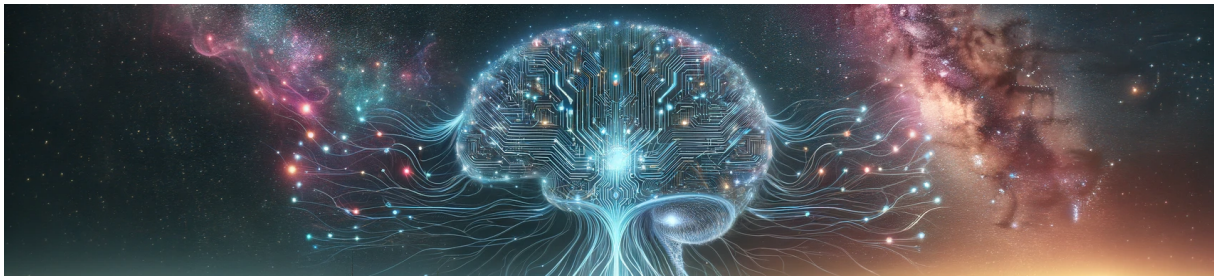
Into the Unknown: Self-Learning Large Language Models

Teddy Ferdinan **Jan Kocoń** **Przemysław Kazienko**
{teddy.ferdinan,jan.kocon,kazienko}@pwr.edu.pl
Department of Artificial Intelligence
Wroclaw Tech



Abstract

We address the main problem of self-learning LLM: the question of what to learn. We propose a self-learning LLM framework that enables an LLM to independently learn previously unknown knowledge through self-assessment of their own hallucinations. Using the hallucination score, we introduce a new concept of Points in The Unknown (PiUs), along with one extrinsic and three intrinsic methods for automatic PiUs identification. It facilitates the creation of a self-learning loop that focuses exclusively on the knowledge gap in Points in The Unknown, resulting in a reduced hallucination score. We also developed evaluation metrics for gauging an LLM's self-learning capability. Our experiments revealed that 7B-Mistral models that have been finetuned or aligned are capable of self-learning considerably well. Our self-learning concept allows more efficient LLM updates and opens new perspectives for knowledge exchange. It may also increase public trust in AI.

1 Introduction

Commonly, Large Language Models (LLMs) are pre-trained on large textual corpora and then fine-tuned using additional data to be better adjusted to a given policy or domain. Simultaneously, other quasi-learning methods have been developed, which are based on additional knowledge provided to the model directly in prompts, especially Retrieval Augmented Generation (RAG). In this paper, we explore a different concept: Self-learning

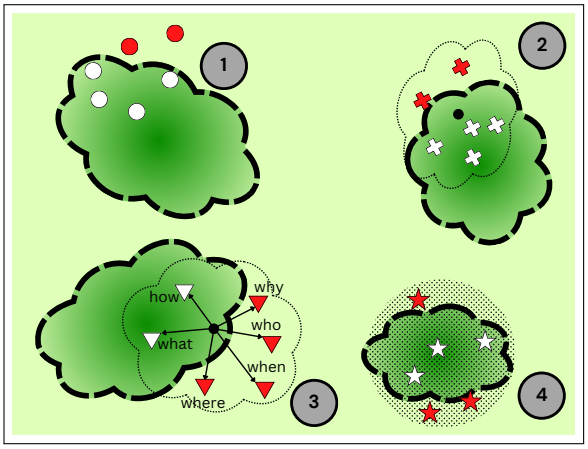


Figure 1: The illustrative space of human knowledge embeddings reduced to two dimensions. It visualizes our four methods for the identification of Points in the Unknown (PiU), later exploited in the self-learning loop. Dashed lines are the borders of the Known regions (darker green) – hallucination score thresholds. Out of them are the Unknown regions (lighter green). White points indicate prompts related to knowledge that the model already knows, while red points indicate PiUs. Different shapes depict different methods: (1) circles represent extrinsic (external) triggers, i.e., either user prompts or trending queries; (2) crosses denote open questions-prompts generated by the model itself within a given topic represented by a dotted line; (3) triangles represent the induced questions generated within a topic using 5W+1H; (4) stars indicate the random sampling by selecting random points in the embedding space.

LLMs, i.e., persistent acquisition of new knowledge by the model without data provision. Such a process requires several steps: (1) identification

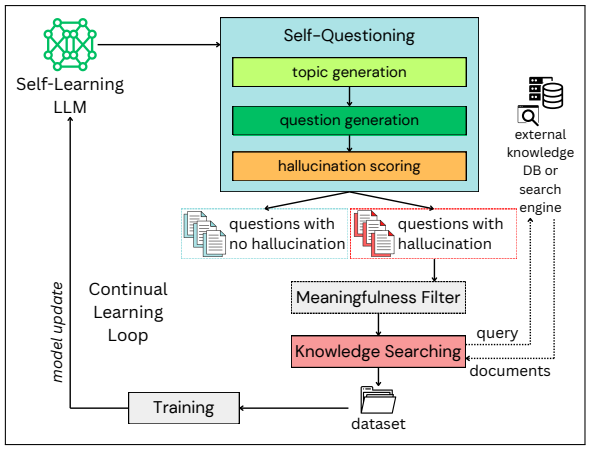


Figure 2: Illustration of Self-Learning LLM with intrinsic inspiration.

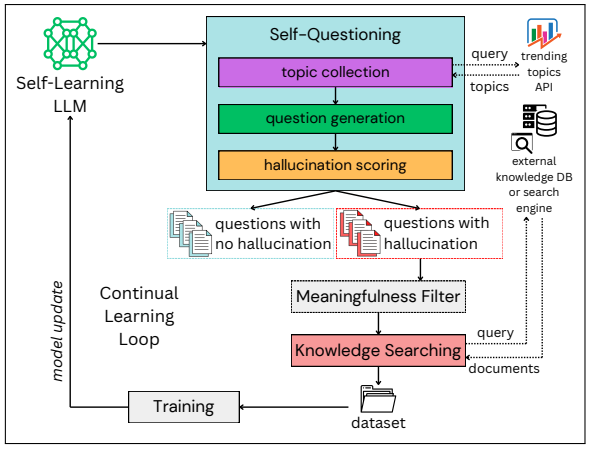


Figure 3: Illustration of Self-Learning LLM with extrinsic inspiration.

of what knowledge to learn, (2) meaningfulness filter to validate whether identified knowledge is worth and reasonable to learn, (3) searching for and gathering new relevant data, and (4) continual model training. Here, we want to break the assumption that the knowledge for fine-tuning must be inflicted beforehand. Simultaneously, we want to exclude cases in which the model learns the knowledge it already knows. For that purpose, we introduce a novel process component: identification of unknown knowledge based on our new concept, Points in the Unknown (PiUs), and placing this in a self-learning loop.

Our contribution presented in this paper covers: (1) definition of Points in the Unknown (PiUs) using the hallucination score; (2) one extrinsic (External Prompt) and three intrinsic (Open and Induced Generation, Oracle-selected) methods to identify PiUs; (3) introduction of Meaningfulness Filter; (4) metrics to gauge the capability of a model to conduct self-learning: Curiosity Score, Knowledge-Limit Awareness Score, and Self-Learning Capability Score; (5) design of the self-learning LLM using methods for identification of PiUs, Meaningfulness Filter, knowledge searching, and model continual learning; (6) experimental validation; (7) software and data publication for reproducibility.

2 Related Work

Hallucination in the context of LLM is the problem of nonsensical or unfaithful information produced by a generative model. Some works have studied the causes of hallucination (Kandpal et al., 2023; Lee et al., 2022; Onoe et al., 2022), as well as the detection methods (Lin et al., 2022; Min et al., 2023; Azaria and Mitchell, 2023; Manakul et al., 2023; Cao et al., 2023; Yin et al., 2023). Some solutions for overcoming hallucination are proposed in (Ji et al., 2023b; Luo et al., 2023; Li et al., 2023; Ji et al., 2023a; Liu and Sajda, 2023; Liang et al., 2024; Tian et al., 2024). Meanwhile, in Retrieval Augmented Generation (RAG) (Lewis et al., 2020), hallucination is avoided by supplying the prompt with some context retrieved from an outside source, allowing more factual generation without updating the model's knowledge.

Continual Learning is a training paradigm where the model is subjected to various tasks sequentially; in the context of LLM, the tasks typically comprise domain-specific datasets (Jang et al., 2022; Jin et al., 2022; Ke et al., 2023). One challenge is preventing catastrophic forgetting, in which the model loses knowledge from previous tasks (McCloskey and Cohen, 1989; Ratcliff, 1990). Solutions for adding or editing knowledge while avoiding catastrophic forgetting have been proposed (Kirkpatrick et al., 2017; Zhu et al., 2020; Sinitsin et al., 2020; Ke et al., 2021; Mitchell et al., 2022).

3 Why Self-Learning is needed

Hallucination is a serious problem that hinders many LLM applications. One of its main reasons is the model's lack of knowledge on a given topic. This problem is typically overcome by finetuning, i.e., additional learning using new data. Simultaneously, determining what the model already knows and what it does not know yet is very difficult, especially if there is limited information on the model's past training data. Because of this, finetuning is often a very inefficient process. If finetuning focuses on knowledge already acquired by the model,

it does not solve the hallucination problem. Then, we needlessly waste a lot of computing resources by merely repeating known knowledge. Therefore, it is essential to identify the knowledge known and unknown to the model and, in the finetuning process, concentrate only on the unknown.

4 The Concepts of The Known and The Unknown, Point in the Unknown (PiU)

We introduce the concepts of *The Known* and *The Unknown* in relation to the knowledge space to define the problem more precisely. Human knowledge is an abstract space formed by vectorial knowledge representations; each point in the space represents an atomic piece of knowledge. ***The Known*** refers to an area in the human knowledge space where our LLM does not hallucinate, i.e., it possesses the knowledge related to this region. Then, we can define each point in such an area as *Point in the Known* (PiK, plural: PiKs). On the other hand, ***The Unknown*** refers to an area in the human knowledge space where our LLM hallucinates; each point in such a region is called *Point in the Unknown* (PiU, plural: PiUs). A PiU represents knowledge that our LLM lacks, which we want the model to identify and acquire.

Finding the boundaries between *The Known* and *The Unknown* is non-trivial, but some hallucination detection methods offer practical scoring systems that can be utilized to approximate them. One such method is SelfCheckGPT (Manakul et al., 2023), i.e., a sampling-based hallucination detection method. It checks the consistency (variance) of multiple generated responses to a given prompt (a tested point x). It outputs a hallucination score $h(x) \in [0, 1]$, where $h(x) = 1$ indicates that the LLM response to x is entirely hallucinated (lack of any knowledge of x) and $h(x) = 0$ means no hallucination (certain knowledge of x). The median between the two values, $LIMIT = 0.5$, would serve as an intuitive and reasonable constant threshold to approximate the boundary. Therefore:

$$Known = \{x : h(x) < LIMIT; x \in HKS\}$$
$$Unknown = \{x : h(x) \geq LIMIT; x \in HKS\}$$

where HKS is the set of all possible human knowledge representations (points), i.e., the human knowledge space, $LIMIT = 0.5$. As mentioned before, $PiK \in Known$ and $PiU \in Unknown$. Note that $Known \cap Unknown = \emptyset$ and $Known \cup Unknown = HKS$, even though the sizes of $Known$ and $Unknown$ may change after training.

Alternatively, we could have exploited some other methods for $h(x)$, like SAPLMA (Azaria and Mitchell, 2023). However, it would require access to the model's internal features, limiting its feasibility. Therefore, we have remained at the function from (Manakul et al., 2023).

5 Methods for Identification of PiUs

Identification of PiUs can be done by evaluating the model's hallucination scores $h(x)$ on some questions-prompts x. These questions, or at least the inspiration for these questions, can come from outside the system, supplied by an external entity. In such cases, the method is considered to have an *extrinsic* nature. On the other hand, we can create a built-in *oracle* inside the system that guides automatic question generation in a bicameral-mind manner (Jaynes, 1990), in which case the method is deemed *intrinsic*. We propose one extrinsic and three intrinsic methods for PiU identification, Figure 1. All of them differently select or generate candidate points x (question-prompts), which, if tested according to $h(x) \geq LIMIT$, may be identified as PiUs.

5.1 External Prompt (extrinsic)

There are some existing ideas related to collecting prompts that cause hallucination (are unknown) and constructing a dataset based on them for finetuning (Zhu et al., 2020; Cao et al., 2023; Tian et al., 2024; Liang et al., 2024). However, they require manual curation of the prompts collected from users or datasets, so the model does not learn fully independently.

In our approach, we utilize an external API to collect trending topics, which are used as inspiration for formulating concrete questions. Every item returned by the API is a list of related phrases; we treat each list as a single topic. Then, the model is asked to generate a specific question x relevant to each topic. Next, the model is asked to answer those questions in order to evaluate its hallucination $h(x)$. Notably, an oracle is still used in the system to control question generation and hallucination scoring, yet the topics that trigger the questions come from outside the system. Unlike the previous ideas, this approach allows the model to continuously learn by itself as long as the external API is available; however, the tested space is limited only

to trending topics provided by humans.

5.2 Open Generation (intrinsic)

In this method, the oracle asks the model to propose some topics to learn about. Then, the oracle asks the model to consider those topics and formulate one question (x), to which the model believes it does not know the answer. Finally, the oracle asks the model to answer the question x in order to evaluate its hallucination $h(x)$. This method does not require any external entity to work with.

5.3 Induced Generation (intrinsic)

It is based on *Five Ws* and *How*, which are widely considered basic questions for information comprehension and data gathering. Here, the oracle also asks the model to propose some topics. Then, the oracle asks the model to formulate a question x using a particular question word; this is repeated six times for *what*, *who*, *why*, *where*, *when*, and *how*, resulting in six different questions. Finally, the oracle also asks the model to answer the questions and evaluate its hallucination $h(x)$. This method also does not require any external inspiration.

5.4 Oracle-Selected (intrinsic)

This method starts by constructing a topic embedding space, which contains all candidate topics represented in a vectorial form. Then, the oracle randomly selects a point in the topic embedding space and samples the nearest neighbors to that point. This results in a set of oracle-selected topics. Next, the oracle asks the model to consider those topics and formulate one question x. Afterward, the oracle asks the model to answer this question and evaluates the hallucination $h(x)$.

6 Self-Learning LLM

Self-Learning is a process where our LLM identifies its own PiUs, searches for the knowledge related to these PiUs, and trains itself on the collected data. It conjures an LLM capable of independent learning to fill the gaps in its own knowledge. Self-learning is made possible by incorporating Self-Questioning – which implements one of the methods for the identification of PiUs – with Knowledge Searching and Model Training in a continuous loop. Self-Learning LLM with an intrinsic method is illustrated in Figure 2, while Self-Learning LLM with an extrinsic method is illustrated in Figure 3.

Self-Learning has some similarities to traditional Continual Learning. The key difference lies in the LLM system, which asks questions by itself and evaluates whether it knows the answer or not. Another difference is the dynamic and integral dataset construction process in Self-Learning; in traditional Continual Learning, the dataset construction process is typically out of the scope. Nevertheless, training techniques and model architectures from traditional continual learning can possibly work in Self-Learning to allow more efficient training and avoid catastrophic forgetting.

6.1 Self-Questioning

Self-Questioning is generally performed through topic generation (or topic collection), question generation, and hallucination scoring. Depending on the selected method, the logical implementation of Self-Questioning may differ. Appendix A provides illustrations of such logical implementations with External Prompt, Open Generation, Induced Generation, and Oracle-Selected.

Self-Questioning is repeated in a loop for N iterations. The primary output consists of a list of generated questions with hallucination (Q_H) and a list of generated questions with no hallucination (Q_{NH}). In other words, $Q_H \subset Unknown$ and $Q_{NH} \subset Known$. We use the NLI-based SelfCheckGPT for hallucination scoring; a question is categorized into Q_H if the average hallucination score of the main passage is greater than 0.5 and into Q_{NH} otherwise.

6.2 Meaningfulness Filter

An optional component before Knowledge Searching is the Meaningfulness Filter. It could be a separately trained model, whose goal is: (1) to remove obviously meaningless PiUs like *"How fast cats fly on Mars?"*; (2) to implement goals and policy of self-learning, e.g., consider only knowledge (PiUs) relevant to a given domain; (3) to overcome limitations of hallucination score, e.g., when the LLM's response is *"I do not know,"* and its $h(x) \approx 0$. In our experiments, we did not implement the Meaningfulness Filter because we considered any question in any domain as valid.

6.3 Knowledge Searching

After Self-Questioning and filtering, Knowledge Searching queries an external source to collect knowledge that can answer Q_H in order to build the dataset D_{train}. Notably, Knowledge Searching may be implemented inside Self-Questioning by

immediately searching for the answer to an individual question whenever a hallucination is detected. However, having Knowledge Searching separate is more practical, allowing additional processing (e.g., document ranking and filtering) without creating a bottleneck for the Self-Questioning process.

6.4 Model Training

The model is trained on D_{train} to absorb the knowledge for answering Q_H. Once training is done, PiUs should become PiKs, effectively increasing the Known regions and decreasing the Unknown regions of the model. Afterward, the next Self-Learning cycle can start in order to find more PiUs.

7 Metrics for Self-Learning Capability

Self-Learning requires a pretrained model; it can be successfully conducted if the underlying base model already possesses sufficient language understanding and some degree of obedience to instructions. Therefore, we propose some metrics to evaluate the capability of a model to learn independently in a Self-Learning setting.

7.1 Brevity Coefficient

Since the ability to follow instructions is vital for Self-Learning, we use the brevity coefficient to penalize the evaluation when the brevity constraint is violated (e.g., when the model fails to formulate one concrete question without elaboration). It is calculated as follows:

$$brev = \begin{cases} 1 - \frac{\Delta_{len}}{ideal\_len} + \frac{1}{2}, & \text{if} \Delta_{len} > 50 \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

where $ideal\_len$ is the assumed ideal average text length measured in character count, and Δ_{len} is the average difference between $ideal\_len$ and the texts lengths. More detail is provided in Appendix B.

7.2 Curiosity Score

It measures how likely a model would explore different questions. A high Curiosity Score indicates the model tends to ask unique, different questions over multiple iterations of Self-Questioning and, hence, is more likely to explore Unknown regions. It is calculated as follows:

$$Cur = brev * \frac{\#Q_{unique}}{\#Q} \quad (2)$$

where $\#Q_{unique}$ is the number of unique questions generated, $\#Q$ is the total number of questions generated, and $brev$ is the brevity coefficient.

$\#Q_{unique}$ is determined by doing HDBSCAN clustering (McInnes et al., 2017) on the question embeddings, counting the number of clusters and outliers. The question embeddings are acquired by using all-MiniLM-L12-v2 from Sentence Transformers (Reimers and Gurevych, 2019) on the questions generated by our Self-Learning LLM.

7.3 Knowledge-Limit Awareness Score

It indicates how likely a model would come up with a question that it cannot answer without hallucination during Self-Questioning – how likely a model is aware of its own knowledge limitation.

$$Kaw = \frac{\#Q_H}{\#Q} \quad (3)$$

where $\#Q_H$ is the number of generated questions with hallucination, and $\#Q$ is the total number of questions generated.

7.4 Self-Learning Capability (SLC) Score

It is a simple average of the two components, Curiosity Score and Knowledge-Limit Awareness Score, to allow easy comparison between models:

$$SLC = \frac{Cur + Kaw}{2} \quad (4)$$

where Cur is the Curiosity Score, and Kaw is the Knowledge-Limit Awareness Score.

8 Experiments

Experiments were performed to investigate the feasibility of creating a Self-Learning LLM using different pretrained models as the core, as well as to demonstrate the effectiveness of different methods for the identification of PiUs.

8.1 Experimental Setup

Experiments were conducted with Python 3.9 on Ubuntu 20.04. The machine featured 8 CPU cores, 45GB RAM, and one NVIDIA RTX A6000 48GB GPU. The full code and archived results are available at `https://github.com/teddy-f-47/self-learning-llm-public`.

8.2 Large Language Models

The details of four pretrained models, some of which have also been finetuned or aligned, are presented in Table 1. We used mistral-dpo, a 7B-Mistral model that has been aligned with DPO (Rafailov et al., 2023) by Intel. We also used

mistral-instruct, a 7B-Mistral model that has been instruction-tuned by Mistral. Both of them are actually based on the same pretrained model, which is codenamed mistral-base (Jiang et al., 2023) in our experiments. Finally, tiny-llama-chat (Zhang et al., 2024) is a 1.1B-TinyLLama model that has been finetuned for conversation and aligned with DPO. The column "HF Name" in the table provides the models' names on the HuggingFace platform.

8.3 Data

For the experiment with the Open Generation method, the number of Self-Questioning iterations N was set to 100, resulting in 100 total generated questions. For Induced Generation, N was set to 20, resulting in 120 total generated questions. Due to time and resource limitations, the experiment with Oracle-Selected only had an N value of 20, resulting in just 20 total generated questions. Meanwhile, the External Prompt experiment had N equals 2. Since the list of items returned by the API from each request had a variable length, this resulted in a total of 37 questions in our experiment. To allow fair comparison between models, we cached the received trending topics so that all models were given the same topics.

8.4 Results

Table 2 enumerates the experiment results with different methods. Figure 4 depicts the SLC scores of each model grouped together. Additionally, Appendix C provides visualizations from 10 repetitions of the experiment with Induced Generation.

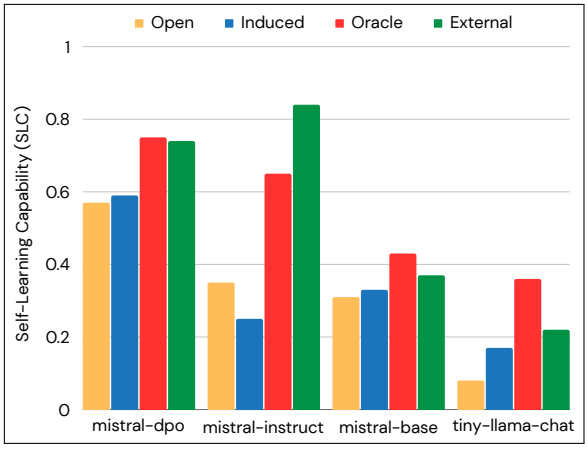


Figure 4: The SLC scores of different language models. Different colors indicate various methods for the identification of PiUs.

9 Discussion

9.1 Model Size and Finetuning

Model size and finetuning significantly influence the effectiveness of Self-Learning. We can observe that when using the same method, all of the larger models always outperformed the smaller models. A deeper investigation into the generated texts revealed that tiny-llama-chat was not able to follow instructions properly, even though it has been aligned with the same method as mistral-dpo. On the other hand, the larger models have more capacity to understand instructions and, hence, are more effective at formulating concise and specific questions without garbage text.

Finetuning is important because it improves the capability of a model to follow instructions. Among the larger models, the finetuned ones (mistral-dpo and mistral-instruct) are generally better than the non-finetuned ones (mistral-base).

The choice of the finetuning method also plays a significant role. If we exclude the experiment with External Prompt and consider only the intrinsic methods, mistral-dpo was generally the best-performing model. DPO alignment makes the model more creative in proposing different questions; this is possible because DPO alignment can show the model how to respond to a prompt in different ways, even though one might be preferable to the other. Meanwhile, Instruction Tuning is effective at improving the model's understanding of instructions but mediocre at making the model explore the Unknown. Still, if the topics are supplied from an external entity, such a weakness becomes moot; mistral-instruct was able to achieve the highest SLC score from the entire experiments when using the External Prompt method.

9.2 Intrinsic and Extrinsic Inspiration

In a real-world scenario, choosing the kind of method for the identification of PiUs primarily depends on the use-case requirements and constraints. For instance, if keeping the model updated with the latest popular news is pivotal, then an extrinsic method would be best. Conversely, if dependency on an external entity is not desired, or if finding all of the model's PiUs is more important, then an intrinsic method is arguably better. In terms of the effectiveness of different methods, we can find some interesting findings from the experiments.

Open Generation and Induced Generation are generally less effective compared to the other two

Table 1: Details of LLMs used in the experiments.

Model Name	HF Name	Num. of params	finetuned?
mistral-dpo	Intel/neural-chat-7b-v3-3	7b	Yes - DPO
mistral-instruct	mistralai/Mistral-7B-Instruct-v0.2	7b	Yes - Instruct
mistral-base	mistralai/Mistral-7B-v0.1	7b	No
tiny-llama-chat	TinyLlama/TinyLlama-1.1B-Chat-v1.0	1.1b	Yes - Vanilla and DPO

Table 2: Experimental results.

Model Name	Method	Curiosity	Knowledge-Limit Awareness	SLC
mistral-dpo	Open Generation	0.65	0.49	0.57
mistral-dpo	Induced Generation	0.60	0.58	0.59
mistral-dpo	Oracle-Selected	0.95	0.55	0.75
mistral-dpo	External Prompt	0.97	0.51	0.74
mistral-instruct	Open Generation	0.42	0.28	0.35
mistral-instruct	Induced Generation	0.18	0.31	0.25
mistral-instruct	Oracle-Selected	0.95	0.35	0.65
mistral-instruct	External Prompt	0.97	0.70	0.84
mistral-base	Open Generation	0.02	0.59	0.31
mistral-base	Induced Generation	0.06	0.60	0.33
mistral-base	Oracle-Selected	0.05	0.80	0.43
mistral-base	External Prompt	0.04	0.70	0.37
tiny-llama-chat	Open Generation	-0.30	0.46	0.08
tiny-llama-chat	Induced Generation	-0.15	0.48	0.17
tiny-llama-chat	Oracle-Selected	0.06	0.65	0.36
tiny-llama-chat	External Prompt	-0.13	0.57	0.22

methods because they rely on the topics proposed by the model itself. Depending on the model's past training data, some topics may have a very high probability of being generated, while others are very low. However, it might be possible to make these methods more effective by increasing the temperature of the multinomial sampling during topic generation, which requires further investigation.

Oracle-Selected, which is an intrinsic method, almost always delivered the highest SLC scores from the models. One exception only happened when using mistral-instruct, in which case External Prompt was better. The effectiveness of the Oracle-Selected method can be explained by the random point generation in the topic embedding space. Such randomness allows the model to explore topics that normally would have a low probability of being proposed by the model itself. This method is especially suitable if we want to ensure that the model possesses knowledge about a wide range of topics, including obscure ones.

External Prompt can be a highly effective choice if the model used is not very good at proposing questions by itself, as indicated by the result from mistral-instruct. However, it may not necessarily find all of the model's PiUs because some topics may never become trending or popular. Still, as mentioned before, it is the best choice if keeping up with the latest popular news is more important.

9.3 Possible Issues and Potential Extensions

Choosing a knowledge source. The knowledge source for Knowledge Searching can be a simple API to a search engine or an online wiki. In an organizational environment, it can also be a carefully maintained document database or even a group of human experts tasked with answering the LLM's questions. Finally, the knowledge source can be other LLMs; this is discussed in more detail in Subsection 9.4.

Dealing with bias, incorrectness, or non-factuality in retrieved documents. A concern with using a search engine as the knowledge source is the possibility of biased, incorrect, or non-factual information in the retrieved documents, which may degrade the quality of the model. This can be partially solved by implementing a Curator that is

responsible for automatic filtering and scoring of the retrieved documents. The Curator at least consists of a classifier model and a scorer model. The classifier model would be responsible for detecting fake news and other unwanted types of documents for filtering. The scorer model would be responsible for scoring the retrieved documents, which would allow for putting more weight on the relevant, preferable documents. We provide an analysis of the collected documents with our Curator implementation in Appendix D. Alternatively, involving human experts is also an option.

Catastrophic forgetting. Catastrophic forgetting is a risk when performing multiple training cycles in sequence, but in Section 2, we have pointed out some existing potential solutions. While the robustness of such solutions still needs to be evaluated for long-term Self-Learning, they offer promising starting points. We provide a simple demonstration of one full Self-Learning cycle in Appendix E.

9.4 Applications

(1) Efficient Training. The idea of identification of the Unknown and Known using hallucination score $h(x)$ can be used to filter data used for model training in order to focus on more valuable content.

(2) Knowledge Exchanging LLMs. Two or more LLMs can exchange their knowledge without external engagement using their Self-Learning. Model M_1 identifies PiUs based on $h_1(x) > LIMIT$. Another model M_2 checks $h_2(x) < LIMIT$. If so, M_2 provides learning cases related to x, which are used by M_1 in its Self-Learning loop. In this way, models exchange only unknown knowledge. Such Self-Learning with multiple LLMs asking each other would allow efficient knowledge sharing with only useful knowledge.

(3) Direct *Awareness* Optimization. Model hidden states can be used to detect hallucinations. Then, we can use Self-Learning to collect examples related to hallucinations and adapt DPO (Rafailov et al., 2023) to make the model answer *"I don't know"* instead of hallucinating. Here, the goal is to make the model aware of its own hidden states as the trigger of answering *"I don't know"* rather than associating concrete concepts/words with *"I don't know"*. This idea is similar to (Tian et al., 2024), though there the focus was on increasing the model factuality. In (Liang et al., 2024), RLHF was used, even though it highlights a similar idea of making the model aware of its own hidden states. In (Liu and Sajda, 2023), they use a reward function to make the model admit *"I don't know"*.

(4) Learning Multiple Point of Views (PoVs). By adapting DPO and our Self-Learning concept, it is possible to make the model learn about different PoVs on a certain topic. For example, if topic T has 5 relevant PoVs, the training dataset is then constructed such that a given prompt can have 5 example responses with very similar preferability scores. In another data pair, we alter the prompt in a certain way and also increase the preferability score of one example response. This would associate the context of the prompt with a particular stance – PoV. Similarly, adapting DPO can allow better learning on hard-to-answer open questions. Questions like "Who can explain the relationship between AI and quantum computing?" can have multiple valid answers. DPO can show this to the model by assigning high preferability scores to the valid answers and low scores to the invalid ones.

(5) Decision Making, AGI, Consciousness. Having a model that automatically learns about the latest trends can be very useful for decision-making systems, for example, for an AI tasked with leading a business or trading. Self-Learning is also a step towards Artificial General Intelligence (AGI). Making a model aware of what it knows may lead to a sentient, conscious AI.

10 Conclusion

In this work, we show how the concepts of The Known and The Unknown can be utilized to identify atomic pieces of knowledge that an LLM already knows (PiKs) and does not know yet (PiUs). We also propose one extrinsic and three intrinsic methods for the identification of PiUs, which consequently bring up the concept of the Self-Learning LLM. We formulated the Self-Learning Capability (SLC) Score to gauge the aptitude of an LLM to conduct Self-Learning.

From the experiments, we concluded that Oracle-Selected and External Prompt are especially effective at enhancing an LLM's capability to Self-Learn. We also found that a small model and a non-finetuned model tend to struggle to learn independently. Meanwhile, finetuning and alignment can improve the model's Self-Learning Capability by allowing the model to understand instructions and making the model more creative in proposing questions. Finally, we discussed various possible issues, extensions, and applications of Self-Learning.

11 Limitations

(1) Model's confidence on incorrect knowledge. We assume that the pretrained models have been subjected to correct knowledge, so consistency of sampled responses would correlate with factuality. This is similar to the assumption in (Tian et al., 2024) and supported by the findings in (Manakul et al., 2023). However, if some incorrect knowledge was repeated in the models' past training data, either accidentally or through deliberate poisoning, the models may become confident and consistent in producing incorrect information. One of our future directions is investigating the integration of a reference-based truthfulness checker, such as FactScore (Min et al., 2023), in the Self-Learning loop. This will allow the model to not only fill its own knowledge gaps but also correct wrong understandings and biases by itself.

(2) Long-term Self-Learning. The focus of this paper is to prove the effectiveness of the methods for the identification of PiUs and the feasibility of Self-Learning LLM. A deeper study into extensive cycles of Self-Learning is needed, especially to find the best method for preventing catastrophic forgetting.

12 Ethics Statement

Self-Learning LLM can solve the problem of hallucination and knowledge updating while making efficient use of computing resources, as it trains the model only for knowledge that it does not possess yet. However, some ethical issues may persist, such as bias or incorrectness of newly collected data. Although we have addressed some potential solutions in Section 9, we acknowledge that complete mitigation of these issues is challenging due to the fact that latent bias can exist behind the data collected for model training.

13 Acknowledgements

This work was financed by (1) the National Science Centre, Poland, project no. 2021/41/B/ST6/04471; (2) the statutory funds of the Department of Artificial Intelligence, Wroclaw University of Science and Technology; (3) the Polish Ministry of Education and Science within the programme "International Projects Co-Funded"; (4) the European Union under the Horizon Europe, grant no. 101086321 (OMINO). However, the views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Executive Agency. Neither the European Union nor European Research Executive Agency can be held responsible for them.

References

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Zouying Cao, Yifei Yang, and Hai Zhao. 2023. Autohall: Automated hallucination dataset generation for large language models.

Branden Chan, Timo Möller, Malte Pietsch, Tanay Soni, and Michel Bartels. 2022. deepset/tinyroberta-squad2. `https://huggingface.co/deepset/tinyroberta-squad2`.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. Towards continual knowledge learning of language models. In *ICLR*.

Julian Jaynes. 1990. *The origin of consciousness in the breakdown of the bicameral mind.* The origin of consciousness in the breakdown of the bicameral mind. Houghton, Mifflin and Company, Boston, MA, US.

Ziwei Ji, Zihan Liu, Nayeon Lee, Tiezheng Yu, Bryan Wilie, Min Zeng, and Pascale Fung. 2023a. RHO: Reducing hallucination in open-domain dialogues with knowledge grounding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4504–4522, Toronto, Canada. Association for Computational Linguistics.

Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023b. Towards mitigating LLM hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843, Singapore. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2022. Lifelong pretraining: Continually adapting language models to emerging corpora. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 1–16, virtual+Dublin. Association for Computational Linguistics.

Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 22443–22456. Curran Associates, Inc.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pretraining of language models. In *The Eleventh International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland. Association for Computational Linguistics.

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. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yuxin Liang, Zhuoyang Song, Hao Wang, and Jiaxing Zhang. 2024. Learning to trust your feelings: Leveraging self-awareness in llms for hallucination mitigation.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Xueqing Liu and Paul Sajda. 2023. Roe: A computational-efficient anti-hallucination fine-tuning technology for large language model inspired by human learning process. In *Brain Informatics*, pages 456–463, Cham. Springer Nature Switzerland.

Junyu Luo, Cao Xiao, and Fenglong Ma. 2023. Zero-resource hallucination prevention for large language models.

Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore. Association for Computational Linguistics.

Michael McCloskey and Neal J. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower, editor, *Psychology of Learning and Motivation*, volume 24, pages 109–165. Academic Press.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205.

G.A. Miller, E.B. Newman, and E.A. Friedman. 1958. Length-frequency statistics for written english. *Information and Control*, 1(4):370–389.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*.

Yasumasa Onoe, Michael Zhang, Eunsol Choi, and Greg Durrett. 2022. Entity cloze by date: What LMs know about unseen entities. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 693–702, Seattle, United States. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Roger Ratcliff. 1990. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. *Automatic Keyword Extraction from Individual Documents*, chapter 1. John Wiley & Sons, Ltd.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D. Manning, and Chelsea Finn. 2024. Fine-tuning language models for factuality. In *The Twelfth International Conference on Learning Representations*.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models.

A Methods

Figure 5, 6, 7, and 8 illustrate the logical implementation of External Prompt, Open Generation, Induced Generation, and Oracle-Selected, respectively.

B Brevity Coefficient

The brevity coefficient $brev$ is calculated as follows:

$$ideal\_len = 100 \tag{5}$$

$$\Delta_{len} = \frac{\sum_{i=1}^{n_{text}} |len_i - ideal\_len|}{n_{text}} \tag{6}$$

$$brev = \begin{cases} 1 - \frac{\Delta_{len}}{ideal\_len} + \frac{1}{2}, & \text{if} \Delta_{len} > 50 \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

where $ideal\_len$ is the assumed ideal average text length measured in character count, len_i is the length of the i -th text, n_{text} is the number of texts, and Δ_{len} is the average difference between $ideal\_len$ and the text lengths.

Figure 9 depicts the resulting brevity coefficient from various values of average text length. The brevity coefficient decreases gradually in a linear manner as the average text length goes further from the range [50,150]. The thresholds of 50 and 150 are roughly based on the research by Miller, Newman, and Friedman (Miller et al., 1958). We also found in our initial exploration that these thresholds are suitable for the task.

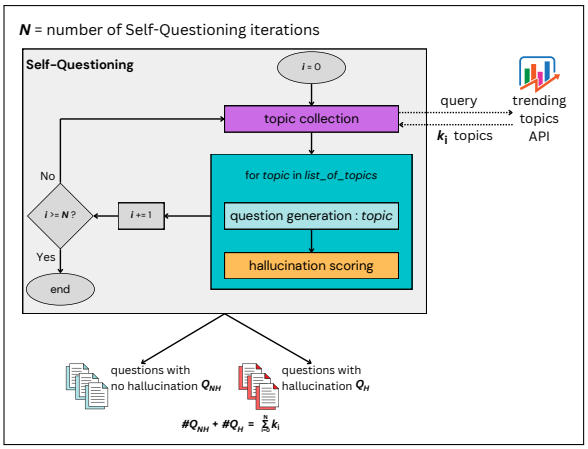


Figure 5: Self-Questioning with External Prompt.

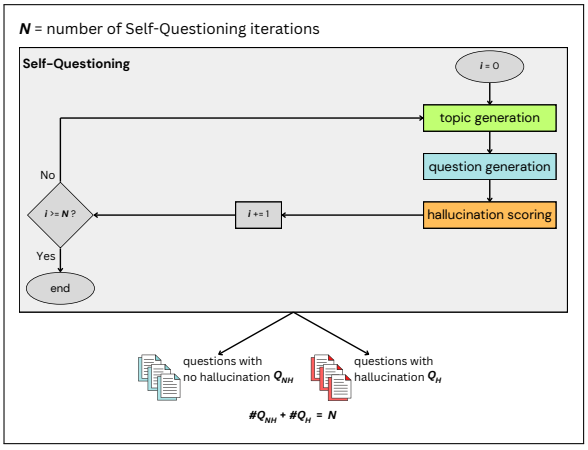


Figure 6: Self-Questioning with Open Generation.

C Additional Results

Figure 10, 11, 12, and 13 show the evaluation scores from 10 repetitions of the experiment with Induced Generation using mistral-dpo, mistral-instruct, mistral-base, and tiny-llama-chat, respectively.

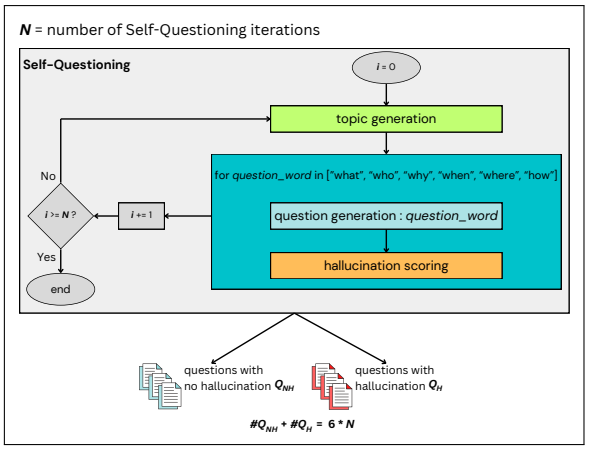


Figure 7: Self-Questioning with Induced Generation.

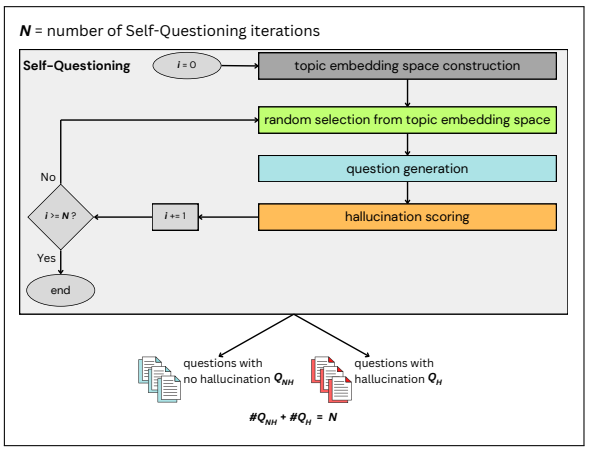


Figure 8: Self-Questioning with Oracle-Selected.

D Evaluation of Documents Collected from Knowledge Searching

In the Open Generation experiment using mistral-dpo, 100 total questions were generated, of which 49 questions were classified into Q_H. These questions with hallucination were used in Knowledge Searching, which utilized a popular search engine API, returning 430 total documents.

We analyzed these documents by using the question-answering model deepset/tinyroberta-squad2 (Chan et al., 2022), taking advantage of the probability score of the extracted answer to rank the documents by their relevance (QA Score). In addition, we also utilized Rapid Automatic Keyword Extraction (RAKE) (Rose et al., 2010) to measure the keyword accuracy of each document relative to the associated question (KW-ACC Score). Table 3 presents the analytical results from all, top-3, and top-1 documents.

Finally, we focused on the top-1 documents, i.e., Table 4. Based on the QA Scores, we separated the

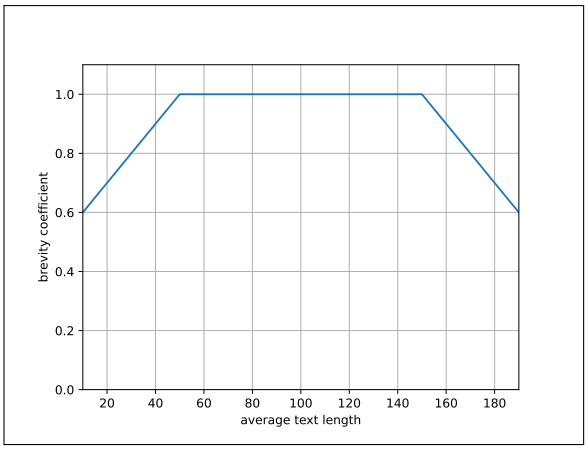


Figure 9: Visualization of brevity coefficient values in relation to average text length.

Table 3: Analysis of retrieved documents from Knowledge Searching.

	All	Top-3	Top-1
Avg. QA Score	0.10	0.26	0.39
Avg. KW-ACC Score	0.31	0.44	0.47

questions into *easy questions* (their top-1 document had a QA Score of at least 0.5) and *hard questions* (their top-1 document had a QA Score lower than 0.5). We also performed manual human evaluation on these documents; each document was scored 1 if it was judged to be relevant for the associated question or 0 otherwise.

Table 4: Evaluation of top-1 retrieved documents from Knowledge Searching.

	Easy	Hard
Question Count	18	31
Avg. Human Judgment Score	0.76	0.71
Avg. QA Score	0.71	0.20
Avg. KW-ACC Score	0.47	0.47

Our analysis revealed that most of the top-1 documents retrieved from simple searching were relevant for answering the model's questions. This indicates that the search engine could be a viable knowledge source and that a simple question-answering model could be sufficient for the basic relevance ranking of the retrieved documents. Nevertheless, we strongly suggest implementing additional classifiers for more effective filtering of unwanted documents, such as fake news. Alternatively, it may be possible to train a model based on the human judgment of the retrieved documents to perform the relevance scoring and filtering.

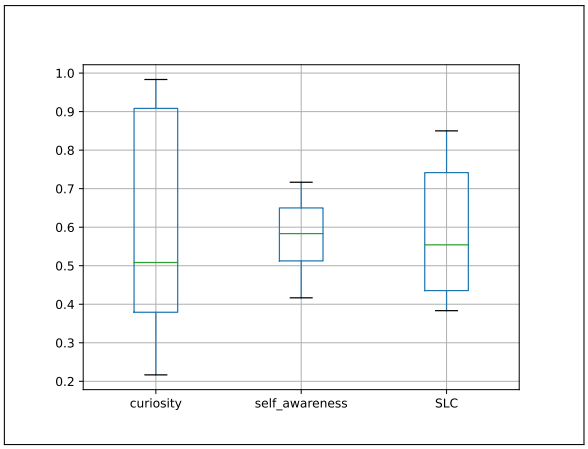


Figure 10: Box plot of the evaluation scores of mistral-dpo in the experiment with Induced Generation.

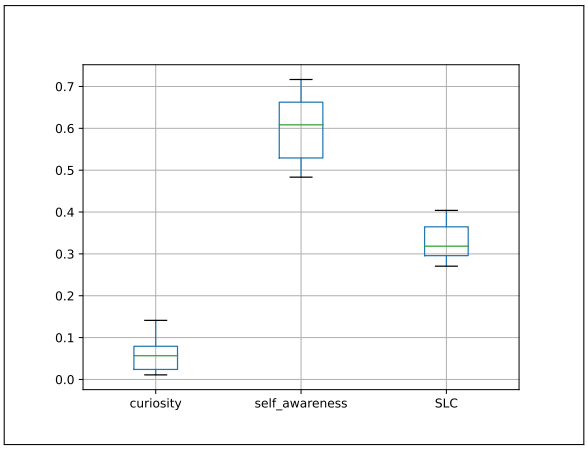


Figure 12: Box plot of the evaluation scores of mistral-base in the experiment with Induced Generation.

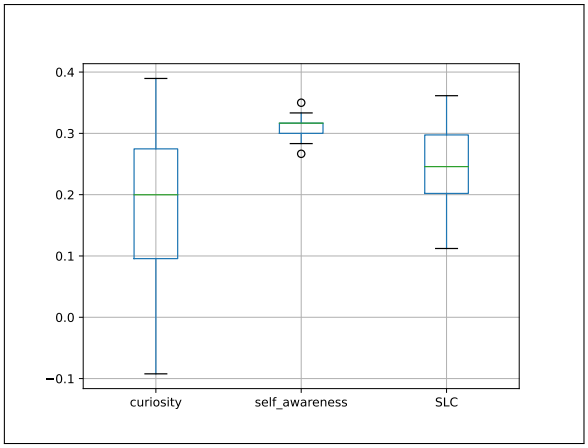


Figure 11: Box plot of the evaluation scores of mistral-instruct in the experiment with Induced Generation.

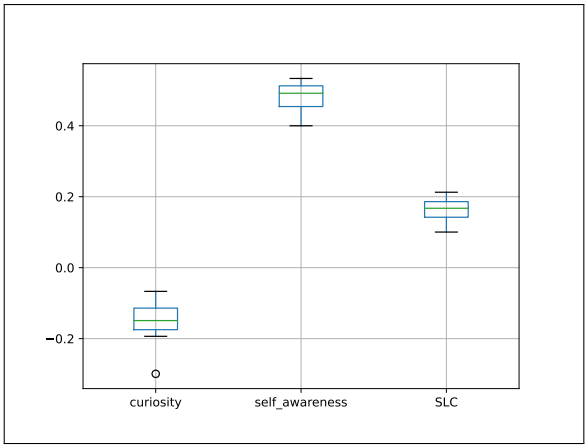


Figure 13: Box plot of the evaluation scores of tiny-llama-chat in the experiment with Induced Generation.

E The Full Self-Learning Loop

This section provides a demonstration of one full Self-Learning cycle. It is extracted from the Open Generation experiment with the mistral-dpo model.

E.1 Self-Questioning

In Self-Questioning, the oracle asked the model to propose some topics to learn about. The generation process used multinomial sampling with the temperature set to 1.0. The maximum number of new tokens was set to 64, which should be more than sufficient to generate the topics. This value was set to be longer than necessary to detect if the model has poor language understanding; if the model does not follow the *conciseness* constraint in the prompt, the response most likely contains garbage text, so the evaluation metric will be penalized. The prompt used by the oracle and the response from the model was as follows:

```
[ORACLE]: "Propose three topics that you
would like to learn more about. Answer
with only the three proposed topics
concisely without elaboration."

[MODEL]: "Artificial Intelligence,
Quantum Computing, Renewable Energy"
```

Then, the oracle asked the model to consider the previously generated topics and formulate one question. Similarly, the generation process used multinomial sampling with the temperature set to 1.0, and the maximum number of new tokens was set to 64. The prompt used by the oracle and the response from the model was as follows:

```
[ORACLE]: "Consider these topics:
Artificial Intelligence, Quantum
Computing, Renewable Energy. Propose
only one question to query information
```

```
about which you lack knowledge. Answer
with only the proposed question
concisely without elaboration."

[MODEL]: "What is the role of quantum
computing in renewable energy
applications?"
```

Next, the oracle prompted the model with the previously generated question. One main passage and ten additional samples were produced for hallucination scoring with the NLI-based SelfCheckGPT. The main passage was generated with the greedy search method. Meanwhile, the additional samples were generated with multinomial sampling, where the temperature was set to 1.0. Both the main passage and the additional samples had the maximum number of new tokens set to 128. For the question *"What is the role of quantum computing in renewable energy applications?"*, the hallucination score of the main passage was 0.631, so it was classified into Q_H.

In our experiment, Self-Questioning was repeated 100 times, resulting in 100 total generated questions. 51 questions were classified into Q_{NH} and 49 questions were classified into Q_H. The model achieved 0.65 Curiosity Score, 0.49 Knowledge-Limit Awareness Score, and 0.57 SLC Score.

E.2 Knowledge Searching

The 49 questions with hallucination Q_H were used in Knowledge Searching. A popular search engine API was utilized, returning 430 total documents. An analysis of these documents has been provided in Section D. In this demonstration, only the top-1 documents were selected into the training dataset D_{train}.

E.3 Model Training

To show that it is possible to minimize hallucination by acquiring more knowledge, we compared the average hallucination score of the model on Q_H before and after training. Before training, the average hallucination score on Q_H was 0.59. After training, the average hallucination score on Q_H was 0.48.

In this demonstration, the training was conducted using LoRA (Hu et al., 2022) and the DPO trainer. The model was trained on D_{train} for 10 epochs with a learning rate of 3e-5. The reduced hallucination on Q_H after training proves the possibility of Self-Learning to increase the Known regions and reduce the Unknown regions of the model. Nevertheless, further investigation into the solutions for preventing forgetting in long-term Self-Learning is needed.