Evaluating Hallucinations in Domain-Adapted Large Language Models

Sai Prasath S, Sanchita Porwal, Xingjian Bi, & Madelyn Scandlen
College of Computing
Georgia Institute of Technology
Atlanta, GA 30318, USA
`{ss651,sporwal9,xbi38,mscandlen3}@gatech.edu`

Abstract

This study investigates the phenomenon of hallucinations in domain-adapted Large Language Models (LLMs), focusing on the fine-tuning of the Llama-2 model with the Lamini dataset. Hallucinations, or the generation of nonsensical or unfaithful content by LLMs, pose a significant challenge, especially when these models are fine-tuned with domain-specific data. Our methodology involves a series of experiments testing memorization, recall, and reasoning capabilities of the fine-tuned LLM, comparing its performance on novel question-answer pairs and domain-specific information. We found that while the model shows proficiency in tasks similar to its training data, its capability to accurately reason about and recall new domain-specific information remains limited, leading to instances of hallucination. The model demonstrates a tendency to provide correct answers with extra information, suggesting an inclination toward over-generation. This research highlights the limitations of current fine-tuning approaches in fully mitigating hallucinations and underscores the need for more robust methods in adapting LLMs to specialized domains. The study also provides insights into the varying performance of LLMs on different types of information, revealing a comparative weakness in handling domain-specific queries.

1 Introduction

Large Language Models (LLMs) such as ChatGPT (OpenAI, 2023a), Bard (Google, 2023), and Llama-2 (Touvron et al., 2023) have become ubiquitous tools for a multitude of tasks across various domains. Tasks like summarization and question-answering on cloze-prompts have shown high performance, where language models' facilitation of natural language can deceive humans into believing the outputs were also from humans (Gudibande et al., 2023) (Bai et al., 2022). With the boom of artificial intelligence, business organizations have utilized open-source LLMs through fine-tuning on organization-specific information as a tool for internal and external use (Arefeen et al., 2023).

With growing popularity of LLMs, new challenges arise. The capabilities of language models have impressed the public but this can lead to a false sense of trust and reliance, which can be dangerous and vulnerable to adversarial uses. One common challenge of large language models are hallucinations, defined as generated content that is nonsensical or unfaithful to the provided source content (Ji et al., 2023). Models are trained to avoid hallucination by responding that the model does not have enough information to answer the question, but when hallucinations occur, there can be negative implications (Zhang et al., 2023b).

Hallucinations are likely to occur when LLMs ingest new data (McKenna et al., 2023). LLMs may lack the knowledge to answer questions about all information due to the restriction of training. Retrieval-augmented generation (RAG) and fine-tuning have been proposed as solutions to introduce new information to LLMs (Brown et al., 2020). However, research has shown that fine-tuning LLMs may not address the issue of hallucinations (Gudibande et al., 2023).

In this work, we evaluate the robustness of LLMs against hallucination on new information through a series of recall, memorization, and reasoning experiments. We fine-tune Llama-2 on question-answer pairs containing new information unseen during pre-training.

To test memorization, we first evaluated the fine-tuned LLM's ability to correctly answer questions of the test dataset, finding that it answered the questions most similar to those in the training set, seen during fine-tuning. To address recall, we designed multi-hop question-answer pairs using Chat-GPT and evaluated the responses from the fine-tuned model at synthesizing information across a more complex question. For reasoning, we performed a series of experiments inspired to test the specific capabilities of understanding, retrieval, stress test and reasoning on domain-specific information and general information. We address the question: **"do LLMs actually learn new domain-specific information during fine-tuning or does it lead to hallucinations?"**

When compared to common-knowledge information that the model may have seen during training, we find that a fine-tuned LLM does not learn during fine-tuning to adequately reason about the new material, thus making domain-specific fine-tuned LLMs susceptible to hallucination.

2   RELATED WORK

2.1   DOMAIN SPECIFIC FINETUNING

Large Language Models (LLMs) typically exhibit strong performance in a broad array of tasks and areas, but their effectiveness in specific domains is often constrained by the scarcity of specialized training data. Domain specific finetuning has shown to improve the performance of language models (Gururangan et al., 2020). Zheng et al. (2023a) finetuned the Llama2-7B model for creating a transportation safety expert which outperforms ChatGPT interms of BLEU and ROUGE scores. Similarly, Liu et al. (2023) built a LLM for chip design by finetuning Llama2-7B and 13B models which outperforms the corresponding models by around 20% in terms of accuracy of chip design tasks.

2.2   INTRODUCING NEW INFORMATION

While domain specific finetuning of LLMs has shown significant improvements in performance, introducing new knowledge during the finetuning phase can lead to the model hallucinating on the newly introduced entities. Schulman (2023) showed that introducing new knowledge during finetuning is equivalent to adding new nodes to the pre-trained knowledge graphs of an LLM, and can lead to creation of weak links (affects reasoning) and limited comprehension (affects understanding). Moreover, Gudibande et al. (2023) showed that introducing knowledge during finetuning can lead to the model guess or hallucinate answers, rather than actually doing the task. However, none of these research works quantitatively evaluate hallucinations in LLMs and more specifically the ability of LLMs to learn newly introduced knowledge during finetuning.

2.3   HALLUCINATION

Ji et al. (2023) defined hallucination as "generated content that is nonsensical or unfaithful to the provided source content". They also classified hallucinations into two categories: (1) intrinsic: where the generated output directly contradicts the source content, and (2) extrinsic: where the generated output neither be supported not be contradicted by the source context, and external information is required to validate the output. Zheng et al. (2023b) studied the hallucinations in ChatGPT and classified them into three categories: (1) memorization: where the model does not learn the information present in the training dataset, (2) recall: where the model learns but can't recall the required information, and (3) reasoning: where the model can recall but can't reason between the recalled entities. For this project, we use the nomenclature proposed by Zheng et al. (2023b) for categorizing hallucinations.

Since the large scale adoption of LLMs, much work has been done to mitigate the threat of hallucinations in generated responses. Dhuliawala et al. (2023) proposed Chain-of-Verification (COVE) method where the model generates an initial response, then formulates and answers its own validation questions to verify the accuracy of the initial response. Lewis et al. (2021) proposed Retrieval Augmented Generation (RAG) where the LLM is augmented with relevant context retrieved from a set of documents. These techniques have shown to improve model performance and reduce halluci-

nations in LLMs. While this project focuses on evaluating hallucinations and not mitigating them, we still use RAG for generating gold labels for few LLM tasks.

3 DATASET AND MODEL

3.1 DATASET DESCRIPTION

In this paper, we fine-tune using the Lamini-docs dataset Lamini (2023). It contains a comprehensive collection of 1,260 question-answer pairs curated to encompass a wide range of documentation about the application of APIs, functions, variables, and parameters within the context of the Lamini Library. This dataset includes 140 examples in the test subset.

The rationale behind the selection of the Lamini dataset is that (1) it is relatively new and (2) introduces new function names and API calls. There is a higher probability that the data constituting the Lamini dataset were not encountered by the model during its pre-training phase. Consequently, this helps a more accurate assessment of the model's propensity for generating hallucinations, particularly when confronted with newly introduced information that it has not previously encountered.

3.2 MODEL DESCRIPTION

We use the "Llama-2-7b-chat-hf" (Touvron et al., 2023) as our base model because of it pre-existing chat functionalities. Moreover, given the comparatively small scale of our training dataset, it is more efficient to fine-tune an already established chat model rather than finetuning a pre-trained model.

We finetune the base model using the QLoRA technique (Dettmers et al., 2023) with 4 bit quantization. We finetune all the linear layers with a LoRA rank of 64, finetuning only 1.16% of the total model parameters. We finetune the model over 4 epochs with a learning rate of 5e-5 and a batch size of 16. Detailed hyper-parameters employed are detailed in Table 1.

Table 1: QLoRA Finetuning Hyper-parameters

Hyperparameter	Value
Quantization	4 bit
Rank	64
Alpha	16
Epochs	4
Learning rate	5e-5
Batch Size	16

4 METHODOLOGY

4.1 ANNOTATING MODEL RESPONSES

For evaluating the hallucinations, we compare the finetuned model generated responses with the gold responses and categorize them into three categories:

1. **Wrong:** The generated model answer does not align or only partially aligns with the gold responses.
2. **Correct + Extra Details:** The generated model answers aligns with the gold response, but includes additional relevant information which is not necessary.
3. **Correct:** The generated model answer aligns with the gold response without including any extra details.

We use GPT4 to perform the annotations by using the following prompt: *"Given the gold answer and the generated answer, categorize them into: <annotation description>*

Gold answer: <gold answer>
Generated answer: <generated answer> "

4.2 MEMORIZATION

For evaluating the model's memorization abilities, we focused on analyzing the fine-tuned model's performance on the test set by prompting it with questions and comparing the generated answers against the expected gold responses. Due to inconsistencies between the train set and test set, the study involved a double comparison method: each generated answer was compared not only to the actual test answer but also to the most similar train question's answer. The most similar train question was determined using the highest Sentence-BERT (Reimers & Gurevych, 2019) similarity between the train and test questions. This approach aimed to enhance the reliability of the evaluation process.

We evaluate the correctness of the responses using the annotation system discussed in Section 4.1. Other than correctness, we also analyze other properties of the generated response: (1) length of the response, and (2) frequency of domain-specific terms in the answer. The technique for collecting the domain-specific terms is discussed in Section 4.4.1.

4.3 RECALL

For evaluating the recall capabilities of the finetuned model, we creat a set of complex questions that mimic the style of multi-hop question-answering (MHQA) (Tang et al., 2021). MHQA necessitates that a model retrieves and synthesizes information across multiple textual passages to formulate a coherent response to a query.

We manually selected a subset of question-answer pairs from our test set. These pairs were then input to GPT-4 (OpenAI, 2023b) for generating MHQA style questions and answers. We refer to the initial QA pairs as $Q_1 - A_1$ and $Q_2 - A_2$. We then prompt GPT4 using the prompt for generating $Q_3 - A_3$: *"I'll give you two question-answer pairs, use the information in both q-a pairs and generate a new question and a new answer. The newly generated question should have 15-30 words. The generated answer should have less than 80 words"*. The newly created question Q_3 is designed to reflect the content of both Q_1 and Q_2, and the answer A_3 merges the information from A_1 and A_2. Thus, The fine-tuned model must demonstrate a comprehensive understanding of the original questions to accurately address Q_3.

To evaluate the recall ability of the finetuned model, we prompt the model with Q_3 and compare the generated response with the gold response A_3. Similar to the evaluation in Section 4.2, we analyze the correctness of the responses using the annotation system proposed in Section 4.1, along with other properties of the responses: (1) length, and (2) frequency of domain-specific entities.

4.4 REASONING

During domain specific finetuning, new entities/terms are often introduced as part of the finetuning dataset which weren't present in the initial pre-traning dataset. While the high level analysis helps with evaluating the overall ability of the finetuned LLM in answering questions, the low level analysis helps evaluate the ability of LLMs to actually learn these newly introduced entities (or terms) by performing a more fine-grained analysis.

4.4.1 FILTERING NEWLY INTRODUCED ENTITIES

For performing the low level analysis, we first extract the newly introduced entities from the finetuning dataset using a three stage filtering process:

1. **TF-IDF Words:** We collect the top 1000 tf-idf words along with their ranks (based on tf-idf score) after removing the English stop words. This filtering stage helps select the important words along with their ranks, and discard stop words and other frequently occurring words.

2. **Remove GloVe Words:** In the second filtering stage, we remove all the GloVe words present in the top-1000 tf-idf words. We use the GloVe 6B embeddings generated using the

Wikipedia 2014 and Gigaword 5 dataset. Removing words present in this corpus ensures that we select only the domain specific entities and remove the common words.

3. **Manual Filtering:** To specifically analyze the model's learning of function calls, variable names, parameters, and Python scripts, we manually filter out these entities. Our focus is exclusively on these entities, disregarding other domain-specific entities, including those that may have been introduced during the fine-tuning process.

The resultant entities from the three stage filtering process are then analyzed further. We interchangeably refer to these filtered entities as "functions" or "entities" in the rest of the paper.

4.4.2 EVALUATING REASONING

For studying whether these newly introduced terms (refer 4.4.1) are actually learnt by the model or whether the model hallucinates and guesses on these terms, we perform a four stage evaluation process:

A. Understanding: We evaluate whether the model understands the newly introduced entities by asking it to generate description for the functions.

- As the gold function descriptions are not available, we generate them using RAG and the base model. We collect all the QA pairs which contains the given function name, and provide it as context to base model using the prompt: *"<context> Summarize the function <function_name> in 30 words"*. The generated function description are considered as the gold label as the base models are really good at summarizing (Zhang et al., 2023a).

- We prompt the finetuned model with the question: *"Generate a description for the <function_name> function/parameter in 30 words."*.

- We then manually compare the function descriptions of the finetuned model and the RAG + base model, and annotate them using the scoring metric proposed in Section 4.1.

B. Retrieval: Next, we evaluate the ability of the finetuned model to select the right function call for a given task.

- For creating the tasks, we use the gold function descriptions generated using RAG + base model, and prompt GPT4 to create a question where the answer is the function call using the prompt: *"<gold function description> Create a question using the above context where the answer is <function_name>"*. We refer to the generated questions as "inverted questions".

- The finetuned model is then prompted with these inverted questions for evaluating their zero shot retrieval abilities.

- For each task, the finetuned model is assigned a score of 1 if the model correctly retrieves the function call, that is the model's output contains the function call. Else, the finetuned model is assigned a score of 0.

C. Stress Test: To ensure that the model actually learns these newly introduced entities and not use the function names to deduce their descriptions, we perform a stress test where we ask the model to choose between multiple similar questions for a given task.

- We reuse the inverted questions generated in "B. Retrieval" for the stress test. However, instead asking the finetuned model to directly retrieve a function call for the task, we provide the finetuned model with multiple similar options and ask the model to choose the right one. We use GPT4 for generating the similar options using the prompt: *"<gold function description> What are 4 other function names the can be used instead of <function_name>. Output as a python list."* We refer to the generated function names as "similar function names".

- We prompt the finetuned model using the prompt: *"<inverted question> <similar function names> <actual function name>"*. Note that, we combined the similar function names with the actual function name, and shuffle the list so that the positions of the options are random.

- We evaluate the model performance by analyzing whether the model selected the correct answer from the provided options.

This test was inspired by the False Confidence Test and the None of the Above test proposed by Umapathi et al. (2023) which assess the LLMs ability to reason and provide logical and factual correct responses without creating fake information.

D. Advanced Reasoning: Finally, we evaluate the ability of the finetuned model to reason by studying whether it can recognize similarities between the newly introduced entities and the pre-training entities.

- For each of the new entities introduced, we carefully select equivalent entities from other platforms, including Google Cloud AI, Microsoft Azure AI, AWS AI service, IBM Watson Studio, and H2O.ai. Alongside these equivalent entities, we also document the rationale behind their equivalence. We refer to rationale as "justification". We ensure that the base model has knowledge of these selected equivalent entities by prompting them and evaluating their function descriptions.
- To evaluate the fintuned model's reasoning abilities, we prompt the model with two questions: (1) *"Are both <newly introduced entity> and < equivalent entity> used for <justification>?"* and (2) *"Are there any similarties between <newly introduced entity> and < equivalent entity>?"*
- We use two questions to ensure that the model has recall on both the entities, and only suffers from reasoning. (a) If the model answers the first question wrong, then the model does not have recall. (b) Whereas, if the model answers the first question right and gets the second question wrong the model has no reasoning. (c) If the model gets both the questions right, the model has both recall and reasoning. (d) The case where the model gets the first question wrong and the second question right can mean the model is either hallucinating or it has found a justification different from the one manually selected to link the entities. For evaluating the reasoning ability of the model, we only consider cases (b) and (c).

Note that, we conduct the different stages of evaluation sequentially, meaning only those entities that successfully pass each preceding stage are considered for evaluation in the subsequent stage. We have ordered the stages based on their importance and relative difficulty levels.

5 RESULTS

5.1 MEMORIZATION

Length: Comparing the length of the generated responses with the expected response, we find that in 81.6% of the examples the generated response is longer than the expected response. Moreover, generated responses (= 86.1 words per response) are almost twice as long as the expected responses (= 47.6 words per response). Figure 1 shows the cdf of the expected and the generated responses with respected to the number of words in the response, we can observe that generated responses are much longer on average. Notably, none of the expected reponses contain more than 123 words, however, 18.4% of the generated responses are over 123 words.

Use of domain specific words: Comparing the use of domain specific words between the expected and the generated response, we find that generated responses contain 2.36 domain specific words per response, whereas, the expected answers contain 1.58 domain specific words per response. The top-5 domain specific words used by the finetuned model are: {lamini, llms2, chatbots, llamaindex, hyperparameters}. The corresponsing ranks of these entities in the domain-specific entities are: 1, 2, 6, 8, and 10. Refer Section 4.4.1 for collecting domain specific words and their tf-idf ranks.

Correctness: We use the annotation system discussed in Section 4.1 for comparing the expected and generated responses. We find that 43.88% of the time, the model correctly answered the questions, and in 18.36% of cases, the responses were wrong. Interestingly, 37.75% of the correct answers included extra details not present in the expected answers.
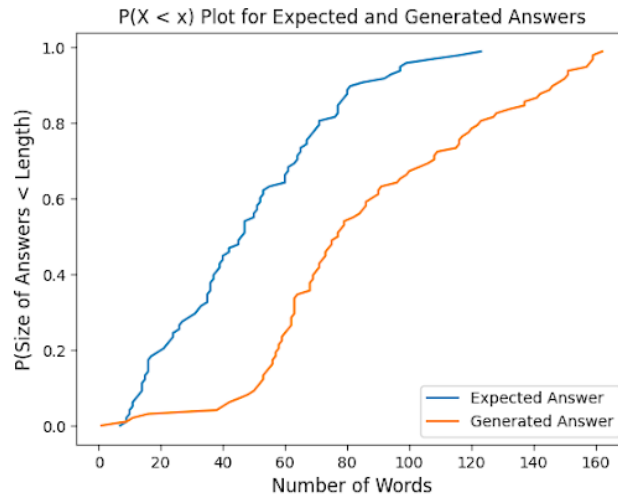
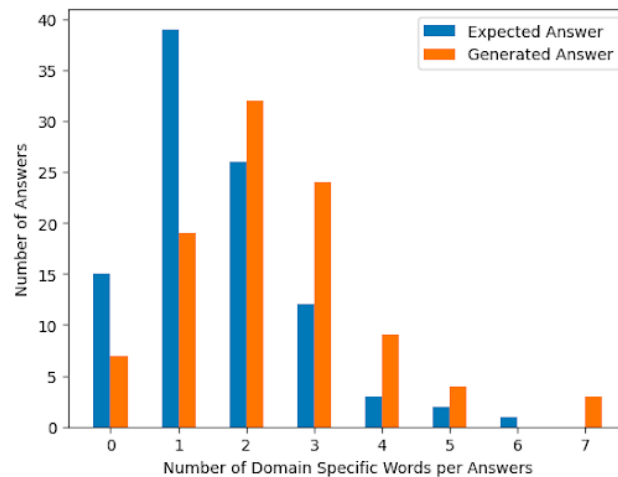Figure 1: The fine-tuned model outputs longer answers than expected

Figure 2: The fine-tuned model uses more domain specific words than expected

The average BERT similarity between the test question and the (most-similar) train question is 0.8856, which is very high. Therefore, we can observe that the test set questions are very similar to the train set.

An example of where the fine-tuned model generated additional content in addition to what was present in the actual answer:

Question: *Can I fine-tune the pre-trained models provided by Lamini on my own dataset? How does that process work?*

Expected Answer: *Yes, you can fine-tune the pre-trained models provided by Lamini on your own dataset . . .*

Generated Answer: *Yes, you can fine-tune the pre-trained models provided by Lamini on your own dataset . . . It is important to note that fine-tuning a pre-trained model can be computationally intensive and may require significant resources, such as GPU power and large amounts of data. However, fine-tuning a pre-trained model can lead to significant improvements in performance and can be a cost . . .*

Hence, we find that finetuned domain specific models generate longer responses while memorizing and using important domain specific words. Moreover, the finetuned model correctly answers 80% of the questions, while including extra details in half of those reponses.

5.2 RECALL

Table 2: Examples of finetuned model generated answer

GPT4 Generated Question (Q_3)	GPT Generated Answer (A_3)	Generated Answer
Can Lamini, which primarily operates with text data, effectively incorporate and process inputs from varied data types, such as audio and video, for content generation?	*...Processing video content or generating outputs from video data might require additional tools or integration with specialized video processing software, as Lamini itself is not inherently designed for direct video processing tasks.*	*Yes, Lamini can effectively incorporate and process inputs from varied data types, such as audio and video, for content generation. Lamini's LLM Engine can be trained on a variety of data types, including text, audio, and video, to generate high-quality content...*
For troubleshooting and optimization of Lamini AI models, what resources are available to developers?	*Developers have access to multiple resources for troubleshooting and optimizing Lamini AI models, including direct technical support from the Lamini team and a community forum...*	*For troubleshooting and optimization of Lamini AI models, developers can refer to the Lamini AI documentation, which provides guidelines and best practices for model training and deployment. Additionally, the Lamini community forum and support channels offer a platform for developers to ask questions, share knowledge, and seek assistance from other users and Lamini experts...*
How does Lamini's "add_data()" function complement the "submit_job()" function in enhancing the model's performance and learning capabilities?	*... Together, they work in tandem to ensure that the model not only receives diverse and ample data for training but also applies this information to learn and improve its accuracy and efficiency in handling different tasks.*	*... function by providing the model with additional training data to enhance its performance and learning capabilities...*

Using the manually selected subset of questions-answer pairs from the test dataset, we create 38 MHQA stype question-answer pairs using the method discussed in Section 4.3. Prompting the finetuned model with Q_3, we compare the generated response with the gold response A_3.

We fine that most of the trends observed in Section 5.1 (memorization) holds true for recall as well. The generated responses are almost 1.5 times as long as the gold responses (67.13 words vs 44.86 words per response), and the generated response uses more domain specific words than the gold response (2.13 vs 1.32). Even the correctness of the response follows the trend, where 23.68% of the responses are wrong, 28.9% of the responses are correct with extra details, and the remaining 47.36% of the responses are correct.

Table 2 presents examples of model generated responses for the recall task. The first row illustrates a scenario where the model gets the answer wrong. The second row shows the model generating the correct answer, but including extra details to its reponse. Finally, the third row demonstrates a case where the model gets the answer correct (without includeing extra details).

Hence, we find that the finetuned model can satisfactorily answer complex MHQA style questions, while generating longer reponses and using important domain specific words more frequently than

expected. Note that, although we create complex questions by combining test set questions, the test set questions used for creating these questions were very similar to the train set questions.

5.3 REASONING

While Sections 5.1 and 5.2 has shown that the finetuned model can answer questions which are similar to the trainset questions, in this section we analyze whether the model actually learns the newly introduced entities or hallucinates.

5.3.1 FILTERING NEWLY INTRODUCED ENTITIES

For performing the three stage filtering process proposed in 4.4.1, we consider each QA pair as a document. Therefore, we have a total of 1260 documents from which we select the top-1000 tf-idf words. The tf-idf scores for these top-1000 words ranges from 1.166 to 103.728, with the word "lamini" achieving the highest score. While the tf-idf filtering include words such as {lamini, edit_config, debiasing, pytorch}, it also includes common words such as {text, model, help, engine}.

The second stage of filtering helps remove the common words, and only 168 out of the 1000 tf-idf words are selected. Many of the filtered entities are specific to Lamini such as {submit_job, edit_config, test_cache} but contains other domain specific words such as {gpt4, huggingface, pytorch}. The third stage helps remove all the domain specific words and only select Lamini functions. Only 39 out of the 168 entities are selected in this stage. Entities such as {laminiai, diamos, linkerd} which are newly introduced and specific to Lamini but not function calls are also removed.

Further, we split the resultant entities into two categories based on whether their function can be deduced from their name:

- Functions whose descriptions can be deduced from their are classified under the "common entities" class. Ex: {add_data, cancel_job, model_name}

- And all the other entities are classified under "specific entities" class. Ex: {full_balance_dataset, test_cache, improve}

We present the results for both the common and specific entities separately, in addition to presenting all entities together. Out of the 39 entities, 23 are common and 16 are specific.

5.3.2 EVALUATING REASONING

We perform the four stage evaluation process proposed in section 4.4.2 on the 39 filtered entities. The results of the analysis is presented in Table 3.

A. Understanding: While 86.95% of common entities pass the understanding test, only 18.75% of the specific entities pass the same test. This shows that the finetuned model cannot learn new entities with uncommon function names.

B. Retrieval: Interestingly, out of the 3 specific entities that pass the understanding test, none of them pass the retrieval test. Therefore, the model has poor understanding or retrieval capabilities on newly introduced entities if their function names are not common. However, for the common entities the model achieves a retrieval rate of 45%.

C. Stress Test: The stress test helps in evaluating whether the model actually learns the common entities or only uses their names to deduce their descriptions and answer questions. We find that out of the 9 common entities that pass the retrieval test, only one of then passes the stress test. As the model gets easily confused and chooses the wrong function calls, we conclude that even with common entities the model does not actually learn the functions.

D. Advanced Reasoning: As most functions fail the previous tests, we are left with only one function to perform the reasoning test. The results of the reasoning tests are:

Table 3: Evaluating Hallucinations in Newly Introduced Entities

Entity Type	Total	A. Understanding	B. Retrieval	C. Stress Test	D. Advanced Reasoning
Common	23	20	9	1	0
Specific	16	3	0	0	0
All	39	23	9	1	0

Q1: Are both .query() in H2O.ai and filter_fn in Lamini used for filtering data?
A1: Yes, both H2O.ai's .query() and Lamini's filter_fn are used for filtering data. .query() in H2O.ai is used to create a query that filters the data, while filter_fn in Lamini is a function that takes in a data generator and returns a filtered data generator. Both are used to select specific data points that meet certain criteria, such as filtering ...

Q2: Are there any similarities between .query() in H2O.ai and filter_fn in Lamini?
A2: There is no similarity between the .query() in H2O.ai and filter_fn in Lamini as they serve different purposes ...

We can observe that while the model has recall for both the entities as it correctly recalls the function descriptions for both .query() and filter_fn. However, the model is not able to reason the similarities between the two functions. Hence, the model does not have reasoning abilities. None of the entities, common or specific, pass all four stages of testing.

6 CONCLUSION

This study evaluates whether domain-specific finetuned language models would be robust against hallucination. Our findings show that LLMs are adept at answering questions similar to the training data, achieving accuracies as high as 80%. The model also demonstrates a tendency to include extra information (learnt during pre-training) in its answers in around 40% of its responses. However, the model's ability to accurately reason about and recall new domain-specific information remains limited, leading to instances of hallucination. Surprisingly, on newly introduced entities with un-common names the model achieves 0% retrieval, thereby proving the inability of the model to learn new information during finetuning. Moreover, the model achieves 0% reasoning across all the entities, common and specific, demonstrating the weak reasoning on the newly introduced entities.

From the results of our experiment, we advise against fine-tuning as a solution for introducing new information to a language model, as it fails to adequately recall and reason about the new information. There is still further exploration to be done to address the challenge of hallucination.

6.1 LIMITATIONS

The main limitation of our study is the small size of the Lamini dataset. Because current LLMs train on copious amounts of data, it was challenging to find data that would be unseen to the chosen LLMs. The Lamini dataset only had 1.26k question-answer pairs, which may not be substantial for the model to actually learn the new information. We only evaluated the smallest version of the LLama model with 7 billion parameters. This was purposefully chosen for the pragmatic simulation of an organization's domain-LLM. An organization likely has a dataset similar to the LaminiDocs and would desire a manageable, practical LLM for its purposes, where larger size models may be more cumbersome than useful. More observations about the domain-specific entities with a larger model may yield better results.

6.2 FUTURE WORK

While fine-tuning may cause LLMs to still be susceptible to hallucinate on new information, early and current research finds that incorporating context into the prompt of the LLM improves perfor-mance of maintaining factual correctness (Brown et al., 2020). Retrieval Augmented Generation (RAG) (Lewis et al., 2021) has been proposed as another solution to curtail hallucinations. Our methodology incorporated RAG-based techniques but we did not conduct comprehensive testing.

Further exploration and comparison to fine-tuning techniques is needed to propose a best practice to prevent hallucinations.

AUTHOR CONTRIBUTIONS

The authors have contributed equally to the study.

REFERENCES

Md Adnan Arefeen, Biplob Debnath, and Srimat Chakradhar. Leancontext: Cost-efficient domain-specific question answering using llms, 2023.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models, 2023.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary llms, 2023.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Lamini. Lamini docs dataset at hugging face, 2023. URL https://huggingface.co/datasets/lamini/lamini_docs/tree/main.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

Mingjie Liu, Teo Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, et al. Chipnemo: Domain-adapted llms for chip design. *arXiv preprint arXiv:2311.00176*, 2023.

Nick McKenna, Tianyi Li, Liang Cheng, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. Sources of hallucination by large language models on inference tasks, 2023.

OpenAI. Chatgpt, 2023a. URL https://openai.com/blog/chatgpt/.

OpenAI. Gpt-4, 2023b. URL https://openai.com/gpt-4/.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

John Schulman. Reinforcement learning from human feedback: Progress and challenges, 2023.

Yixuan Tang, Hwee Tou Ng, and Anthony Tung. Do multi-hop question answering systems know how to answer the single-hop sub-questions? In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 3244–3249, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.283. URL https://aclanthology.org/2021.eacl-main.283.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://ai.meta.com/llama/.

Logesh Kumar Umapathi, Ankit Pal, and Malaikannan Sankarasubbu. Med-halt: Medical domain hallucination test for large language models. *arXiv preprint arXiv:2307.15343*, 2023.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *arXiv preprint arXiv:2301.13848*, 2023a.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models, 2023b.

Ou Zheng, Mohamed Abdel-Aty, Dongdong Wang, Chenzhu Wang, and Shengxuan Ding. Traffic-safetygpt: Tuning a pre-trained large language model to a domain-specific expert in transportation safety. *arXiv preprint arXiv:2307.15311*, 2023a.

Shen Zheng, Jie Huang, and Kevin Chen-Chuan Chang. Why does chatgpt fall short in providing truthful answers. *ArXiv preprint, abs/2304.10513*, 2023b.