Not All Cars Have Radios:
Generics and Exemplars in Consumable LLMs

Mustafa Eyceoz
me2680@columbia.edu

Justin Lee
jjl2245@columbia.edu

Abstract

This is an extension of "Penguins Don't Fly: Reasoning about Generics through Instantiations and Exceptions" by Allaway et al. The initial work addressed the fact that many commonsense knowledge bases encode some generic knowledge, but rarely enumerate exceptions. To remedy this, a novel system was proposed to generate a dataset of generics and exemplars. Our work takes this initial concern a step further, analyzing the current level of understanding LLMs possess for generics and exemplars, and finding the optimal means to alleviate any lack of understanding. After initial experiments, it is clear that large models (like Llama-2-70B) do not have too much issue with question answering and reasoning pertaining to generics and exemplars. But smaller, consumable models (like Llama-2-7B) struggle immensely. There remains the question of whether this is a content and knowledge retainment issue, or a prompting issue. To explore and potentially remedy this, we have built two rival solutions: one utilizing Retrieval Augmented Generation alongside the "Penguin's Don't Fly" dataset, and the other utilizing Prompt Tuning via PEFT (Parameter-Efficient Fine-Tuning), both proving viable.

1 Introduction

1.1 Core Motivation

Before concerning ourselves with any Large Language Model (LLM) performance, we must first answer the initial question: why do we care whether AI can reason about generics and exemplars? Let us begin with a brief explanation of the linguistic concepts, followed by their significance.

A generic is a statement about members of some kind that is usually true, and helps to easily describe our world, but is not necessarily the case. Statements like "birds can fly", "humans have eyes", and "grass is green" are all assumptions that we take as normally the case, in order to form a basic understanding and organization of our surroundings. An exemplar is a related description of a specific member or instance to which a generic pertains. It is an example of the generic, one that either agrees with or counters it. Take "birds can fly" once again. Exemplars would be "sparrows can fly" or "penguins cannot fly".

To be able to reason about generics and exemplars is to be able to understand the world, and all that exists within it. Generics and exemplars serve as a solid reasoning exercise for AI. Does the system understand the concept of a counter-example or supporting evidence, and will it be able to answer questions with universal or existential quantifiers (all, some, and their respective negations)? If an LLM could make its way through this basic first-order logic and reasoning, it may serve as indication of whether the direction we are going with AI is conducive to reasoning and understanding, or if we have strayed a bit too far off this path.

1.2 Initial Discovery

After initial experimentation with LLMs on question answering and reasoning related to generics and exemplars, we have found that there is actually some cause for concern, but not in all cases. More specifically, when using the best and largest open-source SotA models like Llama-2-70B (Touvron et al., 2023), this is not really a problem. When using smaller models that can run on consumer or academic hardware, instead of requiring an A100+, like Llama-2-7B, models begin to have substantial struggles in both being able to recognize exemplars, and also in their reasoning surrounding generics and exemplars. For example, let's consider the following generic and associated exemplar:

- Generic: "Cars have radios"
- Exemplar: "Not all new cars have radios"

We begin by querying Llama-2-70B with: "Do all new cars have radios?" as well as "If cars have radios, do all new cars have radios?" In both cases, the response is acceptable. While it will also provide a lot of excess or unneeded information (and may not directly say "no"), it will usually contain some information to indicate what the correct answer is, like the following:

- "Some newer cars may not have a traditional radio tuner built into the infotainment system. Instead, they may rely on smartphone integration or streaming services to provide access to audio content."

When asking the same question to Llama-2-7B, however, the answers go from "imperfect but acceptable" to simply incorrect and inaccurate. The following responses are from the 7B model, once when prompted with "Do all new cars have radios?" and the other with "If cars have radios, do all new cars have radios?"

- "Yes, most new cars do come equipped with a radio as a standard feature. According to the NHTSA in 2019, 97% of new cars sold in the US had a radio as standard equipment. This is because radios are considered a basic feature in modern vehicles and are expected by many consumers."
- "Not necessarily. While many new cars do come equipped with radios, it is not a standard feature in all new cars. According to a survey conducted by the NHTSA in 2019, only about 70% of new cars sold in the United States came equipped with a radio as a standard feature."

Immediately there are some issues worth noting. First, the model in one case begins the answer with "yes", whereas in the other case begins with "not necessarily". This seems to be an issue with prompt sensitivity. The second, more concerning problem is regarding the reasoning it gave in both instances. In BOTH cases, it hallucinated a survey to back up its claim (the NHTSA produced no such survey), and even in its hallucination, the model was inconsistent, switching from 97% to 70%, depending on the case it was backing up. This points to a potentially greater problem with content and reasoning.

And so we arrive at the core investigation behind our work: are these issues due to the model being too small to retain and make connections to the content necessary to answer these questions, or is it simply the case that they need to be prompted much more specifically to better understand what is being asked? Furthermore, hallucination and knowledge grounding are wider issues in LLMs even beyond the context of generics and exemplars. Perhaps finding the best solution for this case could have more general applications.

1.3 Hypothesis Testing: Possible Solutions

We began by building an initial RAG (Lewis et al., 2021) implementation alongside a "mock dataset" (only two test files) to dynamically pull exemplars from relevant generics in order to better provide the 7B model the context it needs to answer our question. Using this system, we then asked both of the original prompts ("Do all new cars have radios?" and "If cars have radios, do all new cars have radios?") and received approximately the following response for both cases:

- "The answer to the query 'Do all new cars have radios?' is: False. The counterexample is 'New Teslas do NOT have radios.' This contradicts the generic statement 'Cars have radios' because Teslas are a type of car, and they do not have radios. Therefore, the statement 'Do all new cars have radios?' is false."

By using a system to dynamically search and inject relevant information needed to best address the given prompt and its relation to generics, we were able to produce a response that was both factually correct and grounded in real knowledge.

However, it may not be the case that a system like this is entirely necessary. A similar result is achieved if we craft the prompt carefully and specifically:

- PROMPT: Is the following statement true or false? If false provide a counterexample: "All new cars have radios"
- RESPONSE: The statement "All new cars have radios" is false. Counterexample: There are many new cars that do not have radios, such as electric or hybrid vehicles, or cars that are designed for specific markets where radio reception is not reliable (e.g., in areas with weak radio signals or in remote areas without access to radio broadcasts).

For this specific example, it would appear as though the problem can be solved with either route: providing appropriate context or by improving prompting.

1.4 Contribution

Mustafa Eyceoz planned and executed the initial designs and experiments for this project. He was then responsible for data collection and filtering, the baseline experiments, and the RAG solution. Justin Lee was responsible for the PEFT prompt tuning experiments, including model training and iteration, hyper-parameter tuning, and hardware acquisition and management. That being said, nothing was truly worked on in isolation, with both helping the other out at various points. Both managed evaluation for all solutions. The codebase is on GitHub[1]. Paper writing was mostly handled by Eyceoz, and most figures were created by Lee, with overlap in both of these tasks as well.

2 Related Work

This work is primarily focused on single-call LLM reasoning about generic statements, examples, and counter-examples. While there has been recent work on multi-step reasoning faithfulness generally (Creswell and Shanahan, 2022), there is little currently connecting LLMs and generics reasoning. That being said, there is a great deal of research in knowledge grounding, prompt optimization, and task learning that will be incredibly useful for testing our hypotheses.

Lewis et al. did a lot in their original work to show the value of Retrieval Augmented Generation for knowledge grounding in LLMs, and since then, many research papers, groups, and articles have come out supporting these claims (Martineau). With both current research support and our initial experiments, RAG seemed to align well with our knowledge-based hypotheses. In terms of efficient task learning, there is a lot of work supporting Parameter-Efficient Fine-Tuning (Liu et al., 2022). Some methods are actually built around prompt optimization via fine-tuning, which perfectly aligns with our prompt-based hypothesis.

The model being used throughout the experiment for querying and evaluation is Llama-2-7B. There are two notable prior works. Retrieval Augmented Generation (RAG) is the process of dynamically adding relevant context from a set of private data into an LLM query. In practice (Liu), one begins by vectorizing and indexing some desired set of data. Then, every time a user makes a query to an LLM, the query is first converted into a vector embedding, and then a similarity search (e.g. cosine similarity) is done across the vectorized private data to find the most relevant content, to be passed in as context for a more enhanced LLM query. The embedding model being used in the RAG solution is bge-small-en-v1.5 from BAAI (Xiao et al., 2023).

Prompt Tuning (a method of Parameter-Efficient Fine-Tuning, or "PEFT") (Lester et al., 2021) is similar to regular model fine-tuning, but instead of the model weights updating, it is instead a set of virtual embedded tokens as a layer on top of the model that get trained, to be attached as an extension for the prompts being passed into the LLM. Essentially, it is using fine-tuning to discover the best prompt for a given task. An algorithmic approach mirroring model training as an evolution of recent "prompt engineering" efforts.

3 Data

The dataset being used has been provided by Emily Allaway, lead author of "Penguins Don't Fly" (Allaway et al., 2023). It consists of all currently generated examples from templates 3-7, which includes both positive and negative exemplars. Included are ~95K exemplars, spread across the five templates. This breaks down into approximately 40K counter-examples (templates 3 and 4), and 55K positive instances (templates 5-7). These templates, from the original paper, indicate what types of exemplars we will see. In this case, if a generic is comprised of a kind and a property, these exemplars deal with: a subset of the kind (likely a member) and the property, the kind itself with a subset of the property, and potentially a member/subset of the kind with a subset of the property.

Each entry contains the following fields: `{generic,exemplar,exemplar_singular, prob_entail,prob_contradict,prob_valid}`. The fields `prob_entail` and `prob_contradict` indicate whether the exemplar is a counter-example or positive instance, and `prob_valid` is a confidence measurement for the sample. Even when filtering for only confidence above 95% and contradiction/entailment above 95%, we retain all counter-examples, and approximately 12K positive instances. For evaluation, ~5% of exemplars will be selected and utilized as a test set excluded from PEFT training.

Upon closer inspection of the dataset, we can see that the average, minimum, and maximum character length of a generic are 29, 10, and 95 respectively. The average, minimum, and maximum word

[1]https://github.com/Maxusmusti/llm-logic-experiments

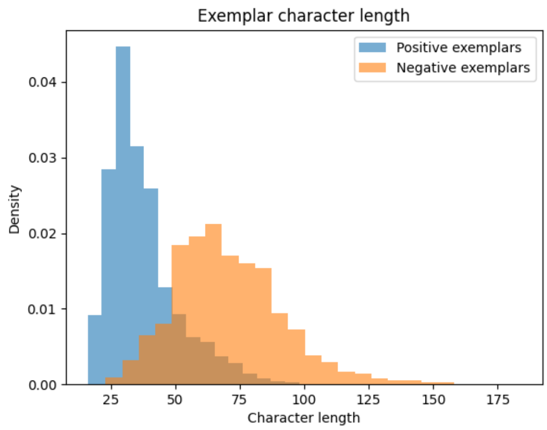

Figure 1: Exemplar Character Length Distribution

count of a generic are 4.5, 2, and 20 respectively. The average, minimum, and maximum character length of an exemplar are 62.8, 16, and 184 respectively. The average, minimum, and maximum word count of an exemplar are 11, 3, and 34 respectively.

Comparing the positive exemplars with the negative exemplars in the dataset, it's apparent that they have vastly different distributions in character length. The average, minimum, and maximum character length of a negative exemplar are 70.6, 23, and 184 respectively. The average, minimum, and maximum character length of a positive exemplar are 37, 16, and 109 respectively. See Figure 1 for these character length distributions.

4 Methods

4.1 Rival Solutions

To answer the question posed in the introduction, we built two rival systems to alleviate issues with generics and exemplars in SotA consumable LLMs:

- The first is a more complete RAG solution utilizing the entire generics and exemplars dataset generated by the method introduced in the "Penguins Don't Fly" paper.
- The second will be to use deep-learned prompt tuning, or PEFT as seen in "The Power of Scale for Parameter-Efficient Prompt Tuning," over the dataset of 95k generics/exemplars to train a set of embedded tokens to serve as the "optimal prompt" to be attached to queries for a better understanding of questions tied to generics/exemplars.

The goal is to build both solutions such that users can benefit from improved QA and reasoning in real-world applications, as well as to conclude which system serves as the better overall solution.

Prompt	Answer	Factual	Justifies
All emus have 3 toes	False, although most emus have 3 toes, there exists emus with less than 3 toes. Thus, not all emus have 3 toes.	Yes	Yes
All emus have 3 toes	True, there exists emus with less than 3 toes, but most emus have 3 toes. This means the statement is true.	Yes	No
All emus have 3 toes	False, not all emus have 3 toes; there is a sub-species of emus called "wolf emu" that are often born with 4 toes.	No	Yes
All emus have 3 toes	False. There has never existed an emu without 3 toes. Thus not all emus have 3 toes.	No	No

Figure 2: Explanation Evaluation Examples

4.2 Question Types

To be more specific about what we will be asking for generics reasoning, there will be two main question categories. The first is questions of universal quantification: "all" and its respective negation "not all". The second is questions of existential quantification: "some" and its respective negation "none". The questions will be about the generics themselves, with the addition of the quantifiers appropriately prepended.

4.3 Evaluation Metrics

These systems are to be evaluated on two fronts: the correctness of the answer (true/false), and the quality of the reasoning provided for the answer. While correctness is a straightforward evaluation, the reasoning will require a more sophisticated evaluation approach.

For correctness, the system answers correctly if and only if it responds with the correct true or false response. In some cases, the system may either not directly answer the question, or answer with both yes and no or true and false; these cases are not considered correct.

For explanation evaluation, we will be performing human evaluation on two binary metrics. The first is factuality: whether the facts and information presented in the explanation are verifiably true, or if they are incorrect/hallucinated. The second is justification: Whether the explanation validly leads to the same conclusion as provided in the answer. This incorporates both knowledge-grounding and natural language inference (NLI) feature analysis for provided reasoning. See Figure 2 for examples of explanation evaluations.

There is one unanswered question remaining: how do you compare a RAG solution, which could be feeding relevant exemplars directly to the LLM, to a PEFT solution that has never seen the correct answer? Well, each solution essentially serves as a parallel to our hypotheses. The RAG solution explores whether there is truly a knowledge gap and missing information about generics that the LLM needs access to for question answering. The Prompt Tuning (PEFT) solution explores whether the LLM simply needs to be prompted more precisely for reasoning about generics, with no additional information necessary. Consider the RAG solution to be an "oracle baseline" of sorts, where the LLM is spoon-fed the exact information it needs to answer the question, using a vast collection of generics and exemplars. The goal of the PEFT solution is to beat this oracle adversary, by learning an optimal prompt and using it alongside only the questions themselves to answer the same questions without any knowledge assistance. The PEFT solution is far more portable and generalizable, requiring no external knowledge source, and being tied to no underlying set of domains. If the PEFT solution can match or closely rival the RAG solution, then it shows that smaller LLMs do have the inherent knowledge and capability to reason about generics and exemplars, and simply need to be prompted well for the task. If the PEFT solution fails to come close to our oracle RAG, however, then it shows that there truly is a knowledge gap to be solved.

4.4 Baselines and Observations

To understand how both experimental systems improve upon the task, we will also evaluate the performance of the base model, Llama-2-7B, without either additional system (no PEFT prompt or RAG), on the test set.

We will also be looking further into implementation details of the two solutions. For example with PEFT, how many token embeddings should be used in the prompt extension? Prior work indicates that a smaller prompt leads to increased generalizability (Gonen et al., 2022). This will also apply to the final RAG-formatted prompts, when it comes to how many exemplars we need to retrieve for acceptable performance.

5 Experiments

The baselines, RAG system, and PEFT system were all developed and evaluated with the same sample of 4000 generics questions. 1000 generics were chosen at random from the full dataset,

BASELINE PROMPTS	Direct y/n	Reframed t/f
Universal	Answer with yes/no and an explanation. Is it true that all birds can fly? Answer with yes/no and an explanation. Is it true that not all birds can fly?	Express whether the statement is true or false and explain why. Statement: All birds can fly. Express whether the statement is true or false and explain why. Statement: Not all birds can fly.
Existential	Answer with yes/no and an explanation. Do some cars have radios? Answer with yes/no and an explanation. Is it never the case that cars have radios?	Express whether the statement is true or false and explain why. Statement: Some cars have radios. Express whether the statement is true or false and explain why. Statement: It is never the case that cars have radios.

Figure 3: Baseline Prompt Examples

and four statements were constructed from each ("All x" and "Not all x" for universal quantification, "Some x" and "It is never that case that x" for existential quantification) alongside a label as to whether the statement is true or false. The following subsections describe how each of the initial systems were constructed.

5.1 Baseline

As a base for comparison, we queried Llama-2-7b directly, both without any sophisticated prompting (direct generics questioning) as well as with some basic prompt engineering (restructuring the problem as true/false, asking the model to think carefully and explain, etc.). For direct generics questioning, to avoid awkward phrasing, we tested with three question formats: "Do all x / Do not all y", "Is it the case that all x / Is it the case that not all y", and "Is it true that all x / Is it true that not all y", always asking for a "yes" or "no" answer. Examples can be seen in Figure 3. We then took the results from the best performing question formation.

5.2 Retrieval Augmented Generation

In order to have a functioning RAG solution, we first need to have an embedding store, or an indexed collection of embeddings, upon which we can do exemplar retrieval. To build this, we started with the full exemplar dataset. We then filtered on exemplar validity with a 95% confidence threshold, and then removed all columns other than those for the generic and exemplar. We reformatted each sample to be labeled independently, i.e. `{generic:...,exemplar:...}`, and then embedded each formatted sample using bge-small-en-v1.5. Once all the embeddings had been generated, they were all written alongside indices that correspond to the original text sample. These indexed embeddings were then stored in JSON files, to be loaded into memory upon initialization of the RAG system. Note that an in-memory approach was uti-

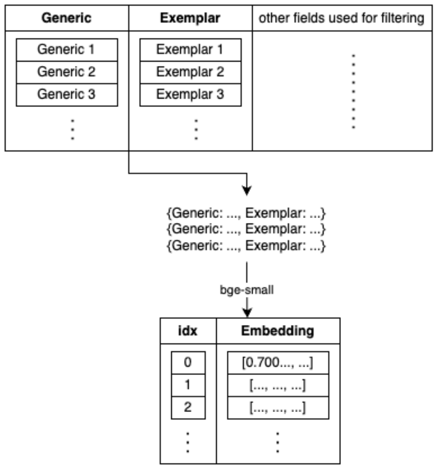

Figure 4: RAG Data Processing

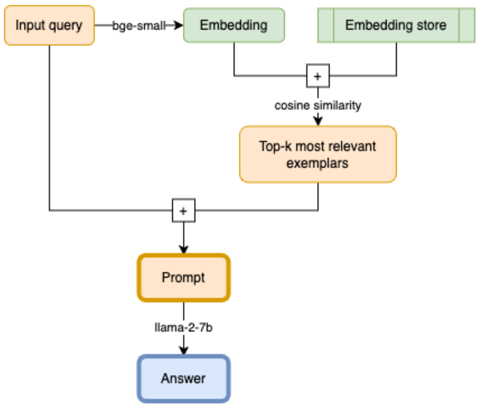

Figure 5: RAG Implementation

lized due to the small size of the total dataset (on the order of magnitude of MBs), but one could easily leverage a dedicated vector database alongside a RAG framework like LlamaIndex if required due to larger data requirements. Figure 4 illustrates the data preprocessing flow.

Once the embedding store was generated, we could now utilize retrieved exemplars in-context before querying Llama-2-7b. Every time an input query is passed into the system, the query is embedded (also with bge-small-en-v1.5), and cosine similarity search is used to retrieve the top-k most relevant exemplars. For our current system, we experimented with k=2, k=3, and k=5, and will be using k=5 for results comparison, as it proved to have the best response quality. Once we have the top matches, they are placed into the final prompt to be passed as the full query, as seen in Figure 5.

Two experiments were done with this system. The first was mirroring the direct question baseline, placing the context into the prompt with explana-

```
TRUE/FALSE RAG PROMPT:
Below are generic statements, followed by examples
(exemplars) that entail or contradict the statement:
---------------------
{GENERIC/EXEMPLAR PAIRS}
---------------------
Using this information, answer whether the below
statement is true or false carefully
Statement: {STATEMENT}
Answer:

YES/NO RAG PROMPT:
Below are generic statements, followed by examples
(exemplars) that entail or contradict the statement:
---------------------
{GENERIC/EXEMPLAR PAIRS}
---------------------
Using this information, answer the below question
carefully (yes/no) and explain
Question: {QUESTION}
Answer:
```

Figure 6: RAG Prompts

tion and then asking the yes/no generics question. The second experiment featured the same reformatting of the second baseline, first explaining what the exemplars represent, followed by a re-framing of the question as true/false. Both formats can be seen in Figure 6.

5.3 Prompt Tuning (PEFT)

In order to have a functioning prompt tuning solution via PEFT, we first loaded a frozen pre-trained Llama-2-7b model from Hugging Face. The dataset of generics and exemplars was then prepared by randomly splitting it with 80% being used for training and 20% being used for testing. The two dataset splits then went through several pre-processing steps such as ensuring that the same generic did not appear in both the train and test splits, tokenizing the generics and exemplars, padding them with padding tokens, creating attention masks, and converting them to PyTorch tensors before loading them into data loaders.

The soft prompt trainable tokens of our PEFT solution were trained by attaching them to the tokenized queries taken from our dataset, passing it through the frozen pre-trained Llama-2-7b model, computing the loss between the ground truth and the model output, and then applying gradient updates through backpropagation as seen in Figure 7. After the training of the tunable soft prompt tokens, they could be easily used when querying Llama-2-7b by attaching them to any tokenized query.

In our PEFT implementation, we defined several configuration parameters: we tested both 8 and

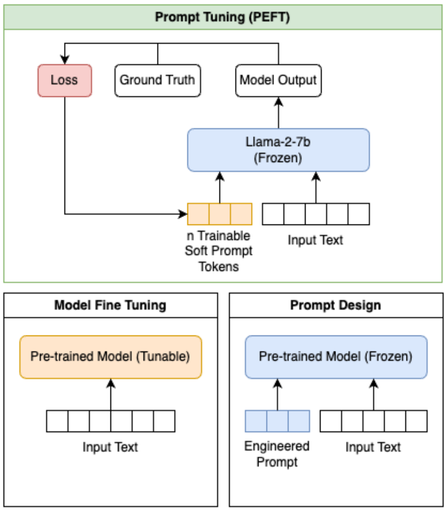

Figure 7: PEFT Implementation

Table 1: RAG Correctness Accuracy (vs Baselines) for Universal Generics QA

	Base(y/n)	Base(t/f)	RAG(t/f)
Win %	68.25	87.70	**96.95**
Tie %	5.10	2.00	**0.75**
Loss %	26.65	10.30	**2.30**

Table 2: RAG Correctness Accuracy (vs Baselines) for Existential Generics QA

	Base(y/n)	Base(t/f)	RAG(y/n)
Win %	89.15	74.85	**94.15**
Tie %	7.15	7.55	**1.75**
Loss %	**3.70**	17.60	4.10

4 trainable tokens (∼32.7k and ∼16.3k trainable parameters) for the tunable soft prompt, used a learning rate of 0.03, training duration of 1 epoch, and batch size of 12. We used Adam optimizer with a linear learning rate scheduler. Finally, FP16 mixed-precision training was used which requires less memory and enabled faster training and testing of our model.

6 Results

After running experiments for the baselines, RAG solution, and PEFT solution, we evaluated both correctness and explanation quality.

6.1 RAG Solution Results

The RAG solution had significant improvements over the baseline in both experiments. In Table 1, we can see how the RAG solution compares to both baselines for questions of universal quantification. In Table 2, we can see how the RAG solution fares when compared to both baselines with respect to questions of existential quantification. Note that a win denotes that the system answered correctly, a loss denotes that the system answered incorrectly, and a tie indicates that the system either did not directly answer the question, or (more commonly) answered both correctly and incorrectly, by answering with both yes and no or true and false.

In both experiments we see noticeable improvement from the baselines to the RAG solution. For universal quantification, we improved from the direct y/n baseline of 68.25% accuracy to a much more confident 96.96% accuracy with the t/f RAG implementation. The t/f baseline came closer, but the RAG result still had considerably more wins, with approximately 11x fewer losses than the y/n baseline, and 4x fewer losses than the t/f baseline. It is worth noting that with some slight prompt engineering, going from 68% to 88% accuracy is a noteworthy jump. This means that while the experiment did illustrate the power of RAG, it also highlights the value of more precise prompting. Even so, the RAG solution clearly outperforms both baselines on correctness accuracy for universal generics question answering. For existential quantification, the story is similar, but not quite as clear. For questions of "some", the baselines on average performed better than with the "all" questions. Most interestingly, though, is that the better prompting strategy flipped. The direct y/n questions did better than t/f across the board here, with the y/n baseline even coming quite close to the RAG solution and having slightly less losses. This once again shows the variance prompting can cause, but the value of RAG is not to be discounted here. Even for existential quantification, a more noticeable difference can be seen when looking at the explanation quality.

Once again, in Table 3, we can see how the RAG solution's explanations compare to the best performing baseline for questions of universal quantification. In Table 4, we can see how the RAG solution's explanations fare when compared to the best performing baseline with respect to questions of existential quantification. Note that "% Factual" represents what percent of explanations contained only verifiably true facts and information, and "% Justified" represents what percent of explanations

Table 3: RAG Explanation Quality (vs Baseline) for Universal Generics QA

	Base	RAG
% Factual	81.70	**99.60**
% Justified	79.20	**98.30**

Table 4: RAG Explanation Quality (vs Baseline) for Existential Generics QA

	Base	RAG
% Factual	84.50	**98.90**
% Justified	94.00	**95.15**

validly led to the same conclusion as the answer given.

Similarly to correctness, there is a more noticeable gap in universal quantification than in existential quantification. Regardless, both show the true value of the RAG solution. Since it is using the validated exemplar dataset to answer the questions, the knowledge presented in the explanations is more grounded by design. This is clearly reflected by the results, with about 99% of RAG explanations being factually correct, compared to the baseline results of 81.7% for universal and 84.5% for existential. While the justification accuracy was also better for both RAG systems, it is worth noting that in the existential case, the justification accuracy of the baseline actually came quite close, with 94% as compared to 95% for RAG.

While we don't expect the PEFT Prompt Tuning solution to reach the same level of factual correctness, if it can match the RAG solution in both answer correctness and justification quality, while improving on the baseline's factual correctness, it may be a compelling case for the prompt hypothesis. As seen by the results, though, the RAG solution definitely sets a high bar.

6.2 PEFT Solution Results

When it comes to answer correctness, the PEFT prompt tuning solution also had significant improvements over the baselines, with both 4 and 8 trained soft prompt tokens. The PEFT model trained with 4 virtual tokens will be referred to as PEFT(4) and the one with 8 virtual tokens will be referred to as PEFT(8).

In Table 5, we can see how it compares to the best performing baseline for universal quantification. In Table 6, we can see how it compares to the best performing baseline for existential quantification. Note again that a win denotes that the system answered correctly, a loss denotes that the system answered incorrectly, and a tie indicates that the system either did not directly answer the question, or answered both correctly and incorrectly, by answering with both yes and no or true and false.

Table 5: PEFT Correctness Accuracy (vs Baseline) for Universal Generics QA

	Base	PEFT(4)	PEFT(8)
Win %	87.70	99.43	**100.00**
Tie %	2.00	0.38	**0.00**
Loss %	10.30	0.19	**0.00**

Table 6: PEFT Correctness Accuracy (vs Baseline) for Existential Generics QA

	Base	PEFT(4)	PEFT(8)
Win %	89.15	97.80	**100.00**
Tie %	7.15	0.00	**0.00**
Loss %	3.70	2.20	**0.00**

We see noticeable improvement from the best performing baselines to the PEFT solution. The baseline was correct 87.70% and 89.15% of the time on universal and existential generics respectively, but both of the PEFT configurations resulted in greater accuracy. Even more notably, both PEFT(4) and PEFT(8) outperform the RAG correctness accuracy. PEFT(8) achieved perfect accuracy in all regards with no ties and no losses for both unviersal and existential generics. PEFT(4) comes close to 100% accuracy, but falls short with 99.43% accuracy for universal generics and 97.80% accuracy for existential generics. Interestingly, PEFT(4) achieves 0% in ties for existential generics which signifies that the model never answered indecisively and always answered the question directly.

In Table 7, we can see how the PEFT solution's explanations compare to the best performing baseline for questions of universal quantification. In Table 8, we can see how the PEFT solution's explanations fare when compared to the best performing baseline with respect to questions of existential quantification. Note again that "% Factual" represents what percent of explanations contained only verifiably true facts and information, and "% Justified" represents what percent of explanations validly led to the same conclusion as the answer

Table 7: PEFT Explanation Quality (vs Baseline) for Universal Generics QA

	Base	PEFT(4)	PEFT(8)
% Factual	81.70	82.32	**85.76**
% Justified	79.20	95.82	**99.05**

Table 8: PEFT Explanation Quality (vs Baseline) for Existential Generics QA

	Base	PEFT(4)	PEFT(8)
% Factual	84.50	**97.80**	93.68
% Justified	94.00	95.60	**100**

given.

Similarly to correctness, we see improvements from both PEFT configurations compared to the best performing baselines in terms of factual correctness and justification quality. Both PEFT(4) and PEFT(8) had a higher percentage of factually correct explanations compared to baselines for both the universal and existential generics cases. However, it is worth noting that in the universal generics case, the factual accuracy of baselines came quite close to the PEFT models, with the baseline achieving 81.70% factual while PEFT(4) achieved 82.32% and PEFT(8) achieved 85.76%. Another result worth noting is that PEFT(8) beats PEFT(4) in nearly all measures except one: PEFT(4) achieves 97.80% factually correct explanations for existential generics while PEFT(8) achieves 93.69%; nevertheless, they both outperform the best performing baseline which had 84.50%. When compared to the RAG results, PEFT(8) also clearly does better in justification quality, making it seem to be the overall top performing solution out of all the experiments, winning in both correctness and justification. When it comes to factuality, though, the RAG solution still does best, as expected, by utilizing an external knowledge source.

6.3 Pattern Breaking

While the results for the PEFT Prompt Tuning approach look incredible, it is important to understand their scope before reaching any conclusions. What we have specifically created and demonstrated here is a tuned prompt and model that performs well for questions about generics, and the task of using examples and counterexamples to support claims. This is by no means a general solution to all LLM reasoning, and we do not wish to pose this as such.

We had initial concerns that the training was specifically learning a lexical pattern matching. After all, when it comes to generics, they are "usually" true. Which means that they are never "always true", never "never true", and always "sometimes true". Thus, statements with the qualifier "all" are always false, "some" are always true, "never" are always false, etc. Thankfully, we were able to avoid this by keeping our tuned soft prompt short (maximum eight virtual tokens) to encourage generality and leave little room for hard-coding (Gonen et al., 2022).

We confirmed this not only by looking at the explanation quality, but by actively testing whether the model could correctly break the pattern described above. It was successfully able to do so, as seen in the following examples. For the query "Some apples that are red are blue", the PEFT model correctly responds with "False, all apples that are red are not blue." For the query "No red apples are blue" the PEFT model correctly responds with "True, all apples that are red are not blue." Something interesting worth noting is that PEFT(8) was better than PEFT(4) in this regard, despite having the longer soft prompt.

So then why do we caution the reader against assuming that we have a general reasoning solution? Well, while it is not taking any lexical pattern matching shortcuts, it is clear that we have still learned a more specific task than "general reasoning." The model does quite well with things that are never the case; to "It is never the case that 1+1=123", the model responds "True, it is never the case that 1+1=123." However, it does struggle more in admitting that some things are always the case. This holds true whether we fit our usual pattern or not, prompting for "all x", or something very different like "it is always the case that x". For example, when prompted "All circles are round", the model responds "False, circles are not always round." Similarly, to "It's always the case circles are round", the model responds "False, circles that are irregular are not round." To "All people start as babies", the model responds "False, people who are born as adults do not start as babies." And when prompted "It is always the case that 1+1=2", the model responds "False, it is not always the case that 1+1=2."

The performance on non-generic-based statements is still quite poor, and so our system is by no means an all-around reasoning model. Perhaps by adding some tautologies to the training data, it could be improved, but for now at least it is not the case. For now, it has simply learned how to reason about generics, and to use counterexamples to show when things aren't the case, and to use examples to support when things are. That being said, by no means should we take away any credit from its performance within that domain. We have shown that the system has not taken any lexical shortcuts, and that it genuinely has learned to reason about generics and provide higher quality explanations with excellent justification and reduced hallucination.

7 Error Analysis

7.1 Baseline

There were a few common mistakes that were repeated by both of the baselines. The biggest failure in terms of factual correctness was exactly what we saw in the introduction. The model will sometimes use studies and statistics to substantiate its claims, and these almost always turn out to be hallucinated. When it came to justification, an interesting issue was with task consistency. The response would begin answering one question, and finish with another. There were also issues in connecting presented evidence to the provided answer. The response, would say a statement is true, then provide a reason as to why it is false, or vice-versa. This is also what led to a lot of "ties", seeing responses that started with one answer, then based on evidence concluded with another. For example, when asked "Is it true that not all bird bones are hollow?", the model would answer with "no", but ultimately conclude with "While many bird bones are hollow...not all bird bones are this way".

Another interesting observation was that the baselines on average did better with questions of existential quantification than with universal quantification. It seems that this is due the model simply having an easier time coming up with one example of a generic than it does coming up with a counterexample. This in turn is likely due to the fact that things that are "usually" true will inherently have far more examples than counter-examples.

7.2 RAG

While the RAG results were strong, the brittleness of the system can be seen when analysing the common errors. The RAG solution seemed to work whenever it provided the correct knowledge needed to the model (as was the goal of the oracle system), and its errors were often when it failed to do so. Since our dataset did have more counterexamples than examples, there were scenarios in which the RAG system would provide only counterexamples for a question that needed an example, and sometimes the opposite occured (albeit far more rarely once we increased to k=5). For example, when asked "Is it never the case that bald eagles have transparent eyelids?", the model was provided with only counterexamples like "bald eagles that are very young do not have transparent eyelids" or "golden eagles do not have transparent eyelids" or "eagles that are incubating eggs do not have transparent eyelids", resulting in the model deciding that it was in fact never the case.

The RAG solution provided a baseline for whether the model could perform well when provided with the required relevant information, and it did that well. There were some justification failures that led to incorrect answers, but less than the baseline. What these errors highlight, however, is that RAG is not the general solution to improved reasoning about generics, at least not in this form. What it does incredibly well is knowledge-grounding via external referencing and verification. With slightly more sophisticated prompting for improved justification, it served as a powerful oracle adversary to the PEFT, but for questions beyond the domain of our dataset, it would ultimately be incompatible.

7.3 PEFT

Some shortcomings of the PEFT solution are apparent when analysing its common errors. In terms of accuracy errors, the PEFT(8) model had 100% accuracy, but we may look at PEFT(4) to see what happens with too few soft prompt tokens. First, looking at instances where PEFT(4) had provided an answer that was a tie, the model would often times not answer the question at all. For example, for the query "Not all sharks are oviparous", the model outputs "Nobody can resist the charms of shark."

Interestingly when PEFT(4) makes accuracy errors, it also has trouble connecting presented evidence to the provided answer. In other words, an incorrect true/false answer is often times followed by a factual and correctly justified explanation for the opposite answer. For the query "No frogs use tongues", the model outputs "True, some frogs use the tongue to catch insects." Similarly, for the query "No cats have long tails", the model outputs "True, some cats have long tails." The model output would have been perfectly correct, factual, and justified if had outputted the opposite answer with the same explanation. These examples in addition to the fact that PEFT(8) did not make any accuracy errors leads us to believe that it's the lack of additional virtual tokens in PEFT(4) that results in the occasional accuracy error.

Both PEFT configurations make similar errors for explanation quality. For factual correctness, the model occasionally hallucinates. For example, the PEFT(8) model gave "True, some bears reach heights of over 20 feet" as an explanation which is factually incorrect, as the tallest bear ever recorded is just over 10 feet tall. Similarly, the PEFT(4) model gave "False, lions do not eat small mammals." as an explanation which is factually incorrect, as lions sometimes eat rodents in times of shortage. Still, by no longer hallucinating citations and statistics, and focusing on examples and counterexamples, factuality improves noticeably over the baseline.

For cases where the explanation lacks in justification quality, both PEFT configurations typically provide counterexamples that do not refute the query. For example, for the query "All pigment absorbs visible light", the model responded with "False, hydrogen does not absorb visible light." Though hydrogen does not absorb visible light, hydrogen is not a pigment and therefore the explanation does not justify its answer correctly. Similarly, for the query "All caterpillars have jaws", the model responded with "False, monarch butterflies do not have jaws." Though monarch butterflies do not have jaws, the question was about caterpillars which monarch butterflies are not. In other rare cases, it seems like the model simply misunderstands the query. For example, when given the query "Some cats have long tails", the model responds with "True, some cats have long hair on their tail."

8 Conclusions, Limitations, and Future Work

We came into this work with a question about the limitation of consumable LLMs in generics reasoning: do they lack the required knowledge to reason about generics, do they just need to be prompted more precisely for the task, or are they simply incapable of consistent performance in generic reasoning? We conclude that the answer to this question is fortunately a positive one. Based on experiments with RAG and PEFT Prompt Tuning, we have found that models like Llama-2-7b are in fact capable of impressive generics reasoning, and are simply just more sensitive to prompting.

By attempting to optimally tune a prompt for the task, we were capable of achieving great correctness accuracy and explanation quality, and had better results in most cases than the oracle RAG system for testing the knowledge hypothesis. The only caveat here is that if one wants to truly prioritize knowledge grounding and factual correctness, it would appear that either external fact verification or a more complex system is still required. This is shown by the RAG solution's near perfect factual correctness by design, as compared to the 86%-94% factual correctness exhibited by our best performing PEFT model. Regardless, when it comes to generics reasoning, consumable LLMs are more than capable, and it would likely be worthwhile to continue reasoning research on this class of models.

This final paragraph is not to be included in a published version of this paper, but rather a list of pre-publication next steps. First, we will share these results with Emily Allaway. Afterwards, there are still a couple more experiments to run before publication. Most notably is testing these results with additional smaller models (GPT-2, FLAN-T5, etc.), and verifying that the general findings still hold. From there, we can continue to add graphs, examples, and further refine the paper based on feedback.

References

Emily Allaway, Jena D. Hwang, Chandra Bhagavatula, Kathleen McKeown, Doug Downey, and Yejin Choi. 2023. Penguins don't fly: Reasoning about generics through instantiations and exceptions.

Antonia Creswell and Murray Shanahan. 2022. Faithful reasoning using large language models.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

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. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Haokun Liu, Derek Tam, Muqeeth Mohammed, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*.

Jerry Liu. Llamaindex: Vector-based retrieval augmented generation.

Kim Martineau. What is retrieval-augmented generation? (ibm research).

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. 2023. Llama 2: Open foundation and fine-tuned chat models.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding.