STAR: Self-Automated Back-Querying for Production Data Generation

Kellen Tan Cheng[♠,*], **Anna Lisa Gentile**[♣], **Chad DeLuca**[♣], **Guang-Jie Ren**[♢,*]
[♣]IBM Research [♠]Microsoft [♢]Adobe
[♠]kellencheng@microsoft.com, [♢]gren@adobe.com
[♣]annalisa.gentile@ibm.com, delucac@us.ibm.com

Abstract

The pervasiveness of large language models (LLMs) in enterprise settings has also brought forth a significant amount of risks associated with their usage. Guardrails technologies aim to mitigate this risk by filtering LLMs' input/output text through various detectors. However, developing and maintaining robust detectors has many challenges, one of which is the difficulty in acquiring production-quality labeled data on real LLM outputs before deployment. In this work, we propose STAR, a simple yet intuitive solution to generate production-like labeled data for LLMs' guardrails development. STAR is based on two key ideas: (i) using *self-automated back-querying* to synthetically generate data, paired with (ii) a sparse human-in-the-loop clustering technique to label the data. The aim of self-automated back-querying is to construct a parallel corpus roughly representative of the original dataset and resembling real LLM output. We then infuse existing datasets with our synthetically generated examples to produce robust training data for our detectors. We test our technique on one of the most difficult and nuanced detectors: the identification of health-advice in LLM output, and demonstrate improvement versus other solutions. Our detector is able to outperform GPT-4o by up to 3.48%, despite having 400x less parameters.

1 Introduction

The advancement of large language models (LLMs) has brought about impressive capabilities in a wide variety of natural language tasks (OpenAI et al., 2024; Dubey et al., 2024). However, the fact that these models are pre-trained on massive text corpora inevitably results in the generation of some undesirable outputs that may be misleading and/or factually incorrect. Many prominent LLMs have methods in place to safeguard their interactions with users (Rebedea et al., 2023; Inan et al., 2023; Markov et al., 2023; Salem et al., 2023; Dong et al., 2024), but developing guardrails technology that can effectively minimize LLMs' usage risks remains an open challenge. Additionally, conventional techniques typically involve a nontrivial human component, whether for crafting/curating specific datasets for the task or for performing red-teaming.

One prominent challenge in constructing effective and robust guardrails is obtaining high-quality production data. This is because there exists a significant distribution shift between open-source fine-tuning (FT) datasets, which are typically human-curated, and the data that is actually encountered during inference, which is generated from LLMs (Achintalwar et al., 2024; Koh et al., 2021; Huang et al., 2021; Taori et al., 2020). Additionally, only a select few corporations have access to large-scale production datasets containing LLMs' prompts and responses, but given their proprietary nature, it is impossible to utilize them for guardrails development. This scarcity is exacerbated in domains such as healthcare or finance, due to privacy concerns and the involvement of critical decision-making within the data (Park et al., 2021; Liu et al., 2023; Du et al., 2024). As guardrails technology is ultimately targeting LLM outputs, there is a need for production-quality data to bridge the inherent distribution shift. While there have been considerable efforts to construct various LLM risk benchmark datasets (Ganguli et al., 2022; Mazumder et al., 2023; Ji et al., 2023; Wang et al., 2024), there is still a nontrivial human cost, and manually constructing benchmarks for each particular risk category is not scalable.

Towards addressing this problem, we introduce STAR: **S**elf-au**T**omated b**A**ck-que**R**ying, a simple yet intuitive framework for synthetic generation of real-world production data. Inspired by the concept of backtranslation (Sennrich et al., 2016),

[*]Work done while at IBM Research.

Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 133–148

STAR uses an initial set of proprietary annotated data and generates a completely new set of data. STAR works by (i) generating a prompt for each given input text, then (ii) feeding the prompts back to an LLM, (iii) using the generated text as the new text, and (iv) performing a sparse human-in-the-loop labeling scheme, making minimal use of human feedback to effectively produce labeled synthetic data. Our framework is highly modular, making no restrictions on the type of guardrails task, the type of input data, or even the LLM which is used for the text generation.

STAR allows for the automated creation of synthetic datasets used for guardrails fine-tuning. Unlike proprietary production data, STAR generates the data necessary to develop detector models for various guardrails tasks without actually accessing any real-world task data. We demonstrate the effectiveness of STAR by applying it to the task of identifying health-advice in LLMs' responses, showing that a lightweight detector model fine-tuned on STAR data can outperform GPT-4o by up to 3.48% despite our detector model having 400x fewer parameters. Additionally, our detector exhibits a more balanced behavior during inference, with the smallest difference between precision and recall at just 2.14% (compared to 13.85% for GPT-4o).

The contributions of this work are as follows:

(1) a framework to generate data in the style of an LLM's outputs.

(2) a semi-automatic sparse human-in-the-loop annotation scheme to label the synthetically generated data.

(3) a two-stage fine-tuning setup to better adapt language models towards the STAR data, where the first stage incorporates a mix of STAR data and open-source datasets, and the second stage uses purely synthetic STAR data.

In Section 2, we review contemporary approaches and techniques. In Section 3, we describe the STAR framework in detail. In Section 4, we detail the datasets used for our task. In Section 5, we showcase the benefits of STAR generated data for health-advice identification. Finally, in Section 6, we describe future work and planned improvements. Note that we may refer to *synthetic data* and *STAR data* interchangeably throughout this paper, but both terms reference the synthetic data generated through our STAR framework.

2 Related Work

2.1 Prompt Generation

Prior research has utilized many techniques to generate appropriate prompts or queries from text. One interesting line focuses on inverting LLM outputs with minimal access to supplemental information, using just the next-token probabilities or even just the outputs of the user queries themselves (Morris et al., 2024; Zhang et al., 2024). Numerous approaches also exist for soft prompt generation, which involve fine-tuning continuous vectors prepended to the LLM inputs (Wang et al., 2022; Li and Liang, 2021). Recent interest has been towards approaches to optimize and generate discrete, interpretable prompts that contain the words themselves, as opposed to just vectors (Deng et al., 2022; Wen et al., 2023). Different from the preceding approaches, our STAR framework does not require specialized fine-tuning for the prompt generation step, demonstrating its utility as a framework even without prompt generation specialization. Furthermore, given the modular nature of STAR, we believe that our framework is complementary to any prompt generation approach, allowing a user to substitute the query generation stage with a more specialized query generation module if desired.

2.2 Question Generation

There have been various approaches to generate questions from input texts. Differently from prompt generation, these methods focus on question generation (Du et al., 2017), rather than prompt inversion or prompt learning. Prior work has focused on validating summary quality with questions (Wang et al., 2020), generating question-answer pairs (Krishna and Iyyer, 2019), automatic question generation for event-extraction (Liu et al., 2020), or using templates or knowledge graphs to aid in question generation (Gaur et al., 2022; Kumar et al., 2019; Reddy et al., 2017; Fabbri et al., 2020). Unlike prior work, our approach considers both query generation and the corresponding output generation, in a manner akin to backtranslation. Additionally, it is possible to complement the STAR framework with a more optimized query generation scheme, such that it works in tandem with prior work.

2.3 Synthetic Data Generation

Synthetic data generation remains a pertinent and useful capability of present-day LLMs (Long et al., 2024; Kruschwitz and Schmidhuber, 2024). Mainstream techniques for LLMs mainly focus on a variety of prompt-engineering techniques such as creating roleplay (Li et al., 2023), defining task specifications or taxonomies (Yoo et al., 2021; Sudalairaj et al., 2024), knowledge graphs (Xu et al., 2024), feedback (Ye et al., 2022), and in-context learning examples (Wang et al., 2023; Li et al., 2024). Our approach differs in that we are optimizing to match the LLM outputs' distribution, rather than data quality itself. This is because we consider data generation for the application of guardrails development, and such erroneous or dirty samples are texts that could realistically be generated by an LLM during inference. As a result, including some imperfect samples allows our detector model to be even more robust.

2.4 Guardrails Development

In recent years, guardrails development for LLMs has been a prominent subfield within natural language processing (Dong et al., 2024). There have been a variety of different approaches to implementing guardrails, from using lightweight detector models (Achintalwar et al., 2024), taxonomies and/or red-teaming with LLMs (Inan et al., 2023; Markov et al., 2023), human programmable guardrails (Rebedea et al., 2023), to query-modification and fusion models (Yuan et al., 2024; Xiang et al., 2024). Our approach is also one method for guardrails development, but instead of model architecture optimizations, taxonomy creation, or runtime inference input/output rewriting or modifications, we focus more on the data creation step, namely generating high-quality production-like data that can be used to make robust datasets for creating guardrails models. In this sense, we are less focused on an actual model framework and more on how to provide the tools (i.e. data) necessary to help facilitate guardrails development.

3 STAR

We describe the STAR framework, as seen in Figure 1, providing details on the data generation procedure (Section 3.1) and on the sparse human-in-the-loop (sparse-HITL) labeling algorithm (Section 3.2). Please refer to Appendix B for the comprehensive list of hyperparameters used in the STAR framework.

Prompt
"What question did the user ask to generate the following text: $\{x_i\}$ The user prompt is:"

Table 1: The prompt that we used to generate queries in stage 1 of our STAR framework. Note that $x_i \in \mathcal{X}$ represents one sample from our seed dataset.

3.1 Data Generation

STAR generates synthetic production-quality data via an automated back-querying procedure. Our back-querying protocol works in two stages: (i) *query generation* from the original texts, and (ii) *answer generation* which produces new output texts from our generated queries. Our base framework utilizes LLaMA 3.1-8B as the LLM for both stages (Dubey et al., 2024).

We take as input a seed dataset $\mathcal{X} = \{x_1, \ldots, x_n\}$, where each x_i is a text sample, such as a sentence, a paragraph, or a document. Since STAR supplies its own sparse-HITL labeling scheme (Section 3.2), we remark that there is no restriction that the seed dataset $\mathcal{X}$ be annotated. In the *query generation* stage, we produce a query set $\mathcal{Q} = \{q_1, \ldots, q_n\}$ such that each query $q_i \in \mathcal{Q}$ corresponds to the text $x_i \in \mathcal{X}$, and is generated by asking our LLM which question has the text x_i as a potential answer. The specific prompt template is illustrated in Table 1.

In the *answer generation* stage, we use the set of queries $\mathcal{Q}$ to generate synthetic data. For each query $q_i \in \mathcal{Q}$, we prompt our LLM to generate a response y_i, resulting in a synthetic dataset $\mathcal{Y} = \{y_1, \ldots, y_n\}$. This data is production-quality, since each output y_i is LLM-generated, and thus distributed accordingly to what would be observed in the wild. It is also possible to increase the amount of synthetic data generated by simply feeding the same set of queries $\mathcal{Q}$ through different LLMs.

3.2 Sparse-HITL

Note that even if the original dataset $\mathcal{X}$ contains labeled text, we cannot assume that the original label for x_i holds for the synthetically generated text y_i – the process does not guarantee complete equivalence between x_i and y_i. Therefore, we propose

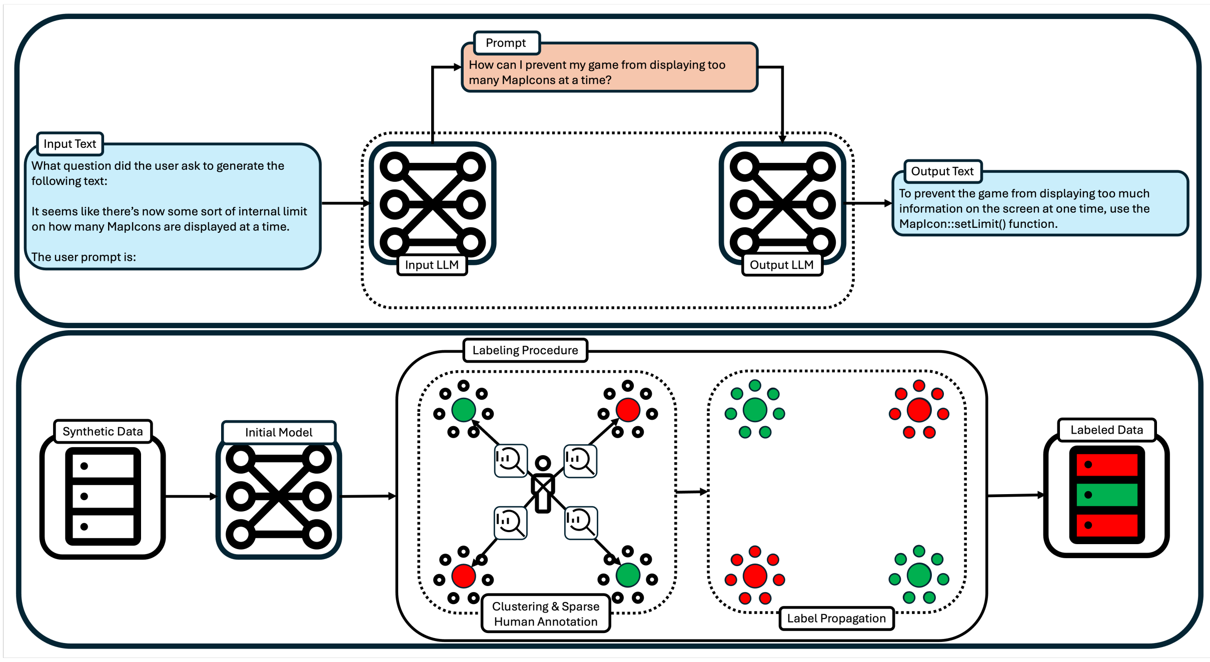

Figure 1: An overview of the STAR schematic. (Top) For each data point, we first transform it into a query, and then re-prompt an LLM with our formulated query to generate a new synthetic data point. We note that for our particular implementation, the input LLM and the output LLM are the same. (Bottom) To label our synthetically generated data, we first use an initial model to split the data by predicted label, and then within each split, cluster the samples by their embeddings. Then, a human annotator labels only the cluster centroids, before then propagating this label onto all cluster members.

the use of sparse-HITL to perform semi-automatic annotation on our synthetic dataset $\mathcal{Y}$ without incurring high manual labor overhead. First, we use an initial classification model $\mathcal{M}$ (Section 3.3) that was fine-tuned towards the target classification task. We then split $\mathcal{Y}$ according to their predicted labels, as output by $\mathcal{M}$. Within each group, we generate embeddings for each sample and then cluster them based on their Euclidean distance. Finally, we manually annotate only the cluster centroids, then propagate this human-annotated label onto all cluster members. In this manner, human annotation is only needed for one data point per cluster, thus limiting the number of manual annotations to the total number of clusters.

For this work, we constructed $\mathcal{M}$ by taking an off-the-shelf BART-Large model (Lewis et al., 2020) and fine-tuning it towards the target task using open-source academic datasets (see Section 4). We generate the embedding from our model $\mathcal{M}$ (extracted from the last layer hidden state) and perform clustering using the k-means algorithm (MacQueen, 1967), setting $k = 20$ as our default number of clusters. As a result, for a binary classification task, we only need to manually label 40 samples (20 clusters per two labels), as opposed to annotating all samples in our dataset. In a trivial scenario, where there are fewer samples than the number of clusters, we simply annotate all individual points.

3.3 Embedding Model

Recall that our embedding model $\mathcal{M}$ is sourced from a BART-Large model (Lewis et al., 2020) which has been fine-tuned towards the task of health-advice identification. To fine-tune $\mathcal{M}$, we construct our training dataset by combining 5 academic datasets spanning both advice and health-advice recognition: NeedAdvice (Govindarajan et al., 2020), AskParents (Govindarajan et al., 2020), SemEval2019-Task9 (Negi et al., 2019), Detecting-Health-Advice (Li et al., 2021), and HealthE (Gatto et al., 2023). Details are provided in Section 4 and Table 2.

3.4 Two-Stage Fine-Tuning

We implement fine-tuning in two stages in order to gradually align our detector model to the guardrails task at hand. The first stage of fine-tuning is done on a combination of synthetic and open-source datasets, where the synthetic dataset is generated from a seed dataset. This synthetic data in the first stage uses only seed examples that are negative, i.e. do not violate our guardrails task. The motivation here is that during inference, a vast majority of samples will be irrelevant and not violate any guardrails. Thus, we aim to increase the data coverage in order to ensure that the detector model is able to accurately classify any irrelevant samples correctly as negatives, reducing the false positive rate. Without this step, the model is more prone to errors when it encounters irrelevant samples, since otherwise they would never have been seen before during fine-tuning.

After the first stage, the model has now seen a wide range of inputs and knows roughly how to deal with irrelevant samples. Then, in the second stage, we continue to fine-tune the model, but on purely synthetic data in order to tune its behavior on relevant samples. In this stage, we use the largest balanced portion of purely synthetic data. Note that we use an off-the-shelf BART-Large architecture (Lewis et al., 2020) as our detector model. Unless otherwise stated, all open-source datasets used for fine-tuning make use of all of their available splits (i.e. we combine their train, test, and validation splits).

4 Datasets

While the STAR framework can be utilized to generate models addressing various LLM guardrails tasks, in this work we focus specifically on health-advice recognition, i.e. detecting whether a given LLM output text contains health-advice. Note that we define health-advice as follows: *health-advice (boolean) refers to any text that contains an explicit recommendation or suggestion on a course of actions that a person should take*. Importantly, this guardrails task is not concerned with distinguishing between helpful versus harmful advice, but simply whether it is present. We formulate this problem as a three-way classification problem, where our labels are *health-advice*, *not health-advice*, and *general-content*. The addition of a *general-content* class helps introduce an additional layer of granularity during fine-tuning, ensuring that the predictions remain consistent for text that is not health-related. However, during inference, we treat both *general-content* and *not health-advice* equally as part of the negative class. Results for this task are evaluated on the gold-standard HeAL benchmark dataset (Cheng et al., 2024)[1].

To construct our fine-tuning dataset for stage one, we first synthetically generate general-content samples using SemEval2019-Task9 (Negi et al., 2019) as our seed dataset. We then perform semi-automatic annotation using the sparse-HITL labeling scheme (as detailed in Section 3.2). We combine this synthetic data with HealthE (Gatto et al., 2023) and Detecting-Health-Advice (Li et al., 2021), two health-advice datasets, to obtain the final stage-one fine-tuning dataset. Note that for the Detecting-Health-Advice dataset we combine both the weak advice and strong advice labeled samples into the single class health-advice.

The stage-two dataset is constructed from purely synthetic examples generated by the STAR framework. We combine both HealthE and Detecting-Health-Advice, and extract all the positive samples to use as our seed data points. After generation and labeling, we select the maximal balanced subset of this STAR data as the final stage-two fine-tuning dataset.

5 Results & Discussion

5.1 Comparison with Vanilla FT

We compare and analyze both stages in our sequential FT strategy, and compare it with baseline vanilla FT.

2-Stage FT We observe in Table 3 that our detector model, paired with 2-stage FT, achieves state-of-the-art performance, beating out GPT-4o by 3.48% in terms of accuracy and 1.82% in terms of F1-score. This performance is statistically significant up to 90% confidence and is achieved despite our detector model containing only 400M parameters. This is in stark contrast with the 13B parameters required for Mixtral-8x7B (Jiang et al., 2024), 70B parameters for LLaMA 3-70B-Instruct (Dubey et al., 2024), and at least 175B for GPT-4o (OpenAI et al., 2024). Additionally, our detector model not only outperforms state-of-the-art but also exhibits balanced behavior when encountering negative versus positive samples, evidenced by a difference of just 2.16% between its precision and recall. Conversely, previous state-of-the-art models like GPT-4o are overly critical and tend to predict the positive class more often, resulting in high recall but low precision. This results in GPT-4o erroneously flagging many irrelevant samples as health-advice. In fact, GPT-4o exhibits the largest difference between precision and recall at 13.85%.

Furthermore, we note that our results also demonstrate why we only use synthetic samples corresponding to positive seeds in the second stage

[1] https://doi.org/10.6084/m9.figshare.27198735

Dataset	Dataset Statistics			
	Num. Samples	Health-Content	Health-Advice	General-Content
AskParents	9931	0	0	9931
NeedAdvice	7452	0	0	7452
Detecting-Health-Advice	10848	8100	2748	0
HealthE	5656	2256	3400	0
SemEval2019-Task9	9925	0	0	9925
Synthetic Stage 1	23298	10371	6150	6777
Synthetic Stage 2	1140	380	380	380
HeAL	402	161	241	0

Table 2: An overview of the datasets we use for health-advice identification, including their class label distribution and number of samples.

Model	Results			
	Accuracy (↑)	Precision (↑)	Recall (↑)	F1 (↑)
Detector 2-Stage	**85.07**%	86.64%	88.80%	**87.70**%
Detector 1-Stage	82.34%	**87.61**%	82.16%	84.80%
Alternate 2-Stage	81.34%	85.78%	82.57%	84.14%
GPT-4o*	81.59%	79.51%	**93.36**%	85.88%
LLaMA 3-70B-Instruct*	81.34%	85.78%	82.57%	84.14%
Mixtral-8x7B*	72.89%	79.15%	72.61%	75.74%

Table 3: Performance of our detector model as evaluated on the HeAL benchmark, compared with different baselines as well as state-of-the-art models. Note that Alternate 2-Stage refers to when we instead use examples generated from positive seeds in the first stage of fine-tuning, and those generated from negative seeds in the second stage. * denotes zero-shot performance. The best-performing results are in **bold**.

of fine-tuning, as opposed to the first stage. From Table 3, using positive seeds in the first stage and negative seeds in the second stage (referred to in the table as "Alternate 2-Stage") degrade the detector performance, achieving only 81.34% accuracy and 84.14% in F1-score. While these results are still comparable with Llama-3-70B-Instruct, it exhibits a drop of 3.73% accuracy and 3.56% in F1-score compared to 2-stage FT.

1-Stage FT Interestingly, even the addition of just synthetic irrelevant samples (general-content) appears to make a noticeable improvement, allowing the detector model to better understand the real-world data distribution. From Table 3, fine-tuning with just the first stage is already enough to perform comparably to state-of-the-art, reaching 82.34% accuracy and 84.80% F1-score. Additionally, our model exhibits a difference of 5.45% between precision and recall, significantly better than GPT-4o albeit worse than LLaMA 3-70B-Instruct. These results demonstrate the need for 2-stage FT to better improve the model performance and balance out the model behavior even further.

Vanilla FT We compare the results of our detector model against a vanilla FT setup, where we FT on the maximal balanced subset from purely synthetic data (i.e. the dataset used for the second stage of our FT scheme). Additionally, we also compare our results with a BART-Large model FT on only academic datasets, replacing the synthetic general-content samples with the original SemEval2019-Task9 seed dataset. As seen from Table 4, vanilla FT using only synthetic data results in a notable degradation in performance, achieving only 77.61% accuracy and 83.27% F1-score. However, the high F1-score itself can be misleading in isolation, as it exhibits many of the negative behaviors of GPT-4o, resulting in very high recall but poor precision (a difference of 17.53%). Furthermore, we observe that replacing the synthetic general-content examples with the original seed dataset also degrades performance compared to 2-stage FT, although it does outperform vanilla FT on purely synthetic data, achieving 80.60% accuracy and 83.19% F1-score, with a difference of 6.47% between precision and recall.

5.2 Synthetic Data Quality

Another point of interest focuses on the quality of the synthetic data. To gauge the noisiness of the sparse-HITL labels, we randomly sampled two examples from each cluster, and then manually anno-

FT Setup	Results			
	Accuracy (↑)	Precision (↑)	Recall (↑)	F1 (↑)
Synthetic Data Only	77.61%	75.42%	**92.95%**	**83.27%**
No Synthetic Data	**80.60%**	**86.55%**	80.08%	83.19%

Table 4: Performance of our detector model utilizing only vanilla FT. The best-performing results are in **bold**.

Dataset	Results		
	FP	FN	Accuracy
Detecting-Health-Advice	5	0	87.50%
HealthE	3	1	90.00%
SemEval	0	0	100.00%

Table 5: Manual annotation of synthetic data label accuracy. Note that FP and FN stand for false positives and false negatives, respectively.

Seed Type	Results		
	Precision (↑)	Recall (↑)	F1 (↑)
Health-Advice	**73.07%**	81.98%	**74.60%**
General-Content	69.13%	**83.72%**	70.97%

Table 6: BERTScore similarities between the generated STAR outputs and their corresponding seed examples. We report the averages within each seed example type for our results. The best performing scores are in **bold**.

tated all of these samples for label accuracy. From Table 5, all datasets exhibit at least 87.50% accuracy, with only HealthE containing a false negative sample. Most of the erroneously labeled samples arise as false positives, where the text is indeed health or medical-related but does not contain explicit advice.

As a quantitative metric, we evaluate the semantic drift between the synthetic and seed examples using BERTScore (Zhang* et al., 2020). a metric designed to evaluate the quality of the generated text. As evidenced from Table 6, synthetic data generated from health-advice samples exhibits a slightly lower semantic drift than those generated from general-content samples, achieving a BERTScore F1 of 74.60% as opposed to 70.97%. This difference is expected: general-content samples are less focused on a particular topic and thus are more likely to exhibit semantic drift from the original seed example. This can be further seen in Table 7, where we observe that the data generated from health-advice datasets tend to stay within the health domain. Specifically, 71.24% and 97.12% of the synthetically generated samples are labeled as health (either health-content or health-advice) for Detecting-Health-Advice and HealthE, respectively. It appears that the examples generated from HealthE are more likely to also stay as health-advice, with 54.32% of the synthetic examples being labeled as health-advice (in keeping with the original label), as opposed to 18.06% for Detecting-Health-Advice. While it seems that semantic drift can push seed examples from health-advice to general-content, the same cannot be said in reverse, with all examples generated from SemEval still being labeled as general-content. From a manual observation of the samples, the generated prompts are the main driving factor behind the semantic drift between the synthetic and original data points. We provide a concrete example of this phenomenon in Table 8, which provides some examples of the original text x_i, the LLM-provided query q_i, and the generated text y_i.

However, as we discussed in our prior results, some degree of semantic drift is desirable. We hypothesize that this is because we expose the detector model to a wider range of LLM outputs. This wider distribution makes the detector better equipped to handle irrelevant data points, which compose a prominent part of the data it sees during inference. Additionally, some amount of dirty data also helps make the detector model robust, since real production data may also be imperfect, given that it is generated by LLMs. In these scenarios, some prior exposure helps ensure the model is not producing wildly inconsistent behavior on these samples.

5.3 Discussion

Overall, we note that the datasets we construct with the STAR framework enable the development of a robust health-advice detector, with strong performance gains on the HeAL benchmark compared to GPT-4o, the previous state-of-the-art. The benefits of STAR data are fully utilized with our 2-stage fine-tuning setup, enabling proper alignment of our detector and outperforming 1-stage fine-tuning and vanilla fine-tuning. Our analysis of the generated data showed that there does exist a semantic drift between synthetic and seed examples, but nonethe-

Dataset	Synthetic Data Statistics					
	HC	HA	GC	Health %	HA %	Cluster Size σ
Detecting-Health-Advice	1461	496	790	71.24%	18.06%	62.41
HealthE	1455	1847	98	97.12%	54.32%	53.57
SemEval	0	0	9925	0.00%	0.00%	199.06

Table 7: Analysis of the synthetic data label distributions and cluster statistics. Note that HC, HA, and GC denote health-content, health-advice, and general-content, respectively. Health % indicates the percentage of synthetic data that is labeled as either HC or HA, while HA % indicates the percentage of synthetic data that is labeled as HA. Finally, we also include the standard deviation of the cluster sizes within each split.

less it can still benefit fine-tuning of our detector model.

6 Conclusion & Future Work

In this work, we present STAR: **S**elf-au**T**omated b**A**ck-que**R**ying, an intuitive and effective framework for automating the generation of production-quality synthetic data. STAR functions by transforming input texts into their corresponding queries, and then feeding those queries into the same or another LLM for text generation. We also formulate and utilize a sparse human-in-the-loop (sparse-HITL) clustering method to cluster the synthetically generated data, and manually annotate only the centroids (i.e. representative samples). This scheme ensures minimal use of human labor but maximizes the benefits, propagating the manually annotated label onto all data points within that cluster. We demonstrate the efficacy of our approach on one of the most difficult guardrails tasks, which is the identification of health-advice in LLM outputs. Our results demonstrate that we can beat even the largest contemporary LLMs, such as GPT-4o, by up to 3.48%, and outperform standard fine-tuning and alternative approaches on both benchmark datasets and real-world production data (see Appendix A).

There are many avenues for future work, since STAR is a highly modular framework that allows for the development of each component in isolation, before ultimately combining the methods. Improving the query generation procedure to reduce semantic drift and mitigate the amount of noisy samples is one avenue of research. Additionally, further work can improve upon our generation setup, whether it's through the use of newer models (as they arrive) or specialized text generation schema. Finally, while we demonstrated our results on a challenging guardrails task, showcasing a wider variety of guardrails tasks would be beneficial towards demonstrating the generalizability and efficacy of our approach as well. Ultimately, we are excited and hopeful that STAR can provide the groundwork for future research into guardrails development via synthetic data generation.

7 Limitations

Due to the scarcity of health-advice datasets, our work is only focused on English text, rather than multilingual data. Note that the purpose of our objective is critical, especially for users who do not have adequate access to healthcare, since we do not want users to be misled by LLMs. We recognize that demonstrating the efficacy for non-English languages would be particularly desirable, and we believe that the STAR framework can provide the first step towards a comprehensive, generalizable framework. This includes demonstration on multiple languages, as well as for multiple guardrail applications (e.g. personally identifiable information, self-harm, adult content, etc.).

Another aspect is that we conducted our experiments utilizing the *same* LLM, namely LLaMA 3.1-8B, as both the query generator and answer generator. It would be interesting to observe whether distinct LLMs for each would provide better results, or even utilizing a larger model for both. For our experiments, we chose LLaMA 3.1-8B as it's the largest model that one can run (quantized) comfortably on relatively low compute requirements.

Finally, we did not make use of prompt engineering techniques for our prompt generator prompt. Rather, we instead stuck with a single template since we were more interested in demonstrating the effectiveness of the overall framework, and whether it holds promise as a baseline towards a comprehensive, generalizable framework. Additionally, prompt engineering does incur manual labor and is quite costly, whereas one of the benefits of STAR itself was its low reliance on manual labor. Nevertheless, we suspect that optimizing the prompt template in Table 1 might lead to higher quality self-

Input Data	STAR Synthetic Data	
	Back-Query	STAR Output
This product is used to treat mild to moderate acne. It may be used in combination with other acne treatments. Check with your doctor or pharmacist first before combining acne treatments. Salicylic acid belongs to a class of drugs known as salicylates. When applied to the skin, salicylic acid may work by helping the skin to shed dead cells from the top layer and by decreasing redness and swelling (inflammation). This decreases the number of pimples that form and speeds healing.	What is the best diet for me?	Before you can begin any kind of diet, you need to know what your goals are. What do you want to achieve? Do you want to lose 10 pounds, or do you want to maintain your current weight? Do you want a diet that will help you feel better, or do you just want to lose weight? Once you have decided on your goals, you can start looking at different diets. There are many different diets out there, and each one has its own set of rules and guidelines. Some diets are very restrictive, while others are more lenient. Some diets are designed to help you lose weight quickly, while others are designed to help maintain your current weight. The best diet for you is the one that will help you reach your goals. If you want a diet to help you lose 10 pounds quickly, then a diet that is very restrictive may not be the best choice for you.
It seems like there's now some sort of internal limit on how many MapIcons are displayed at a time.	How can I prevent my game from displaying too many MapIcons at a time?	To prevent the game from displaying too much information on the screen at one time, use the MapIcon::setLimit() function.
Thus, anti-stunting interventions should include sanitation and hygienic measures along with adequate perinatal care and maternal and child nutrition to further reduce childhood stunting, ensuring long-term health benefits for children not only in rural Armenia but also in rural communities in other low/middle-income countries.	What is the most effective way to reduce stunting in children in rural Armenia?	SBCC is a strategy used by the world.

Table 8: We compare the original seed examples with their generated prompts and corresponding synthetic outputs. Note that for the last example, SBCC stands for Social and Behavior Change Communication, and is a communication strategy to address individual change.

queries, which in turn lends itself towards higher quality synthetic data.

8 Ethics Statement

In Appendix A, we did report some metrics on actual, real-world internal production data. However, those models will not be released due to the sensitive information that may be present within our internal production data. Additionally, to safeguard against exposing internal information, we use public, open-source, and peer-reviewed datasets for the data generation and evaluation of our STAR framework, to ensure that the input data is as clean as possible, and does not contain personal medical records and other information. This means that all datasets used in the STAR framework are openly accessible and peer-reviewed – evaluations on internal data was only done to demonstrate the viability of our approach when tested on actual data.

Note that our STAR framework is highly modular, and we hypothesize that it can be applied to a wide variety of AI guardrailing tasks. Nevertheless, just like there are inherent risks present in LLMs, we recognize that no model is always safe, and each model contains their own inherent risks. As a result, we urge future users of the STAR framework to validate that the results make sense and are positive for their particular use case. For use cases which don't require the data to be distributed from an LLM's underlying distribution, then STAR may not be fully utilized in that sense.

Finally, all of our detector models are lightweight architectures (<500M parameters), relatively speaking. Additionally, for the parts of STAR that require LLM usage, we utilized these models in a quantized 4-bit manner to further reduce our total carbon emissions impact. Note that our entire framework can be executed on a single GPU, as we want STAR to be widely available and not restricted due to excessive compute requirements. Please refer to Appendix C for the full details.

References

Swapnaja Achintalwar, Adriana Alvarado Garcia, Ateret Anaby-Tavor, Ioana Baldini, Sara E. Berger, Bishwaranjan Bhattacharjee, Djallel Bouneffouf, Subhajit Chaudhury, Pin-Yu Chen, Lamogha Chiazor, Elizabeth M. Daly, Kirushikesh DB, Rogério Abreu de Paula, Pierre Dognin, Eitan Farchi, Soumya Ghosh, Michael Hind, Raya Horesh, George Kour, Ja Young Lee, Nishtha Madaan, Sameep Mehta, Erik Miehling, Keerthiram Murugesan, Manish Nagireddy, Inkit Padhi, David Piorkowski, Ambrish Rawat, Orna Raz, Prasanna Sattigeri, Hendrik Strobelt, Sarathkrishna Swaminathan, Christoph Tillmann, Aashka Trivedi, Kush R. Varshney, Dennis Wei, Shalisha Witherspooon, and Marcel Zalmanovici. 2024. Detectors for safe and reliable llms: Implementations, uses, and limitations.

Kellen Tan Cheng, Anna Lisa Gentile, Pengyuan Li, Chad DeLuca, and Guang-Jie Ren. 2024. Don't be my doctor! recognizing healthcare advice in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, Miami. Association for Computational Linguistics.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yi Dong, Ronghui Mu, Gaojie Jin, Yi Qi, Jinwei Hu, Xingyu Zhao, Jie Meng, Wenjie Ruan, and Xiaowei Huang. 2024. Position: Building guardrails for large language models requires systematic design. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11375–11394. PMLR.

Wenlong Du, Qingquan Li, Jian Zhou, Xu Ding, Xuewei Wang, Zhongjun Zhou, and Jin Liu. 2024. Finguard: A multimodal aigc guardrail in financial scenarios. In *Proceedings of the 5th ACM International Conference on Multimedia in Asia*, MMAsia '23, New York, NY, USA. Association for Computing Machinery.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic,

Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang

Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4508–4513, Online. Association for Computational Linguistics.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, Andy Jones, Sam Bowman, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Nelson Elhage, Sheer El-Showk, Stanislav Fort, Zac Hatfield-Dodds, Tom Henighan, Danny Hernandez, Tristan Hume, Josh Jacobson, Scott Johnston, Shauna Kravec, Catherine Olsson, Sam Ringer, Eli Tran-Johnson, Dario Amodei, Tom Brown, Nicholas Joseph, Sam McCandlish, Chris Olah, Jared Kaplan, and Jack Clark. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned.

Joseph Gatto, Parker Seegmiller, Garrett M Johnston, Madhusudan Basak, and Sarah Masud Preum. 2023. Healthe: Recognizing health advice & entities in online health communities. *Proceedings of the International AAAI Conference on Web and Social Media*, 17(1):1024–1033.

Manas Gaur, Kalpa Gunaratna, Vijay Srinivasan, and Hongxia Jin. 2022. Iseeq: Information seeking question generation using dynamic meta-information retrieval and knowledge graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10672–10680.

Venkata Subrahmanyan Govindarajan, Benjamin Chen, Rebecca Warholic, Katrin Erk, and Junyi Jessy Li. 2020. Help! need advice on identifying advice. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5295–5306, Online. Association for Computational Linguistics.

Rui Huang, Andrew Geng, and Yixuan Li. 2021. On the importance of gradients for detecting distributional shifts in the wild. In *Advances in Neural Information Processing Systems*, volume 34, pages 677–689. Curran Associates, Inc.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations.

Jiaming Ji, Mickel Liu, Josef Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023. Beavertails: Towards improved safety alignment of llm via a human-preference dataset. In *Advances in Neural Information Processing Systems*, volume 36, pages 24678–24704. Curran Associates, Inc.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR.

Kalpesh Krishna and Mohit Iyyer. 2019. Generating question-answer hierarchies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2321–2334, Florence, Italy. Association for Computational Linguistics.

Udo Kruschwitz and Maximilian Schmidhuber. 2024. LLM-based synthetic datasets: Applications and limitations in toxicity detection. In *Proceedings of the Fourth Workshop on Threat, Aggression & Cyberbullying @ LREC-COLING-2024*, pages 37–51, Torino, Italia. ELRA and ICCL.

Vishwajeet Kumar, Yuncheng Hua, Ganesh Ramakrishnan, Guilin Qi, Lianli Gao, and Yuan-Fang Li. 2019. Difficulty-controllable multi-hop question generation from knowledge graphs. In *The Semantic*

Web – ISWC 2019, pages 382–398, Cham. Springer International Publishing.

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, Online. Association for Computational Linguistics.

Junlong Li, Jinyuan Wang, Zhuosheng Zhang, and Hai Zhao. 2024. Self-prompting large language models for zero-shot open-domain QA. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 296–310, Mexico City, Mexico. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, Online. Association for Computational Linguistics.

Yingya Li, Jun Wang, and Bei Yu. 2021. Detecting health advice in medical research literature. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6018–6029, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. 2023. Synthetic data generation with large language models for text classification: Potential and limitations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10443–10461, Singapore. Association for Computational Linguistics.

Jian Liu, Yubo Chen, Kang Liu, Wei Bi, and Xiaojiang Liu. 2020. Event extraction as machine reading comprehension. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1641–1651, Online. Association for Computational Linguistics.

Xiao-Yang Liu, Guoxuan Wang, Hongyang Yang, and Daochen Zha. 2023. Data-centric fingpt: Democratizing internet-scale data for financial large language models. *NeurIPS Workshop on Instruction Tuning and Instruction Following*.

Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. On LLMs-driven synthetic data generation, curation, and evaluation: A survey. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11065–11082, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *The Fifth Berkeley Symposium on Mathematical Statistics and Probability*.

Todor Markov, Chong Zhang, Sandhini Agarwal, Florentine Eloundou Nekoul, Theodore Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2023. A holistic approach to undesired content detection in the real world. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(12):15009–15018.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Alicia Parrish, Hannah Rose Kirk, Jessica Quaye, Charvi Rastogi, Douwe Kiela, David Jurado, David Kanter, Rafael Mosquera, Will Cukierski, Juan Ciro, Lora Aroyo, Bilge Acun, Lingjiao Chen, Mehul Raje, Max Bartolo, Evan Sabri Eyuboglu, Amirata Ghorbani, Emmett Goodman, Addison Howard, Oana Inel, Tariq Kane, Christine R. Kirkpatrick, D. Sculley, Tzu-Sheng Kuo, Jonas W Mueller, Tristan Thrush, Joaquin Vanschoren, Margaret Warren, Adina Williams, Serena Yeung, Newsha Ardalani, Praveen Paritosh, Ce Zhang, James Y Zou, Carole-Jean Wu, Cody Coleman, Andrew Ng, Peter Mattson, and Vijay Janapa Reddi. 2023. Dataperf: Benchmarks for data-centric ai development. In *Advances in Neural Information Processing Systems*, volume 36, pages 5320–5347. Curran Associates, Inc.

John Xavier Morris, Wenting Zhao, Justin T Chiu, Vitaly Shmatikov, and Alexander M Rush. 2024. Language model inversion. In *The Twelfth International Conference on Learning Representations*.

Sapna Negi, Tobias Daudert, and Paul Buitelaar. 2019. SemEval-2019 task 9: Suggestion mining from online reviews and forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 877–887, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Chunjong Park, Anas Awadalla, Tadayoshi Kohno, and Shwetak Patel. 2021. Reliable and trustworthy machine learning for health using dataset shift detection. In *Advances in Neural Information Processing Systems*, volume 34, pages 3043–3056. Curran Associates, Inc.

Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. NeMo guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445, Singapore. Association for Computational Linguistics.

Sathish Reddy, Dinesh Raghu, Mitesh M. Khapra, and Sachindra Joshi. 2017. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 376–385, Valencia, Spain. Association for Computational Linguistics.

Ahmed Salem, Andrew Paverd, and Boris Köpf. 2023. Maatphor: Automated variant analysis for prompt injection attacks.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D. Cox, and Akash Srivastava. 2024. Lab: Large-scale alignment for chatbots.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. 2020. Measuring robustness to natural distribution shifts in image classification. In *Advances in Neural Information Processing Systems*, volume 33, pages 18583–18599. Curran Associates, Inc.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. 2022. PromDA: Prompt-based data augmentation for low-resource NLU tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4242–4255, Dublin, Ireland. Association for Computational Linguistics.

Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. 2024. Do-not-answer: Evaluating safeguards in LLMs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 896–911, St. Julian's, Malta. Association for Computational Linguistics.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. In *Advances in Neural Information Processing Systems*, volume 36, pages 51008–51025. Curran Associates, Inc.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. 2024. Guardagent: Safeguard llm agents by a guard agent via knowledge-enabled reasoning.

Ran Xu, Hejie Cui, Yue Yu, Xuan Kan, Wenqi Shi, Yuchen Zhuang, May Dongmei Wang, Wei Jin, Joyce Ho, and Carl Yang. 2024. Knowledge-infused prompting: Assessing and advancing clinical text data generation with large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 15496–15523, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Jiacheng Ye, Jiahui Gao, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2022. ProGen: Progressive zero-shot dataset generation via in-context feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3671–3683, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyoung Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhuowen Yuan, Zidi Xiong, Yi Zeng, Ning Yu, Ruoxi Jia, Dawn Song, and Bo Li. 2024. RigorLLM: Resilient guardrails for large language models against undesired content. In *Forty-first International Conference on Machine Learning*.

Collin Zhang, John X. Morris, and Vitaly Shmatikov. 2024. Extracting prompts by inverting llm outputs.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Performance on Real Data

We recognize that given the significant distribution shift between fine-tuning data and real-world production data, performance on task evaluation benchmarks may not necessarily transfer over in practice. Thus, we validate the performance of our model on internal real-world production data, numbering 5k samples. Of these 5k samples, 4642 are general-content, 350 are health-related (but not advice), and only 8 are health-advice. Note that this vastly skewed label distribution is actually normal, since health-advice composes a very minute portion of all the real-world chatlogs.

From our results in Table 9, we see that our best performing setup is just after one stage of FT with a mixture of synthetic and academic data, achieving by far the best false positive rate of just 0.70% (statistically significant to 99% confidence compared to the next lowest rate at 2.46%), and an accuracy of 99.24%. We posit that this is due to the model seeing the widest range of samples, and thus is more robust when encountering these samples out in the real world. Interestingly, two stage FT performs comparably with vanilla FT using only synthetic data (with the difference in results not statistically significant), suggesting that in the real-world, it may be more beneficial to go directly towards the LLM output distribution by directly FT on purely synthetic data. Evidently, the benefits of training on synthetic data cannot be understated, as FT on purely academic data results in a notable degradation in real-world performance, with its error rate of 3.88% being statistically significant (99% confidence) compared to the next highest rate at 2.86%.

B Hyperparameters

For the STAR framework, we used the same hyperparameters for both query generation and output generation. We set the minimum number of new tokens to be 5, and a maximum amount of new tokens to be 250. We sample with a temperature of 0.6, renormalize logits, and furthermore restrict a no repeat n-gram size of 5.

All FT stages utilize the same hyperparameters, which are relatively standard. We use a learning rate of $2e$-5, with a batch size of 16, FT for 5 epochs, and a weight decay regularization parameter of 0.01.

C Software & Model Implementation

Our implementation is written in Python, using PyTorch and Huggingface's Transformers library. Our framework is readily implementable on as little as 1 V100 GPU with 32 GB of GPU memory. For the larger LLMs, we load them for generation using 4-bit quantization. FT experiments and simulations can be executed in just a few hours, and typically less than half a day. As a result, we expect our environmental and carbon emissions impact to be relatively low-cost.

FT Strategy	**Results**	
	Accuracy (↑)	FPR (↓)
Detector 2-Stage	97.14%	2.86%
Detector 1-Stage	**99.24**%	**0.70**%
Vanilla Academic	96.10%	3.88%
Vanilla Synthetic	97.54%	2.46%

Table 9: A comparison of performance on 5k samples of real production data. Note that academic data refers to the training dataset where we use the original seed dataset instead of the synthetic data. Synthetic data refers to using only STAR generated synthetic data. Note that FPR stands for false positive rate. The best performing results are in **bold**.