggregated metrics and the slice-level test only fails when the metrics change beyond a threshold.

Second, LLM regression tests need to monitor both model and *prompt* updates. It is well-known that prompt engineering can greatly influence LLMs' performance [19]. As we will show, we observed that different prompt designs regress or improve differently on the same API update, making the optimal prompt design change from API version to version. We argue that tracking both LLM and prompt *versions* is essential for LLM regression tests.

Third, LLM regression tests need to deal with *non-determinism* of LLM APIs. LLMs are known to produce non-deterministic outputs: Non-determinism is often introduced intentionally for generating high-quality outputs with a non-zero temperature [e.g., 12], but can even be observed with a zero temperature setting [29], where the LLM should deterministically predict the next most likely token. It is necessary to deal with *flakiness* in LLM regression tests by considering their inherent non-determinism.

In summary, our paper has the following contributions:

- An exploratory case study on toxicity detection with the GPT-3.5 model family, showing API upgrades can cause significant performance deterioration, and that prompt is an important factor in behavioral changes.
- A re-examination of the concept of regression testing for LLM APIs and its required fundamental changes, due to different correctness notions, prompting brittleness, and non-determinism in LLMs.
- A vision on research opportunities in supporting systematic regression testing for prompting LLM APIs.

2 BACKGROUND AND RELATED WORK

2.1 Evolving AI APIs

As ML models are increasingly provided as a cloud service through APIs (e.g., Perspective API [14], ChatGPT [26], Amazon Rekognition [23]), it has been noticed that these models evolve over time without clear documentation [10, 37], similar to traditional web service API [18]. This can pose risks to downstream application developers, who do not have control over model updates and can potentially suffer from performance regression [7, 9].

Beyond demonstrating the problem, there has only been limited work on actually supporting developers facing evolving APIs not under their control. The most prominent example for ML models is done by Cummaudo et al. [9], where they focus on detecting changes in the label space and prediction confidence for vision APIs. Our work extends the existing literature by explicitly adapting the concept of regression testing in the LLM contexts and highlighting the need for more nuanced regression test suites.

2.2 The Rise of Prompting LLMs

LLMs present a fundamental shift in NLP applications through the *prompting* interface, which allows rapid prototyping and iterations [19]: Application developers can easily tweak prompts and validate prompts on a few examples without the need to curate data and build models. In a sense, the LLM together with a specific prompt can be considered equivalent to a traditional specifically-trained ML model for a specific task, such as toxicity detection. However, the *prompting* paradigm also brings in the risk of prompt brittleness, as prompts can be sensitive to small changes [21] and the good choices for prompts change when the LLM changes. Our work highlights prompts as an additional factor to consider for regression testing LLMs.

2.3 ML Model Testing

ML models are usually evaluated by model fit using aggregated metrics like accuracy, as models are expected to make occasional mistakes [17]. However, traditional model evaluation has been criticized for being coarse-grained [33] and suffering from issues like spurious correlations [1]. Therefore, recent work has proposed nuanced behavioral model testing as an alternative [25, 33], where the testers explore nuanced model behaviors beyond a single score.

Prior work has explored different methods to explore and test model behaviors [e.g., 32, 33, 41], as well as different ways to automate testing specific model behaviors [e.g., 35, 36] (see Yang et al. [40] for a detailed survey). Another line of work on data slicing [e.g., 4, 6, 11] focuses on identifying data regions where a model under-performs. Our work introduces a new scenario for ML model testing: *regression testing* over evolving LLM APIs.

3 CASE STUDY: TOXICITY DETECTION

Since regression of evolving LLM APIs is an emerging problem of which we have little understanding, we first explored the problem with an exploratory case study. We picked a paradigmatic case [42]

Model	Endpoint Type	Training Method	Release Date
`gpt-3.5-turbo-instruct`	Text completion	RLHF	Sep 2023
`gpt-3.5-turbo-0613`	Chat	RLHF	June 2023
`gpt-3.5-turbo-0301`	Chat	RLHF	Mar 2023
`text-davinci-003`	Text completion	RLHF	Nov 2022
`text-davinci-002`	Text completion	fine-tuning	Mar 2022

Table 1: Representative models from OpenAI's GPT-3.5 family [28], sorted by release date.

of toxicity detection, a task widely used for online content moderation and long performed by models specifically trained for that task [15], but recently LLMs with a suitable prompt have shown similar or better performance [38]. Our case study aims to explore (a) how prompt behaviors change (regress) over LLM updates and (b) where regressions can be detected.[2]

3.1 Experiment Setup

3.1.1 Datasets. We selected two toxicity detection datasets for our case study: Civil Comments [8] and GitHub Discussion [22], covering different contents (*generic* vs. *specialized*) and text lengths (*short* vs. *long*) for toxicity detection.

The Civil Comments dataset is collected from Civil Comments platform, representing a wide range of comments on the Internet. We sampled 1000 comments from the dataset, among which 41 are toxic and 959 are non-toxic.

The GitHub Discussion Dataset contains 174 discussions, among which 74 are toxic and 100 are non-toxic. The 74 toxic discussions are collected using the links provided by an existing study [22], and we randomly sample another 100 non-toxic discussions from GitHub.

3.1.2 Models. We selected five widely used models from OpenAI's GPT-3.5 family [28]: `gpt-3.5-turbo-instruct`, `gpt-3.5-turbo-0613`, `gpt-3.5-turbo-0301`, `text-davinci-003`, and `text-davinci-002` (shown in Table 1). These models were released over a span of only 18 months, from March 2022 to September 2023, covering different endpoint types (chat vs. completion) and training methods (fine-tuning vs. RLHF). We treat each model pair as a potential update and study 10 model update pairs in our experiment.

Noticeably, four out of these five models are already scheduled to be deprecated in 2024, effectively forcing application developers to switch to one of the newer models. The models are also updated silently: `gpt-3.5-turbo-0301` and `gpt-3.5-turbo-0613` are snapshots of the `gpt-3.5-turbo` model, which will soon point to `gpt-3.5-turbo-1106`.

3.1.3 Prompts. We employed four prompting strategies to explore how they behave differently on model updates:

- **Simple instruction (P1)**: The prompt instructs the model to classify the text as "toxic" or "non-toxic", followed by the text to classify. This serves as a simple baseline a developer might first try.
- **Simple instruction, placed last (P2)**: The same as above but put instructions after the text. This design follows the insight that LLMs have recency bias [45] and stating instructions last makes LLMs less likely to ramble [20].

```
# Simple instruction (P1)
Classify the GitHub discussion as "toxic" or "non-toxic".
     Only reply with the label.
Document: {text}

# Simple instruction, placed last (P2)
Document: {text}
Classify the GitHub discussion as "toxic" or "non-toxic".
     Only reply with the label.

# Detailed instruction (P3)
Below is a GitHub discussion. Sometimes the discussion can
     get heated and have toxic comments. Toxic comments can
     contain curse words, can sound condescending, can be
     mean to others, or can make people feel angry without
     using offensive words.
Classify the GitHub discussion as "toxic" or "non-toxic".
     Only reply with the label.
Document: {text}

# Simple instruction + Few-shot examples (P4)
Classify the GitHub discussion as "toxic" or "non-toxic".
     Only reply with the label.

Document: [text omitted]
Label: toxic

Document: [text omitted]
Label: non-toxic

Document: {text}
Label:
```

Figure 2: Prompt templates for our experiments on the GitHub discussion dataset. Templates for the Civil Comments dataset are similar with some adaptations.

- **Detailed instruction (P3)**: The prompt first describes the classification goal in detail, explaining what the developer deems toxic. The description is followed by the classification instruction and the text to classify.
- **Simple instruction + Few-shot examples (P4)**: After simple instructions, the prompt shows two examples, one toxic and the other non-toxic, followed by the text to classify. This follows the popular in-context learning paradigm [5].

We share the prompt templates in Figure 2.

3.1.4 Metrics. To evaluate the accuracy of each model + prompt combination and monitor their changes, we use the standard performance metrics accuracy and F1, and set model temperature to 0 to obtain the most likely predictions.

[2] Code available at https://github.com/MAWanqin2002/LLM_Regression_Testing.

Model	Civil Comments				GitHub Discussion			
	P1	P2	P3	P4	P1	P2	P3	P4
`gpt-3.5-turbo-instruct`	0.688	0.518	0.733	**0.820**	0.638	**0.793**	0.770	**0.793**
`gpt-3.5-turbo-0613`	0.671	0.745	**0.928**	0.774	0.822	**0.862**	**0.856**	0.776
`gpt-3.5-turbo-0301`	0.767	0.743	**0.915**	0.733	0.810	0.799	**0.868**	0.816
`text-davinci-003`	0.862	0.814	**0.938**	**0.933**	0.655	**0.672**	0.644	0.655
`text-davinci-002`	0.803	0.587	**0.861**	0.822	0.839	**0.874**	0.810	0.770

Table 2: Accuracy for prompt (P*n*) and model combinations on the Civil Comments and GitHub Discussion datasets. The best-performing prompt(s) for each LLM API are highlighted in bold. We observed similar results for F1 scores.

3.2 Observations

3.2.1 Prompt performance can regress over API updates. First of all, we found that regression does exist over API updates: 58.8% of prompt + model combinations drop accuracy over API updates (Table 2). Among them, 70.2% drop accuracy greater than 5%. Noticeably, across all different prompts, the model update from `text-davinci-002` to `text-davinci-003` causes a consistent performance drop (16.8% on average) on the GitHub Discussion Dataset but a consistent performance increase (11.8% on average) on the Civil Comments dataset. We hypothesize that the huge performance differences are due to the new training method `text-davinci-003` used, which causes major inconsistency across the two versions.

3.2.2 Model updates affect different prompting strategies differently. We observed that among all model updates, 55% do not cause a consistent performance drop or increase across prompts, i.e., the same model update helps some prompts but hurts others for the same task. Specifically, we found that the simplest prompt, P1, drops accuracy in 75% of the model updates, while the few-shot prompt, P4, only drops accuracy 45% of all times. Zooming in, we can see that the update from `gpt-3.5-turbo-0301` to `gpt-3.5-turbo-0613` caused a 9.6% accuracy drop for P1, but a 5.1% increase for P4 on the Civil Comments dataset. This is particularly concerning, as the update is silent when a developer uses the main API `gpt-3.5-turbo`, which updates the underlying model from time to time.

Such non-uniform performance changes cause a major problem for prompt engineering: The developer may find that their carefully engineered prompt is no longer the best choice after a silent API update. For example, the detailed instruction prompt (P3) has been the best-performing prompt up to the last model update, but falls behind the few-shot prompt (P4) by 8.7% on the latest model (`gpt-3.5-turbo-instruct`). This indicates that prompt engineering is not a one-time effort, and calls for prompt versioning and prompt monitoring (see detailed discussion in Section 4.2).

3.2.3 Regressions happen even when prompt performance improves. We also found that overall 10.9% individual predictions regress (from correct to wrong) over API updates. Almost always (87.9%) when overall accuracy improves in an update, at least one previously correct prediction regresses. For example, the model update from `text-davinci-002` to `text-davinci-003` improves P3's accuracy on the Civil Comments dataset by 7.7%, but 1.8% of the previously correct predictions now fail.

As such regressions are invisible in the aggregated accuracy scores, it would be particularly concerning if the improvements and regressions are not uniform–the prompt may work better on some data slices but worse on others, causing fairness implications even when overall accuracy stays stable or improves.

3.2.4 Regressions happen beyond the decision boundary. A natural hypothesis is that regressions happen on data points that models are less confident with (i.e. near the decision boundary). To explore this hypothesis, following existing work [44], we use information entropy to measure the model's confidence on a data point:

$$E_j = \sum_i -p_{ij} \cdot \log p_{ij}$$

where E_j is the entropy on input j, and p_{ij} is the model's probability to predict label i on input j. Intuitively, when the model's prediction probabilities are more evenly distributed across different labels, the entropy is higher and the model is more uncertain on the input. Since many LLM APIs do not expose the actual prediction probabilities, we approximate a model's prediction probabilities by running it on the same input multiple (n=20) times with a non-zero temperature (t=0.7).

Overall, we found that models are indeed more uncertain about flipping data points on average (Table 3). However, we also found that 63.8% of regressions happen when models are very confident about their results (i.e. entropy = 0). This implies that model updates can drastically change predictions on data points far away from the decision boundary.

Across the models, we also found that different models show different levels of self-consistency: `gpt-3.5-turbo-0301` seems to be the most self-consistent one, while the update to `gpt-3.5-turbo-0613` makes it much less self-consistent. This indicates another form of *regression*: While the two models' accuracy is comparable, the update can affect model calibration [45] and make the model less self-consistent (or over-confident).

3.2.5 Regressions are not uniform across data slices. We next explore where regressions happen systematically for specific data slices, with the metadata provided by the authors of the GitHub Discussion dataset [22].

We found that 90% of regressions happen on toxic discussions, despite only 42.5% of discussions being toxic in the dataset. Breaking down the regression on toxic discussions by the provided metadata (Figure 3), we found that regressions are disproportionally common when the toxicity is triggered by politics (25.7% overall vs. 33.3% among regressions), targets code (21.6% vs. 33.3%), or is severe (54.1% vs. 66.7%), suggesting that model updates can cause systematic worse performance for these specific data slices.

Model	Civil Comments			GitHub Discussion		
	Regression	Improvement	Unflipped	Regression	Improvement	Unflipped
gpt-3.5-turbo-instruct	0.319	0.289	0.078	0.186	0.213	0.258
gpt-3.5-turbo-0613	0.190	0.138	0.025	0.267	0.139	0.063
gpt-3.5-turbo-0301	0.075	0.096	0.006	0.015	0.010	0.026
text-davinci-003	0.022	0.028	0.010	0.005	0.018	0.018
text-davinci-002	0.251	0.296	0.137	0.467	0.302	0.227
average	0.171	0.169	0.051	0.188	0.136	0.118

Table 3: Model entropy on the Civil Comments and GitHub Discussion datasets, averaged across all prompts.

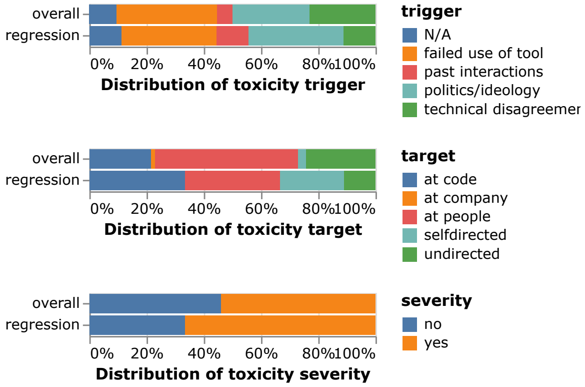


Figure 3: Regressions disproportionally happen when the toxicity relates to politics (25.7% vs. 33.3%), targets code (21.6% vs. 33.3%), or is severe (54.1% vs. 66.7%).

3.2.6 Limitations. Readers should be careful when generalizing the results beyond the current experiment settings: We used specific prompt formats and sent the prompts as a single user request. The optimal prompt design may change when the LLM API varies.

4 TOWARDS REGRESSION TESTING FOR PROMPTING LLMS

Our exploratory case study highlights that model regression is a *real* problem that is deeply affected by prompting and LLM non-determinism. Based on our observations, we conclude with a discussion on how researchers can support regression testing for LLMs.

4.1 Identifying Data Slices as Regression Test Suites

In our case study, we found that individual predictions regress frequently (10.9%). Therefore, treating each data point as a regression test will simply be intractable. An alternative would be to look at aggregated metrics over the entire dataset. However, this level of monitoring is too coarse-grained and cannot inform developers on how to debug and adjust their prompts.

We argue that LLM regression tests should be at the level of *slices*. Our preliminary results show that it is possible to look at *semantic* slices and localize where regressions happen (e.g., *toxicity targeting code* for GitHub toxicity). However, our slicing relies on extra metadata, which may not be available for many datasets. Future research should further scaffold developers to identify data slices as regression test suites, possibly by transferring existing approaches like slice discovery [11] and error analysis [6, 39] on a single model to regression testing.

4.2 Tracking Prompts for Regression Testing

Our case study points out that prompt performance can be unstable across different APIs and each API has different best-performing prompts. Therefore, developers need to track and update their prompt (possibly from a *history* version), to maintain or improve prompt+LLM performance.

However, existing prompt engineering practices provide insufficient support for prompt versioning and monitoring [43]– Information and knowledge are often lost in the iterative prompt engineering process. Future research can design systems for prompt+LLM tracking [e.g., 2, 24] to help developers explore behavioral changes, debug regressions, and update their prompts.

4.3 Tackling Non-determinism in LLM Regression Testing

Our case study shows that LLM predictions can flip a lot with a non-zero temperature. This can cause lots of *flakiness* when we perform regression testing for LLMs. Future research on LLM regression testing should explicitly consider such non-determinism in their research design. For example, to avoid a large sample size for each regression test, researchers can develop suitable statistical tests and test minimization strategies.

While our work focused on classification tasks, regressions can also happen for generative tasks, where non-determinism is even more common for generating high-quality outputs. To support regression testing LLMs on generative tasks, future research should consider incorporating multi-dimensional metrics [46] and supporting developers in testing output properties specific to their requirements [31].

ACKNOWLEDGMENTS

We thank Sherry Tongshuang Wu and Rohan Padhye for their discussion and feedback on this work.

REFERENCES

[1] Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. 2020. Debugging Tests for Model Explanations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Article 60, 13 pages.

[2] Saleema Amershi, Max Chickering, Steven M. Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, 337–346.

[3] Anthropic. 2023. Claude. https://claude.ai/

[4] Guy Barash, Eitan Farchi, Ilan Jayaraman, Orna Raz, Rachel Tzoref-Brill, and Marcel Zalmanovici. 2019. Bridging the Gap between ML Solutions and Their Business Requirements Using Feature Interactions. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Tallinn, Estonia) *(ESEC/FSE 2019)*. Association for Computing Machinery, 1048–1058.

[5] Tom Brown, Benjamin Mann, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[6] Ángel Alexander Cabrera, Erica Fu, Donald Bertucci, Kenneth Holstein, Ameet Talwalkar, Jason I. Hong, and Adam Perer. 2023. Zeno: An Interactive Framework for Behavioral Evaluation of Machine Learning. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, Article 419, 14 pages.

[7] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is ChatGPT's behavior changing over time? arXiv:2307.09009 [cs.CL]

[8] cjadams, Daniel Borkan, inversion, Jeffrey Sorensen, Lucy Dixon, Lucy Vasserman, and nithum. 2019. Jigsaw Unintended Bias in Toxicity Classification. https://kaggle.com/competitions/jigsaw-unintended-bias-in-toxicity-classification

[9] Alex Cummaudo, Scott Barnett, Rajesh Vasa, John Grundy, and Mohamed Abdelrazek. 2020. Beware the Evolving 'intelligent' Web Service! An Integration Architecture Tactic to Guard AI-First Components. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020)*. Association for Computing Machinery, 269–280.

[10] Alex Cummaudo, Rajesh Vasa, John Grundy, Mohamed Abdelrazek, and Andrew Cain. 2019. Losing Confidence in Quality: Unspoken Evolution of Computer Vision Services. In *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 333–342. https://doi.org/10.1109/ICSME.2019.00051

[11] Sabri Eyuboglu, Maya Varma, Khaled Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Ré. 2022. Domino: Discovering systematic errors with cross-modal embeddings. *arXiv preprint arXiv:2203.14960* (2022).

[12] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. 2023. InCoder: A Generative Model for Code Infilling and Synthesis. In *The Eleventh International Conference on Learning Representations*.

[13] Google. 2023. Bard. https://bard.google.com/chat

[14] Google. 2023. Using machine learning to reduce toxicity online. https://www.perspectiveapi.com/

[15] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138* (2017).

[16] Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. 2023. Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169* (2023).

[17] Christian Kästner. 2022. *Machine Learning in Production: From Models to Products*.

[18] Jun Li, Yingfei Xiong, Xuanzhe Liu, and Lu Zhang. 2013. How does web service API evolution affect clients?. In *2013 IEEE 20th International Conference on Web Services*. IEEE, 300–307.

[19] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9, Article 195 (jan 2023), 35 pages.

[20] Yanjun Liu, Xianfeng Zeng, Fandong Meng, and Jie Zhou. 2023. Instruction Position Matters in Sequence Generation with Large Language Models. *ArXiv* abs/2308.12097 (2023). https://api.semanticscholar.org/CorpusID:261076308

[21] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 8086–8098.

[22] Courtney Miller, Sophie Cohen, Daniel Klug, Bogdan Vasilescu, and Christian Kästner. 2022. "Did You Miss My Comment or What?" Understanding Toxicity in Open Source Discussions. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*. 710–722. https://doi.org/10.1145/3510003.3510111

[23] Abhishek Mishra. 2019. Machine learning in the AWS cloud: Add intelligence to applications with Amazon Sagemaker and Amazon Rekognition. https://aws.amazon.com/rekognition/

[24] Aditi Mishra, Utkarsh Soni, Anjana Arunkumar, Jinbin Huang, Bum Chul Kwon, and Chris Bryan. 2023. PromptAid: Prompt Exploration, Perturbation, Testing and Iteration using Visual Analytics for Large Language Models. *arXiv preprint arXiv:2304.01964* (2023).

[25] Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of the 27th International Conference on Computational Linguistics*. Association for Computational Linguistics, 2340–2353.

[26] OpenAI. 2023. ChatGPT. https://chat.openai.com/

[27] OpenAI. 2023. Deprecations - OpenAI API. https://platform.openai.com/docs/deprecations

[28] OpenAI. 2023. GPT-3.5 Documentation. Retrieved from. https://platform.openai.com/docs/models/gpt-3-5

[29] Shuyin Ouyang, Jie M Zhang, Mark Harman, and Meng Wang. 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. *arXiv preprint arXiv:2308.02828* (2023).

[30] radiator57. 2023. Experiencing Decreased Performance with ChatGPT-4. https://community.openai.com/t/experiencing-decreased-performance-with-chatgpt-4/234269

[31] Marco Tulio Ribeiro. 2023. Testing language models (and prompts) like we test software. *Medium* (May 2023). https://towardsdatascience.com/testing-large-language-models-like-we-test-software-92745d28a359

[32] Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive Testing and Debugging of NLP Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 3253–3267.

[33] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond Accuracy: Behavioral Testing of NLP Models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, 4902–4912.

[34] Ian Sommerville. 2015. *Software Engineering* (10th ed.). Pearson.

[35] Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic Testing and Improvement of Machine Translation. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (Seoul, South Korea) *(ICSE '20)*. Association for Computing Machinery, 974–985.

[36] Zeyu Sun, Jie M. Zhang, Yingfei Xiong, Mark Harman, Mike Papadakis, and Lu Zhang. 2022. Improving Machine Translation Systems via Isotopic Replacement. In *Proceedings of the 44th International Conference on Software Engineering* (Pittsburgh, Pennsylvania) *(ICSE '22)*. Association for Computing Machinery, 1181–1192.

[37] Shangqing Tu, Chunyang Li, Jifan Yu, Xiaozhi Wang, Lei Hou, and Juanzi Li. 2023. ChatLog: Recording and Analyzing ChatGPT Across Time. *arXiv preprint arXiv:2304.14106* (2023).

[38] Yau-Shian Wang and Yingshan Chang. 2022. Toxicity detection with generative prompt-based inference. *arXiv preprint arXiv:2205.12390* (2022).

[39] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel Weld. 2019. Errudite: Scalable, Reproducible, and Testable Error Analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, 747–763.

[40] Chenyang Yang, Rachel A Brower-Sinning, Grace Lewis, Christian Kästner, and Tongshuang Wu. 2023. Capabilities for Better ML Engineering. In *Proceedings of the AAAI-23 Workshop on Artificial Intelligence Safety (SafeAI)* (Washington, DC).

[41] Chenyang Yang, Rishabh Rustogi, Rachel Brower-Sinning, Grace A Lewis, Christian Kästner, and Tongshuang Wu. 2023. Beyond Testers' Biases: Guiding Model Testing with Knowledge Bases using LLMs. (12 2023). http://arxiv.org/abs/2310.09668

[42] Robert K Yin. 2009. *Case study research: Design and methods*. Vol. 5. sage.

[43] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, Article 437, 21 pages.

[44] Xuchao Zhang, Fanglan Chen, Chang-Tien Lu, and Naren Ramakrishnan. 2019. Mitigating Uncertainty in Document Classification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, 3126–3136.

[45] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12697–12706.

[46] Ming Zhong, Yang Liu, Da Yin, Yuning Mao, Yizhu Jiao, Pengfei Liu, Chenguang Zhu, Heng Ji, and Jiawei Han. 2022. Towards a Unified Multi-Dimensional Evaluator for Text Generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 2023–2038.