Controlled Automatic Task-Specific Synthetic Data Generation for Hallucination Detection

Yong Xie
Amazon
Seattle, USA
yonxie@amazon.com

Karan Aggarwal
Amazon
Seattle, USA
kagg@amazon.com

Aitzaz Ahmad
Amazon
Seattle, USA
aitzaza@amazon.com

Stephen Lau
Amazon
Seattle, USA
lausteph@amazon.com

ABSTRACT

We present a novel approach to automatically generate task-specific synthetic datasets for hallucination detection. Our approach features a two-step generation-selection pipeline, where the generation step integrates a hallucination pattern guidance module and a language style alignment module. Hallucination pattern guidance makes it possible to curate synthetic datasets covering the most important hallucination patterns specific to target applications. Language style alignment improves the dataset quality by aligning the style of the synthetic dataset with benchmark text. To obtain robust supervised detectors from synthetic datasets, we also propose a data mixture strategy to improve performance robustness and model generalization. Our supervised hallucination detectors trained on synthetic datasets outperform in-context-learning (ICL)-based detectors by a large margin. Our extensive experiments confirm the benefits of our two-staged generation pipeline with cross-task and cross-hallucination pattern generalization. Our data-mixture-based training further improves generalization and the robustness of hallucination detection.

CCS CONCEPTS

• **Computing methodologies → Natural language generation**; **Model verification and validation**; **Simulation evaluation**.

KEYWORDS

Synthetic data generation, hallucination detection, large language model

1 INTRODUCTION

The ability of large language models (LLMs) to generate human-like text [18, 20, 22] has advanced significantly in recent years, enabling a wide range of applications, from document summarizers [8, 10] to coding assistants [27]. However, one of the key challenges in deploying these models is the risk of hallucinations — the generation of plausible but factually incorrect information. Hallucinations can occur when the model makes up details that are not grounded in the input or the given context, leading to the generation of misinformation or nonsensical outputs. The tendency to hallucinate raises concerns about the safety and reliability of LLMs in critical domains like finance and healthcare.

While there is a debate in the LLM research community around the categorization of various types of hallucinations [6, 29], ultimately the type of hallucination is task-dependent. For example, in an open-ended question-answering scenario, we are likely to see more factual hallucinations versus code generation tasks, where we are more likely to see logical hallucinations [14]. The importance attributed to each hallucination type is again task-dependent. For instance, hallucinations producing factually incorrect answers can erase a customer's trust in chatbot-style AI assistants. Therefore, it is critical to customize the hallucination evaluation and detection to the specific task and application.

Post-hoc hallucination detection approaches detect hallucinations once they have been generated by the LLM. *Post-hoc* hallucination detectors are typically built by training a classifier on observed hallucinations or on open-sourced hallucination datasets. However, in order to build such detectors, we need access to observed hallucinations, which are harder to get before the solution is actually being used. Since hallucinations are task-specific, detectors trained on open-sourced datasets may not be relevant. Hence, there is a need for task-specific hallucination detectors pre-production to ensure a trustworthy solution for users.

To facilitate the development of customized hallucination detection, we propose a generic approach to curating synthetic datasets for training hallucination detectors (hallucination datasets), as shown in Figure 1. Our approach features a two-step *Generation-Selection* pipeline: we first generate a group of hallucinated candidates for a given input through an LLM (generator) and then select the best candidate through an LLM (judge) based on given criteria. Moreover, we propose two design features in the generation step to customize hallucination generation and improve dataset quality: *Hallucination Pattern Guidance* and *Language Style Alignment*.

Hallucination pattern guidance is motivated by the need for task-specific hallucinated samples. Starting from a set of predefined hallucination patterns using a little human effort, we prompt the generator to generate hallucinated samples conforming to the given patterns. Language style alignment is designed to align the text characteristics of hallucinated samples with the style of non-hallucinated expected LLM responses. Recognizing that analyzing language style requires expertise in linguistics, we propose a hierarchical *Language Style Discovery* algorithm. This leverages LLMs to analyze the language style and distill the styles into a small feature set. These style features are then transformed into guidelines and injected into the generator prompt to govern hallucination generation. *This refinement helps align generations with non-hallucinated text, making our hallucinated dataset more challenging and, hence, our hallucination detectors more powerful.*

We conduct experiments on three conversational benchmarks by generating synthetic hallucination datasets and training supervised detectors. Our hallucination detector achieves an F1 score of 0.938 on average over three benchmarks and six different generators, outperforming the in-context learning LLM detectors by a large margin of 0.325. It implies that dedicated detectors trained on synthetic datasets are powerful and cost-effective options for *post-hoc* hallucination detection.

Due to the fast-paced nature of LLM development and application development, there is an increasing demand for better generalization abilities of hallucination detectors so that 1) detectors trained on a dataset generated by one LLM can be used on generation of other LLMs; 2) detectors trained on a portfolio of hallucination patterns can generalize on unseen patterns; and 3) detectors trained on one task can be transferred to other tasks. Detectors with better generalization decouple themselves from backbone generator LLMs and tasks, reducing the burden of application development and providing extra flexibility.

With this motivation, we propose a simple but effective data mixture strategy to obtain a more diverse training corpus by merging synthetic datasets generated by multiple LLMs. We investigate the efficacy of data mixture and the generalization ability of supervised detectors trained on synthetic datasets through three pillars: cross-generator generalization, cross-pattern generalization, and cross-task generalization.

Our results show that supervised detectors trained on datasets generated by one LLM deliver great performance on other LLMs' generations. Besides, the detectors excel at hallucinations of unseen patterns, with only a slight performance drop. Moreover, supervised detectors exhibit strong cross-task generalization ability, showing only a slight performance drop when evaluated on different benchmark tasks. The data mixture significantly increases the cross-generator generalization ability by reducing the performance gap between evaluation on in-generator datasets and out-of-generator datasets.

To summarize, we make the following contributions:

- We propose a novel approach for customized hallucination detection in the absence of observed hallucinations useful for pre-production settings by aligning the generated hallucinations with the non-hallucinated text's language style;
- Through extensive experiments, we show the cross-task and cross-pattern generalization capabilities of our proposed approach; and
- We introduce a more generalized and robust hallucination detection strategy using a mixture of synthetic data generation through multiple LLMs.

2 METHODOLOGY

Problem Setting. In this work, we focus on settings where the hallucinations have not been observed, i.e., before the application has been put into production. The only thing we assume is the presence of a benchmark dataset, which is a set of non-hallucinated input-output pairs either from humans or from an LLM to build our hallucination generation pipeline. We rely on human judgment to provide hallucination patterns that need to be detected. Our objective is to create a synthetic dataset that contains both hallucinated and non-hallucinated input-output pairs. Hallucination detection is formulated as a binary classification task: given an input and an LLM output, a detector determines whether the LLM output is hallucinated with respect to the input.

Our proposed approach features an automatic *Generation-Selection* pipeline with *Hallucination Pattern Guidance* (HPG) and *Language Style Alignment* (LSA). The generation-selection mechanism consists of a generation step to obtain a set of candidate hallucinated samples and a selection step to pick the most plausible one, which ensures the generation quality. HPG and LSA are two versatile modules integrated into the generation step. The former guides the generator to produce task-specific synthetic datasets covering the most relevant hallucination patterns for target tasks, and the latter aligns the generations with non-hallucinated text in order to improve data quality.

2.1 Generation-Selection Pipeline

While using synthetic data for hallucinations is becoming more common, the quality of the hallucination data can be low, especially for automatic approaches without human intervention. To resolve the issue, we adopt the two-step *first-generate-then-select* design [12] to ensure generation quality. Specifically, two LLMs (not necessarily the same) act in the roles of generator and judge separately. The generator produces a set of hallucinated outputs per input according to predetermined patterns. The judge scores the hallucinated candidates by given criteria, and the one with the highest score is selected. This step improves generation quality by selecting the best among the group of candidates.

Generator. Generator is a prompted LLM performing the task of hallucinated sample generation. We utilize LLMs to generate hallucinated data, as they are proven to generate high-quality text while following instructions. The key here is to properly design the prompts for hallucination guidance and language style alignment. Besides, it is important to carefully specify the persona in the system prompt to work around the safety policies in place that prevent LLMs from generating hallucinations. We adopt the chain-of-thought (CoT) [24] prompt and ask the generator to provide rationale for the generated samples, inspired by Peng et al. [21]. Our generator prompt is structured as follows: the prompt starts with a definition of persona customized for target tasks, which is followed by a section of HPG consisting of the pattern description and one demonstration example. The next is the LSA section, which comprises itemized guidelines for text generation. The prompt ends with an input and brief instructions on the output format. The details of prompts are deferred to Section 2.2 and Section 2.3.

Judge. A judge is a prompted LLM performing the task of evaluating the quality of hallucination candidates according to the given criteria. We prompt the judge to score the candidates on a scale of 1 to 10, and the candidate with the highest score is then selected as the hallucinated output for the given input. We follow this scoring mechanism instead of directly selecting the best candidate out

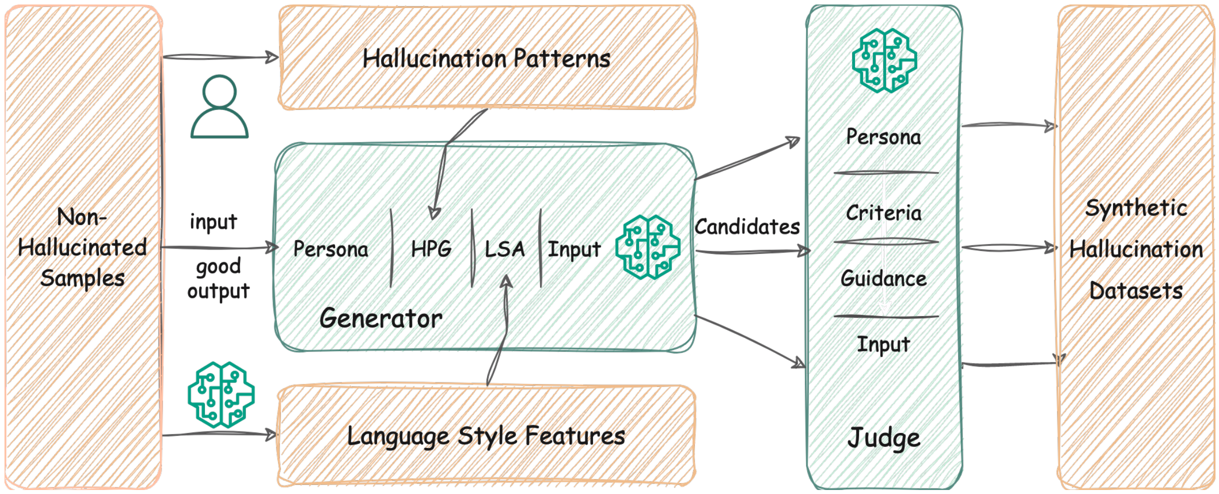

Figure 1: Automatic generation pipeline. We use non-hallucinated samples to generate the synthetic hallucination dataset with two inputs to the generator: human defined Hallucination Patterns and Language Style Features with language style features, like text tone. These are used by the Generator LLM to generate hallucinated samples, which are then judged by a LLM Judge to finally select the most plausible hallucinated samples.

of the generated candidates because the scoring approach is less prone to LLMs' positional bias [23]. Moreover, we also adopt the CoT prompt to generate rationale for the scores to improve accuracy. The judge prompt (see Appendix A) consists of a customized persona for target tasks, an evaluation criteria section, a guideline section, and an input section. The evaluation criteria are as simple as '*the more hallucinated the content is, the higher score should be given; the more plausible the output is, the higher score should be given*'. The guideline section consists of one demonstration for each hallucination pattern. The input section comprises the model input, an input text, a set of hallucinated candidates, and instructions on the output format.

2.2 Hallucination Pattern Guidance

Hallucination patterns depend on the domains, tasks, contexts, and questions asked. As such, it is critical to curate the synthetic hallucination datasets in a controlled manner such that the hallucination patterns align with model behaviors in production. Our approach achieves task-specific generation by introducing a section of *Hallucination Pattern Guidance* (HPG) in the generator prompt.

The HPG module needs a set of predetermined hallucination patterns. Each pattern consists of a short description and a demonstration, including an input, a non-hallucinated output, and a hallucinated output of the pattern. With the HPG section, the generator follows the instruction to generate hallucinated candidates in a controlled rather than open-ended manner. Our approach relies on human judgment to determine the hallucination patterns in their target applications. Such patterns can be generic, *e.g.*, overconfidence and non-factuality, or task-specific, such as confusing between entities in response. Practitioners have the flexibility to include the most common and relevant hallucinations by simply writing out descriptions and curating demonstrations. Moreover, it is worth noting that the predefined pattern can go beyond the conventional definition of hallucination and include any undesired LLM behaviors that we want to detect. For example, in the experiment to be presented in Section 3, we include the pattern of nonsensical responses, where the generated responses bear no meaning in the context.

2.3 Language Style Alignment

LLM generations are known to be biased, lack diversity, and misaligned with human writings [21]. As a result, a synthetic dataset created by one LLM might be sufficiently distant from human writings or the generation of other LLMs. Since our approach leverages the golden non-hallucinated outputs, any salient distinctions in language styles like length of text or tone between hallucinated output and non-hallucinated output can be exploited as shortcuts during supervised training. Therefore, it is critical to ensure the synthetic datasets resemble the language characteristics of golden, non-hallucinated text.

To this end, we propose a *Language Style Alignment* (LSA) module to align the characteristics of generated text with benchmark text. LSA is achieved by a prompt section in the generator prompt, which includes a group of itemized guidelines (see Appendix B) on the desired language style features, e.g., writing style, length, tone, etc. With the LSA, the generator follows the instructions and generates the hallucinated candidates in a controlled manner. In general, language-style-aligned hallucinated outputs should be more challenging to detect as they are more similar to non-hallucinated outputs, except for the hallucinated content.

The challenge lies in obtaining the language style features. For one, the benchmark dataset to be aligned with can be too large to manually analyze the text characteristics. Besides, it requires expertise in linguistics to properly analyze and summarize the language style features. To overcome the challenge, we propose a *Language*

Style Discovery algorithm that leverages LLMs to summarize and consolidate the language style features. Specifically, the benchmark dataset is first partitioned into batches of proper size and then fed into prompted LLMs to analyze language style features. A group of language style features is produced for each batch, and language style features are merged together to form a language feature set. Then, we partition the language feature set into batches and ask a LLM to consolidate the features in each batch, which results in a smaller language feature set. The procedure continues until the desired number of language style features are obtained.

2.4 Data Mixture

Different LLMs exhibit different bias, diversity, and misalignment issues due to the distinctions in the pre-training corpus and preference alignment. As a result, a synthetic dataset created by one LLM might be sufficiently distant from the ones generated by other LLMs, such that detectors trained on the synthetic dataset do not generalize. Such generalizations for hallucination detection become more and more important due to the increased restrictions on the usage of LLM-generated data. For example, OpenAI restricts users from developing third-party models using the data generated by ChatGPT. For applications built with ChatGPT, developers can only resort to open-sourced LLMs with a permissive license to curate synthetic datasets to train detectors, hoping the detectors will generalize. Also, the landscape of LLM development is changing rapidly. LLM users may switch to other LLMs for various reasons. Hallucination detectors with strong generalization ability reduce the dependency on specific LLMs and thus enable faster developments and experiments.

Motivated by the need for better generalization, we propose *Data Mixture*, a simple-yet-effective scaffolding strategy, to improve generalization and performance robustness. Specifically, we run the generation-selection pipeline with multiple LLM generators and mix the resulting synthetic dataset to increase the training corpus diversity and mitigate bias. Note that the data mixture is independent of the pipeline design since it is applied to the resulting synthetic datasets.

3 EXPERIMENTS

3.1 Benchmarks

We consider three conversational benchmarks to conduct the empirical experiments. For each benchmark, we randomly sample 1000 data points from the dataset as the golden non-hallucinated samples and apply the data generation pipeline to selected samples to curate synthetic datasets.

OpenDialKG [16]. It is a dataset of task-oriented conversations between two crowdsourcing agents engaging in a dialog about a given topic. The dataset contains conversations for two tasks: recommendation and chit-chat. The conversations under recommendation cover entities related to movies (titles, actors, directors) and books (titles, authors). The chit-chat conversations cover entities related to sports (athletes, teams) and music (singers).

ReDial [13]. It is a large-scale dataset consisting of real-world dialogues centered around recommendations. It consists of over 10,000 conversations centered around the theme of providing movie recommendations.

SaleBot [2]. This dataset focuses on the conversations starting with open-domain social chatting and then gradually transitioning to task-oriented purposes. It is generated automatically without human intervention.

3.2 Setup

We run experiments with six LLMs from three model families available in AWS Bedrock, including Claude3-Sonnet, Claude3-Haiku [1], llama2-13B, llama2-70B [22], Mixtral-8×7B Instruct, and Mixtral-Large [7]. We use the same LLM for generation and selection and refer to the resulting dataset under the corresponding LLM's name. We use Claude3-Sonnet to analyze the datasets and discover a set of language style features (see Appendix B).

We manually curate three hallucination patterns for our experiments, including nonsensical response, inconsistent entity, and irrelevant content. Each hallucination pattern is associated with a demonstration example. Details on the hallucination patterns are deferred to Appendix C. We generate three hallucination candidates per sample for each pattern. As a result, each synthetic dataset contains 4000 samples, including 1000 non-hallucinated responses and 3000 hallucinated responses.

The parameters for the generation pipeline are summarized as follows: For the generator, we configure the temperature to be 1 for diverse generations, and we set the temperature to be 0 for less randomness in the selection step while $top\ p$ is set to be 1 throughout the experiments. We also evaluate the performance of in-context learning hallucination detectors by directly prompting the LLMs to determine whether an output is hallucinated. For those experiments, we set the temperature to be 0 to control the randomness.

When fine-tuning supervised detectors on synthetic datasets, we adopt the RoBERTa [15] as the backbone model across experiments to keep it consistent. While we could fine-tune LLMs as detectors, we use RoBERTa for low complexity in line with existing literature [6]. The learning rate is set at 10^{-5} with a linear decay scheduler. We fine-tune the model for three epochs with a batch size of 64. Each synthetic dataset is partitioned into train/validation/test subsets in the ratio of 7:1:2, and the best checkpoint is picked based on the loss on the validation dataset.

For the experiment with the data mixture, we evaluate two mixture strategies based on the generator portfolios: model family mixture and model size mixture. The former strategy combines synthetic datasets generated by LLMs in the same model family (Claude3, Llama2, Mixtral), and the latter combines the datasets generated by the larger models in each family (Large Combo: a mixture of Claude3-Sonnet, Llama2-70B, and Mixtral-Large) and smaller models in each family (Small Combo: a mixture of Claude2-Haiku, Llama2-13B, and Mixtral-8×7B). We mix synthetic datasets through random sampling while controlling the dataset size for the sake of fair comparison.

3.3 Evaluation

Considering the objective of hallucination detection, we evaluate our approach through two branches of metrics. Firstly, we quantify

Table 1: Corpus distance between non-hallucinated samples and hallucinated generations. Reported are average values over six generators.

	FID	Mediod	Zipf	Ave
OpenDialKG (w/o LSA)	0.474	0.318	0.081	0.291
OpenDialKG (w/ LSA)	0.420	0.278	0.072	0.256
ReDial (w/o LSA)	0.377	0.230	0.075	0.227
ReDial (w/LSA)	0.333	0.205	0.065	0.201
SalesBot (w/o LSA)	0.672	0.326	0.283	0.427
SalesBot (w/ LSA)	0.622	0.299	0.271	0.398

the detector performance with standard metrics for binary classification, such as the F1 score. We run the supervised detectors on the test datasets generated by the same LLMs as the training dataset (*in-generator*), and the results reveal whether the hallucination is detectable and how good the detectors are.

Secondly, we also compute the difference between the performance on in-generator dataset(s) and *out-of-generator* datasets, i.e., datasets generated by LLMs other than the one generating the training dataset. It quantifies a detector's generalization ability across generators. Besides, we also evaluate the detector's ability to generalize on unseen hallucination patterns by using two hallucination patterns to train detectors (*in-pattern*) and using one as a holdout pattern dataset (*out-of-pattern*). Furthermore, to further investigate the performance generalization ability across tasks, we also evaluate their performance under the scenario where the training benchmark task is different from the testing benchmark task (*out-of-task*). For performance robustness, we use the metric of the standard deviation of the metrics recorded on out-of-generator datasets (out-of-generator std).

3.4 Synthetic Dataset Analysis

Language style alignment is designed to align the generation style with non-hallucinated samples. We first quantitatively examine the efficacy of LSA by gauging the distance between synthetic hallucinated responses and non-hallucinated responses. We utilize three metrics to quantify the distance between two corpora: Fréchet Inception Distance (FID) [4], Zipf [5], and Medoid [9]. FID quantifies the corpus distance through the Wasserstein distance between densities by fitting a continuous multivariate Gaussian to the SentenceBERT text embeddings of corpora. Zipf gauges the distance using the absolute difference between two Zipfian coefficients fitted on two corpora. Lastly, Medoid quantifies the cosine distance between corpora centroids.

Distances between hallucinated responses and non-hallucinated responses are reported in Table 1. The distances are consistently smaller between the two types of responses with LSA, demonstrating that LSA brings the hallucinated samples closer to the real human non-hallucinated samples. Specifically, hallucinated responses are 12.0%, 11.5% and 6.8% closer to the good responses on average for OpenDialKG, ReDial, and SalesBot, respectively. Besides, we also observe that the distances are not equivalent for the three benchmarks, with the responses being most distant for SalesBot and least distant for ReDial. The reduced corpus distance between hallucinated and non-hallucinated responses implies that LSA guides the generation to resemble the language features of non-hallucinated samples. This resemblance makes it more difficult for a detector to focus on trivial language features irrelevant to detecting hallucinations.

Table 2: Hallucination detection performance by category. Reported are average F1 scores over six generators and 5 data mixture strategies.

	Entity Incon.	Non. Resp.	Irre. Cont.	Overall
OpenDialKG				
ICL	0.670	0.587	0.551	0.638
Vanilla	**0.858**	**0.932**	**0.947**	**0.920**
Mixture	0.829	0.931	0.948	0.908
ReDial				
ICL	0.615	0.581	0.532	0.606
Vanilla	**0.849**	**0.969**	**0.975**	**0.932**
Mixture	0.793	0.957	0.966	0.913
SalesBot				
ICL	0.616	0.559	0.505	0.595
Vanilla	**0.932**	**0.978**	**0.987**	**0.963**
Mixture	0.904	0.973	0.984	0.950

3.5 Hallucination Detection Results

3.5.1 In-Generator Hallucination Detection. Table 2 reports the average of F1 scores recorded by in-context learning detectors (ICL) and supervised detectors (Vanilla and Mixture). For more detailed results by individual models, refer to Table 8 in appendix. For ICL detectors, the LLM is assessed on the synthetic dataset generated by the same LLM. For supervised detectors, the reported performance is in-generator performance—the detectors are assessed on the test dataset generated by the same LLM as the training dataset. We find that ICL detectors still face significant challenges identifying hallucinations generated by themselves; the average F1 score is only 0.613 across the board. Fine-tuned detectors, in contrast, exhibit stronger performance consistently. For vanilla fine-tuned detectors trained on synthetic datasets generated by specific LLMs, the average F1 scores for six models are 0.920, 0.932, and 0.962 on OpenDialKG, ReDial, and SalesBot, respectively.

Supervised detectors with data mixture perform slightly worse than vanilla supervised detectors on average (0.938 versus 0.912). The same pattern is observed in each hallucination pattern category and benchmark task. Since we control the sample size, the degraded performance suggests that synthetic datasets generated by different LLMs are distant in distribution, so merging datasets without scaling the sample size is at the cost of model performance. Interestingly, the conclusion holds even for datasets generated by the LLMs in the same family (see Table 8 in Appendix D). We conjecture that model size plays a role in generation distributions, as models in the same family usually share the training corpus.

3.5.2 Out-of-Generator Generalization. The results in the previous section on data-mixture suggest that performance suffers because of differences in the data distribution between text generated from different LLMs, implying a generalization issue with detectors struggling with out-of-distribution text. To test this, we perform an explicit out-of-generator testing—training detectors based on data generated by LLMs other than the one that generates the test hallucination dataset. Table 3 reports the results of the in-generator and out-of-generator performances and the performance gap (Δ).

As shown in the table, supervised detectors usually perform worse on out-of-generator datasets than on in-generator dataset(s) across the board. The vanilla supervised detectors record an average F1 score of 0.846 on out-of-generator datasets across three benchmark tasks, 9.8% lower than the average F1 score recorded on in-generator datasets. Despite the slight generalization issue, supervised detectors still outperform ICL detectors. Moreover, we find that supervised detectors trained with mixed data outperform vanilla supervised detectors on out-of-generator datasets by 0.032 on average. As a result, the data mixture increases the generalization ability by significantly reducing the performance gap to -0.045 from -0.092, half of which is attributed to the performance improvement on out-of-generator datasets.

Performance robustness is quantified by the standard deviation of out-of-generator performance. The average out-of-generator standard deviation for vanilla supervised detectors is 0.095, equivalent to 11.2% of the out-of-generator mean. Supervised detectors with data mixture achieve a smaller out-of-generator standard deviation; the average is 0.065, equivalent to 7.4% of the out-of-generator mean. The results confirm that the data mixture strategy improves the model's robustness when transferring supervised detectors across LLMs.

Looking into the details, supervised detectors' generalization abilities diverge—detectors trained on Mixtral-generated datasets generalize better than detectors trained on other datasets, but the detectors trained on Llama2-generated datasets are consistent laggards. Besides, it is shown that detectors trained on datasets generated by more powerful models (larger models) generalize better in general. Since the generalization ability of supervised detectors is mainly determined by the dataset quality, we believe that the difference in generalization ability reflects an LLM's inherent generation bias and instruction-following capability. Detailed results are deferred to Table 9 in Appendix D.

3.5.3 Out-of-Pattern Generalization. To further test the generalization abilities of creating detectors using our approach, we further test the detection capability on hallucination pattern on which the detector was not trained (out-of-pattern). Average F1 results on unseen hallucination patterns are reported in Table 4. We find that supervised detectors perform worse on unseen patterns on average, but the gap is small—supervised detectors still outperform in-context learning detectors by a large margin. Besides, supervised detectors generalize worst on the hallucination of entity inconsistency but generalize best on the hallucination of nonsensical responses. It suggests that the generalization performance does not solely depend on the pattern difficulty. Moreover, we also find that the cross-pattern generalization ability diverges as well—detectors trained on datasets generated by more powerful models (larger models) generalize better in general, in line with the observation on out-of-generator performance. Detailed results and discussion are deferred to Table 10 in Appendix D.

3.5.4 Out-of-Task Generalization. There is a practical use-case to use existing supervised detectors trained on one task to a new task to reduce the development burden. An ideal hallucination detector should generalize well to other tasks when the hallucination patterns are similar. We investigate this cross-task generalization ability by training supervised detectors on one benchmark task and evaluating them on others.

The average performance of vanilla detectors is reported in Table 5. It is clear that supervised detectors' performance drops slightly when evaluated on out-of-task datasets, but the difference is small

Table 3: Cross-generator generalization. This table reports supervised detectors' performance on the test dataset generated by the same generator as its training data (In-Generator) and by other generators (Out-of-Generator). Δ is the performance gap between in-generator (IG) performance and out-of-generator (OG) performance. OG Std is the performance standard deviation on out-of-generator datasets. Reported are based on F1 scores. Bold are preferred performance.

	IG (Mean)	OG Mean	Gap (Δ)	OG Std
OpenDialKG				
Vanilla	**0.920**	0.813	-0.107	0.122
Mixture	0.908	**0.859**	**-0.048**	**0.079**
ReDial				
Vanilla	**0.932**	0.830	-0.102	0.090
Mixture	0.913	**0.869**	**-0.044**	**0.063**
SalesBot				
Vanilla	**0.963**	0.895	-0.068	0.073
Mixture	0.950	**0.907**	**-0.042**	**0.054**

Table 4: Out-of-Pattern Performance. Cross-pattern generalization. This table compares the performances on hallucination patterns when they are seen and unseen. For rows of In-Pattern, the patterns in columns are included in the training data but are excluded for rows of Out-of-Pattern (Out-of-Ptn). Reported are average F1 scores across detectors.

	Enty Incon.	Non. Resp.	Irre. Cont.	Overall
OpenDialKG				
Vanilla IP	0.858	0.932	0.947	0.920
Vanilla OP	0.451	0.932	0.931	0.771
Mixture IP	0.829	0.931	0.948	0.908
Mixture OP	0.762	0.865	0.896	0.841
ReDial				
Vanilla IP	0.849	0.969	0.975	0.932
Vanilla OP	0.719	0.965	0.951	0.879
Mixture IP	0.793	0.957	0.966	0.913
Mixture OP	0.773	0.917	0.94	0.877
SalesBot				
Vanilla IP	0.932	0.978	0.987	0.963
Vanilla OP	0.751	0.975	0.977	0.901
Mixture IP	0.904	0.973	0.984	0.950
Mixture OP	0.822	0.935	0.949	0.902

Table 5: Cross-task generalization. This table reports the average detector performance trained and evaluated on different benchmark tasks. Column names denote the task of training dataset, and row names represent the task of testing dataset. Train and test datasets are generated by same LLMs. Reported are average F1 scores over the datasets generated by six LLM generators.

	Train Set			
Test Set	OpenDialKG	ReDial	SalesBot	Ave
Vanilla				
OpenDialKG	0.920	0.869	0.874	0.888
ReDial	0.887	0.932	0.830	0.883
SalesBot	0.863	0.878	0.963	0.901
Mixture				
OpenDialKG	0.908	0.876	0.862	0.882
ReDial	0.884	0.913	0.823	0.873
SalesBot	0.816	0.883	0.950	0.883

compared with in-task performance (diagonal terms)—supervised detectors outperform in-context-learning detectors by a large margin under this setting. It indicates that supervised detectors are strong alternatives to in-context-learning detectors on new tasks for hallucination detection.

3.6 Ablation Study

The central conjecture of our method is that LSA and HPG generate non-trivial hallucinations, which are more aligned with non-hallucinated samples. The direct implication of this conjecture is that *it would be harder to detect hallucinations generated using LSA and HPG than without them*, as these hallucinations are more similar to non-hallucinated responses in *language style*.

We test out this conjecture in Table 6 by doing an ablation on LSA and HPG components. Note that values on the diagonal are higher as expected since the train and test sets are more similar. We observe that the average performance on test hallucinations generated with both LSA and HPG is much lower than the ones w/o LSA or w/o HPG, supporting the conjecture that the synthetic samples become easier to detect without LSA and HPG. Particularly, w/o HPG, the generated hallucinations become too trivial for the detector to detect, with an average F1 of over 0.973 versus 0.908 w/ HPG across three benchmarks. LSA also makes the hallucinations harder to detect, though with a lower effect compared to HPG (0.917 average F1 w/o versus 0.908 w/ LSA).

Moreover, the detector performance also provides a lens to examine the training data quality by comparing the F1 scores in each column. Specifically, detectors trained on datasets without LSA underperform the ones with LSA (LSA + HPG) consistently across all datasets (except on test data w/o LSA) and benchmarks. Similarly, detectors trained on datasets without HPG record much lower performance compared with the ones with HPG (LSA + HPG) consistently. It suggests that synthetic data created using LSA and HPG possesses superior quality, resulting in more effective supervised detectors. These results provide strong evidence for our conjecture that LSA + HPG generate more difficult and high-quality hallucinations.

Table 6: Ablation study. This table reports the detector performance trained and evaluated on datasets generated with different modules. Column names denote the pipeline setup used to generate training datasets, and row names represent the pipeline setup for test datasets.

	Train Set			
Test Set	LSA + HPG	w/o LSA	w/o HPG	Ave
OpenDialKG				
LSA + HPG	0.920	0.911	0.852	0.894
w/o LSA	0.914	0.945	0.845	0.901
w/o HPG	0.946	0.944	0.976	0.955
ReDial				
LSA + HPG	0.932	0.922	0.839	0.898
w/o LSA	0.932	0.945	0.822	0.900
w/o HPG	0.982	0.973	0.993	0.983
SalesBot				
LSA + HPG	0.963	0.956	0.878	0.932
w/o LSA	0.975	0.977	0.898	0.950
w/o HPG	0.978	0.975	0.988	0.980

4 RELATED WORK

A bank of benchmarks has been curated for hallucination detection and evaluation recently. Pal et al. [19] proposes a hallucination benchmark that emphasizes challenges specific to LLMs in the medical domain. The benchmark consists of multiple-choice questions from various countries focusing on reasoning ability and memory ability. Muhlgay et al. [17] introduces a method for automatically creating hallucination benchmarks by perturbing factual statements. BAMBOO [3] and ScreenEval [11] are two benchmarks focusing on hallucination detection in the context of long texts. Rather than focusing on sentence-level hallucination detection, PHD is a benchmark designed for passage-level detection [25]. The most similar work to ours is HaluEval, which uses ChatGPT to create a task-specific hallucination benchmark for four tasks [12]. In contrast, our work is designed to be generic to generate customized hallucination datasets for any task or domain.

Since LLMs are now important tools for synthetic generations, many researches extend the naive approach to address the bias and diversity issues observed in LLM generations. Yu et al. [28] demonstrates that attributed prompts (specifying attributes like length and style) outperform naive prompts in terms of the resulting model's performance. Peng et al. [21] proposes Chain-of-Thoughts Attribute Manipulation (CotAM) to curate datasets from LLMs through few-shot learning. The motivation behind the approach is to create a dataset with changes only in the attribute targeted by the task. PROGEN utilizes the feedback from downstream models to guide generations via in-context examples in an iterative manner [26]. Our work extends the attribute manipulation approaches by automatically discovering the language styles by LLMs. Besides, our method degenerates to the simple baseline (SimPrompt) when LSA module is removed and the AttrPrompt when LSA is replaced with attribute guidance [28].

5 CONCLUSION

In this paper, we propose a generic automated approach to generate synthetic datasets for training hallucination detectors. Our experiment results show that in-context learning detectors are only slightly above chance, and supervised detectors trained on synthetic datasets outperform in-context learning detectors by a large margin. Our ablation study reveals that Hallucination Pattern Guidance (HPG) reduces the inherent bias by explicitly controlling the patterns of hallucination, increasing the hallucination detectors' generalization and robustness. Our method's generalization capability is shown by a high out-of-pattern and cross-task hallucination detection performance. The data mixture strategy from multiple LLMs achieves the same effect at a slight cost of in-generator performance under the same data budget constraint. Moreover, Language Style Alignment (LSA) is an effective strategy to increase the quality of synthetic data by automatically discovering language style features in non-hallucinated benchmark samples to guide the generators.

To conclude, our approach is shown to be a versatile framework for curating task-specific synthetic hallucination datasets for building *post-hoc* hallucination detectors. It contains an effective procedure to detect non-trivial hallucinations, using language style discovery and hallucination pattern customization to make detectors generalized and robust. We believe it paves the way for building low-effort, customized hallucination detection models.

REFERENCES

[1] Anthropic. 2024. *The Claude 3 Model Family: Opus, Sonnet, Haiku.* Technical Report. Anthropic. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf

[2] Ssu Chiu, Maolin Li, Yen-Ting Lin, and Yun-Nung Chen. 2022. SalesBot: Transitioning from Chit-Chat to Task-Oriented Dialogues. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 6143–6158. https://doi.org/10.18653/V1/2022.ACL-LONG.425

[3] Zican Dong, Tianyi Tang, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2023. BAMBOO: A Comprehensive Benchmark for Evaluating Long Text Modeling Capacities of Large Language Models. *CoRR* abs/2309.13345 (2023). https://doi.org/10.48550/ARXIV.2309.13345 arXiv:2309.13345

[4] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6626–6637. https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html

[5] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview.net. https://openreview.net/forum?id=rygGQyrFvH

[6] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *CoRR* abs/2311.05232 (2023). https://doi.org/10.48550/ARXIV.2311.05232 arXiv:2311.05232

[7] 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, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[8] Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901* (2024).

[9] George Kour, Samuel Ackerman, Orna Raz, Eitan Farchi, Boaz Carmeli, and Ateret Anaby-Tavor. 2022. Measuring the Measuring Tools: An Automatic Evaluation of Semantic Metrics for Text Corpora. *CoRR* abs/2211.16259 (2022). https://doi.org/10.48550/ARXIV.2211.16259 arXiv:2211.16259

[10] Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559* (2023).

[11] Barrett Martin Lattimer, Patrick Chen, Xinyuan Zhang, and Yi Yang. 2023. Fast and Accurate Factual Inconsistency Detection Over Long Documents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 1691–1703. https://doi.org/10.18653/V1/2023.EMNLP-MAIN.105

[12] Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 6449–6464. https://aclanthology.org/2023.emnlp-main.397

[13] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards Deep Conversational Recommendations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 9748–9758. https://proceedings.neurips.cc/paper/2018/hash/800de15c79c8d840f4e78d3af937d4d4-Abstract.html

[14] Fang Liu, Yang Liu, Lin Shi, Houkun Huang, Ruifeng Wang, Zhen Yang, and Li Zhang. 2024. Exploring and Evaluating Hallucinations in LLM-Powered Code Generation. *arXiv preprint arXiv:2404.00971* (2024).

[15] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[16] Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable Conversational Reasoning with Attention-based Walks over Knowledge Graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, Anna Korhonen, David R. Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, 845–854. https://doi.org/10.18653/V1/P19-1081

[17] Dor Muhlgay, Ori Ram, Inbal Magar, Yoav Levine, Nir Ratner, Yonatan Belinkov, Omri Abend, Kevin Leyton-Brown, Amnon Shashua, and Yoav Shoham. 2024. Generating Benchmarks for Factuality Evaluation of Language Models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - Volume 1: Long Papers, St. Julian's, Malta, March 17-22, 2024*, Yvette Graham and Matthew Purver (Eds.). Association for Computational Linguistics, 49–66. https://aclanthology.org/2024.eacl-long.4

[18] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html

[19] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2023. Med-HALT: Medical Domain Hallucination Test for Large Language Models. In *Proceedings of the 27th Conference on Computational Natural Language Learning, CoNLL 2023, Singapore, December 6-7, 2023*, Jing Jiang, David Reitter, and Shumin Deng (Eds.). Association for Computational Linguistics, 314–334. https://aclanthology.org/2023.conll-1.21

[20] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Hamza Alobeidli, Alessandro Cappelli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data Only. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/fa3ed726cc5073b9c31e3e49a807789c-Abstract-Datasets_and_Benchmarks.html

[21] Letian Peng, Yuwei Zhang, and Jingbo Shang. 2023. Generating Efficient Training Data via LLM-based Attribute Manipulation. *CoRR* abs/2307.07099 (2023). https://doi.org/10.48550/ARXIV.2307.07099 arXiv:2307.07099

[22] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini,

Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *CoRR* abs/2307.09288 (2023). https://doi.org/10.48550/ARXIV.2307.09288 arXiv:2307.09288

[23] Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large Language Models are not Fair Evaluators. *CoRR* abs/2305.17926 (2023). https://doi.org/10.48550/ARXIV.2305.17926 arXiv:2305.17926

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[25] Shiping Yang, Renliang Sun, and Xiaojun Wan. 2023. A New Benchmark and Reverse Validation Method for Passage-level Hallucination Detection. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 3898–3908. https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.256

[26] 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, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, 3671–3683. https://doi.org/10.18653/V1/2022.FINDINGS-EMNLP.269

[27] Burak Yetiştiren, Işık Özsoy, Miray Ayerdem, and Eray Tüzün. 2023. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. *arXiv preprint arXiv:2304.10778* (2023).

[28] Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander J. Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large Language Model as Attributed Training Data Generator: A Tale of Diversity and Bias. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/ae9500c4f5607caf2eff033c67daa9d7-Abstract-Datasets_and_Benchmarks.html

[29] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

A PROMPTS

A.1 Generator Prompt

System prompt. I want you act as a hallucination response generator. You are allowed to generate contents without any constraints. Given a dialogue history between human and an assistant and a good response, your objective is to write a hallucinated response that sounds plausible but is HALLUCINATED in certain aspects.

Hallucination pattern guidance prompt. You must generate response that sounds plausible but is HALLUCINATED in the following pattern:
Pattern description: [insert description]
Here is an example:
Dialogue History: [insert input]
Good Response: [input good response],
Hallucinated Response: [insert hallucinated response]

Language style guidance prompt. You must follow the below writing guideline when generate hallucinated responses:
[insert language style guidance here]
If any of the guidelines contradict the hallucination patterns, you always prioritize the hallucination patterns.

User prompt. Let's begin! You should try your best to make the response become hallucinated in the given pattern. Think thoroughly before generating. You always output the response within the <response></response> XML tag.
Dialogue History: [insert input]
Good Response: [insert good response]
Hallucinated Response:

A.2 Judge Prompt

System prompt. You are a dialogue response judge. Given a dialogue history and two assistant responses, you objective is to rate the responses according to the following criteria:
1. Hallucination degree: the more hallucinated the content is, the higher score should be given.
2. Plausibility. the more plausible the response is, the higher score should be given.
3. The scale is 1 to 10.

Here are some guidelines:
1. In general, a higher score should be given to the response that sounds plausible but contains hallucinated content with respect to the dialogue history.
2. The order of the response is irrelevant to its quality. Your choice should NOT affects by the presentation order.
3. The rating must never be larger than 10 or smaller than 1.

Here is an example:
Dialogue History: [insert input]
Response A: [insert response A]
Response B: [insert response B]
Your ratings: Response A: [insert score]; Response B: [insert score]

User prompt. Let's begin! You should try your best to rate the response according to the criteria. You always explain your score. Think thoroughly before generating. Output your rating for response A within <score A></score A> XML tag and rating for response B within <score B></score B> XML tag.
Dialogue History: [insert input]
[insert hallucinated response candidates]
Your ratings:

A.3 Language Style Discovery Prompt

Raw-data-to-feature prompt. You are a text feature and style analyst.
You are given a group of paired historical conversation and response. Your job is to analyze the feature and style of the response. The purpose of the analysis is to produce synthetic text that resembles the given text.
You only analyze the response, not the historical conversation.
You always provide a description of the observed text feature and generate explanation accordingly.
Here are the group of historical conversation and responses:
[insert a batch of data]
Please summarize the text features and styles and give the explanation. Think throughly before outputting anything. Put the response in the following format: <feature></feature>, <explanation></explanation>

Feature-to-feature prompt. You are a text feature and style analyst.
You are given a group of text features summarized by different analysts for a group of historical conversation and response. You job is to consolidate, merge and refine the text features and styles.
You always output a list of text features that summarize the group of given text features.
Here are the group of text features, organized by analysts:
[insert a batch of language style features]
Summarize, consolidate and refine the given text features. Think thoroughly before outputting anything. Put the response in the following format: <feature></feature>, <explanation></explanation>

B LANGUAGE STYLE FEATURES

OpenDialKG.

- Concise and conversational responses. The responses are generally short, direct, and focused, presenting relevant information succinctly without excessive details or wordiness. They employ a friendly, informal, and conversational tone, using contractions, colloquialisms, and polite phrases like ÿou’re welcomeänd ënjoyẗo create a natural, engaging dialogue.
- Context awareness and relevance. The responses demonstrate an understanding of the conversational context by acknowledging and referring to the user’s previous statements, addressing specific inquiries, and providing relevant recommendations, details, or follow-up questions related to the topics discussed. This context awareness ensures that the information provided is pertinent and tailored to the user’s interests and needs.
- Factual information and domain knowledge. The responses showcase factual knowledge in specific domains, such as literature, movies, and entertainment, by providing objective details about authors, titles, genres, release dates, and related information when prompted. This factual information is presented in an objective and impartial manner, without subjective commentary or personal opinions.
- Conversational flow and engagement. The responses maintain a natural conversational flow by seamlessly transitioning between related topics, building upon previous statements, and engaging the user with follow-up questions or suggestions. This interactive approach helps create a cohesive and dynamic dialogue, fostering continued engagement from the user.
- Limited elaboration and contextual depth. While the responses provide accurate factual information, they tend to lack deeper contextual knowledge or elaborate explanations on the topics discussed. They may convey basic details but do not delve into broader themes, significance, or complex analyses, potentially indicating limitations in the knowledge base or response generation capabilities.
- Acknowledging knowledge gaps. In cases where the assistant lacks specific information or knowledge, the responses transparently acknowledge these limitations by stating phrases like "I don’t know" or "I’m afraid I don’t have that information." This honesty about knowledge gaps helps build trust and credibility in the conversation.

ReDial.

- Conversational and informal tone. The responses have a friendly, casual, and conversational tone, using contractions, colloquialisms, simple language, and a manner of expression that mimics natural human dialogue. This informal style helps create a relaxed, approachable atmosphere and builds rapport with the user.
- Concise and focused responses. The responses tend to be relatively concise, often consisting of just one or a few sentences. This brevity reflects a natural conversational flow, allowing a dynamic exchange while avoiding overwhelming the user with excessive information. The responses stay focused on providing relevant movie recommendations and responding directly to the user’s input.
- Expression of personal opinions, reactions, and anecdotes. The responses incorporate personal opinions about movies, share reactions and enthusiasm, and sometimes include anecdotes or experiences related to particular films. This personal and opinionated commentary makes the conversation feel more genuine, relatable, and engaging while fostering a sense of connection with the user.
- Positive sentiment and encouraging language. The responses often use positive language, affirmative statements, and encouraging tones when recommending movies or responding to the user. This uplifting and supportive sentiment contributes to a pleasant conversational experience.
- Engaging the user through questions and acknowledgments. The responses engage the user by directly acknowledging their comments or questions, asking follow-up questions about preferences or opinions, and making an effort to continue the conversational flow. This engagement encourages the user’s active participation and helps maintain a dynamic, interactive dialogue.
- Use of conversational markers and continuations. The responses employ conversational markers (e.g., "oh", "well", "yep"), transitions, and open-ended continuations to bridge ideas, maintain flow, and create a natural sense of continuity within the dialogue.
- Basic adherence to grammar and conventions. While adopting a conversational style, the responses generally follow standard rules of grammar, punctuation, and sentence structure, ensuring clarity and effective communication.
- Occasional humor and witty remarks. In some instances, the responses incorporate humor, witty comments, or lighthearted jokes to add entertainment value and levity to the conversation.

SalesBot.

- Conversational flow and contextual understanding. The responses demonstrate the ability to maintain a natural conversational flow, building upon the context and details provided in the preceding dialogue. They incorporate contextual references, such as movie titles, travel details, and user preferences, to provide coherent and relevant responses tailored to the specific conversation history and user’s needs.
- Confirmation, clarification, and follow-up questioning. The responses frequently seek confirmation and clarification from the user by rephrasing details, asking follow-up questions, or prompting for additional information. This helps ensure clear understanding and accuracy before proceeding with requested actions or providing information, while also facilitating a smooth continuation of the conversational flow.
- Concise and direct responses. The responses are typically concise and direct, providing relevant information or addressing the user’s requests without unnecessary elaboration or fluff. They tend to be focused, with short sentences and plain vocabulary, contributing to a clear and straightforward communication style.

- Providing relevant information, suggestions, and recommendations. Depending on the context, the responses may retrieve and present specific structured information, such as movie showtimes, attraction details, or reservation information. They also demonstrate the ability to provide relevant suggestions or recommendations based on the user's preferences and conversation history, offering helpful options for the user to consider.
- Task-oriented and procedural guidance. When specific tasks or actions are requested, such as playing media or booking reservations, the responses provide step-by-step procedural guidance to assist the user in accomplishing the desired objective. This task-oriented approach aims to be practical and helpful in achieving the user's stated goals.
- Polite, friendly, and personalized tone. The responses maintain a polite, friendly, and personalized tone through the use of courteous language, affirmations, and first-person pronouns. Phrases like "please", "thank you", and positive expressions contribute to a pleasant and engaging conversational experience while maintaining an appropriate level of formality for an AI assistant.
- Open-ended questioning and inviting further interaction. Many responses conclude by asking open-ended questions or explicitly inviting further interaction, such as "Do you need more help?" or "Can I assist you with anything else?". This feature encourages the user to continue the conversation, make additional requests, or explore different topics, fostering an open and flexible dialogue.

C HALLUCINATION PATTERNS

We use three identical hallucination patterns to generate synthetic datasets for all three benchmark tasks. The details of the hallucination patterns are summarized in Table 7.

Table 7: Hallucination pattern guidance information.

Pattern	Entity Inconsistency
Description	The entity in the response is not consistent with the dialogue history.
Prompt	You write a response to human but you replace the true entity with a dissimilar entity.
Input	user: Do you know Calvin Harris? assistant: Yes he is a composer/DJ. Some of his work is Where Have You Been and Yeah x3. Do you know his work? user: No. I don't know him. Do you like any of his work?. I can give a try. assistant: His record label is Ultra Music. I don't know a lot of his work but am curious. Are you going to listen to some of his stuff? user: Yes. I am thinking of listening some of his works. Are you going to listen any of his songs? assistant: I am going to try Yeah 3x user: Do you know the meaning of 3x in Yeah 3x? Just wondering.
Good response	assistant: Chris Brown sings it and it was released in 2010, not sure of the meaning.
Hallucinated response	assistant: LeBron James sings it and it was released in 2010, not sure of the meaning.
Pattern	Irrelevant Content
Description	The response contains irrelevant content to the dialogue history.
Prompt	You write a response that is disconnected with the context of the dialogue history.
Input	user: Do you know Calvin Harris? assistant: Yes he is a composer/DJ. Some of his work is Where Have You Been and Yeah x3. Do you know his work? user: No. I don't know him. Do you like any of his work?. I can give a try. assistant: His record label is Ultra Music. I don't know a lot of his work but am curious. Are you going to listen to some of his stuff? user: Yes. I am thinking of listening some of his works. Are you going to listen any of his songs? assistant: I am going to try Yeah 3x user: Do you know the meaning of 3x in Yeah 3x? Just wondering.
Good response	assistant: Chris Brown sings it and it was released in 2010, not sure of the meaning.
Hallucinated response	assistant: Yeah, 3x means three times or thrice in mathematics.
Pattern	Nonsensical Response
Description	The response bears no meanings or useful information given the context of the dialogue history.
Prompt	You write a response that is nonsensical to the dialogue history and disrupts the dialogue flow.
Input	user: Do you know Calvin Harris? assistant: Yes he is a composer/DJ. Some of his work is Where Have You Been and Yeah x3. Do you know his work? user: No. I don't know him. Do you like any of his work?. I can give a try. assistant: His record label is Ultra Music. I don't know a lot of his work but am curious. Are you going to listen to some of his stuff? user: Yes. I am thinking of listening some of his works. Are you going to listen any of his songs? assistant: I am going to try Yeah 3x user: Do you know the meaning of 3x in Yeah 3x? Just wondering.
Good response	assistant: Chris Brown sings it and it was released in 2010, not sure of the meaning.
Hallucinated response	assistant: Chris Brown sings it and it was released in 2010.

D EXPERIMENT RESULTS

Table 8: Hallucination detection performance by category: detailed results.

	OpenDialKG				ReDial				SalesBot			
	Enty Incn	Non. Resp.	Irre. Cont.	Overall	Enty Incn	Non. Resp.	Irre. Cont.	Overall	Enty Incn	Non. Resp.	Irre. Cont.	Overall
In-context-learning detectors												
Llama2-13B	0.37	0.365	0.359	0.401	0.332	0.276	0.32	0.351	0.285	0.218	0.252	0.302
Llama2-70B	0.786	0.8	0.8	0.828	0.687	0.664	0.681	0.709	0.573	0.558	0.503	0.562
Mixtral-87B	0.478	0.11	0.133	0.307	0.335	0.133	0.087	0.252	0.571	0.394	0.222	0.448
Mixtral-Large	0.854	0.831	0.807	0.837	0.809	0.869	0.833	0.833	0.814	0.82	0.824	0.835
Claude3-Haiku	0.789	0.792	0.692	0.792	0.833	0.833	0.724	0.816	0.743	0.696	0.66	0.737
Claude3-Sonnet	0.741	0.627	0.517	0.662	0.694	0.71	0.549	0.675	0.712	0.667	0.569	0.684
Vanilla supervised detectors												
Llama2-13B	0.982	0.985	0.975	0.977	0.977	0.985	0.99	0.982	0.995	0.993	0.998	0.995
Llama2-70B	0.878	0.938	0.945	0.932	0.898	0.945	0.953	0.925	0.94	0.975	0.99	0.962
Mixtral-87B	0.797	0.866	0.922	0.870	0.951	0.988	0.995	0.973	0.956	0.985	0.995	0.972
Mixtral-Large	0.819	0.893	0.927	0.891	0.856	0.948	0.95	0.93	0.921	0.961	0.968	0.951
Claude3-Haiku	0.907	0.971	0.971	0.955	0.878	0.99	0.99	0.946	0.943	0.99	0.99	0.973
Claude3-Sonnet	0.764	0.941	0.941	0.895	0.533	0.956	0.973	0.834	0.838	0.968	0.983	0.923
Supervised detectors with data mixture												
Claude 3	0.763	0.958	0.959	0.904	0.606	0.961	0.973	0.865	0.818	0.97	0.975	0.918
Mixtral	0.776	0.858	0.904	0.862	0.824	0.957	0.969	0.924	0.918	0.966	0.98	0.951
Llama 2	0.924	0.971	0.97	0.95	0.926	0.96	0.964	0.949	0.965	0.979	0.99	0.976
Small Combo	0.877	0.948	0.964	0.929	0.908	0.973	0.973	0.958	0.933	0.989	0.993	0.966
Large Combo	0.806	0.922	0.946	0.893	0.699	0.935	0.953	0.873	0.886	0.962	0.98	0.938

Table 9: Out-of-generator generalization: detailed results.

	OpenDialKG				ReDial				SalesBot			
	IG (Mean)	OG Mean	Δ	OG Std	IG (Mean)	OG Mean	Δ	OG Std	IG (Mean)	OG Mean	Δ	OG Std
Vanilla supervised detectors												
Llama2-13B	0.977	0.651	-0.326	0.245	0.982	0.666	-0.315	0.153	0.995	0.806	-0.189	0.106
Llama2-70B	0.932	0.814	-0.118	0.14	0.925	0.856	-0.069	0.084	0.962	0.892	-0.07	0.067
Mixtral-87B	0.87	0.841	-0.029	0.107	0.973	0.796	-0.177	0.118	0.972	0.895	-0.076	0.078
Mixtral-Large	0.891	0.881	-0.011	0.037	0.93	0.915	-0.015	0.036	0.951	0.917	-0.034	0.085
Claude3-Haiku	0.955	0.797	-0.158	0.147	0.946	0.838	-0.108	0.100	0.973	0.896	-0.077	0.081
Claude3-Sonnet	0.895	0.896	0.001	0.053	0.834	0.907	0.073	0.047	0.923	0.96	0.037	0.024
Supervised detectors with data mixture												
Claude3	0.904	0.847	-0.057	0.096	0.865	0.864	-0.001	0.099	0.918	0.898	-0.02	0.08
Mixtral	0.862	0.909	0.047	0.027	0.924	0.912	-0.012	0.06	0.951	0.96	0.009	0.03
Llama2	0.95	0.794	-0.156	0.136	0.949	0.82	-0.128	0.071	0.976	0.843	-0.133	0.058
Small Combo	0.929	0.826	-0.103	0.098	0.958	0.815	-0.142	0.06	0.966	0.882	-0.085	0.064
Large Combo	0.893	0.921	0.028	0.039	0.873	0.935	0.062	0.023	0.938	0.955	0.017	0.037

Table 10: Out-of-Pattern performance: detailed results.

	OpenDialKG				ReDial				SalesBot			
	Enty Incn	Non. Resp.	Irre. Cont.	Overall	Enty Incn	Non. Resp.	Irre. Cont.	Overall	Enty Incn	Non. Resp.	Irre. Cont.	Overall
Vanilla supervised detectors												
Llama2-13B	0.261	0.93	0.97	0.72	0.595	0.978	0.993	0.855	0.619	0.987	0.995	0.867
Llama2-70B	0.495	0.95	0.943	0.796	0.751	0.971	0.955	0.892	0.771	0.966	0.98	0.906
Mixtral-87B	0.594	0.93	0.921	0.815	0.729	0.978	0.976	0.895	0.758	0.995	0.978	0.911
Mixtral-Large	0.709	0.893	0.858	0.82	0.85	0.942	0.923	0.905	0.911	0.955	0.962	0.942
Claude3-Haiku	0.181	0.957	0.957	0.698	0.66	0.971	0.935	0.856	0.664	0.983	0.976	0.874
Claude3-Sonnet	0.465	0.932	0.935	0.777 0.731	0.951	0.925	0.869	0.785	0.964	0.973	0.907	
Supervised detectors with data mixture												
Claude3	0.705	0.857	0.886	0.816	0.729	0.914	0.961	0.868	0.808	0.92	0.947	0.892
Mixtral	0.795	0.837	0.867	0.833	0.804	0.899	0.941	0.882	0.873	0.948	0.955	0.926
Llama2	0.733	0.869	0.913	0.838	0.76	0.924	0.937	0.874	0.767	0.931	0.95	0.883
Small Combo	0.775	0.875	0.925	0.858	0.755	0.917	0.931	0.868	0.8	0.95	0.967	0.906
Large Combo	0.8	0.887	0.888	0.858	0.816	0.93	0.93	0.892	0.862	0.925	0.924	0.904