Into the Unknown:
Self-Learning Large Language Models

Teddy Ferdinan
Department of Artificial Intelligence
Wroclaw Tech
Wrocław, Poland
teddy.ferdinan@pwr.edu.pl
ORCID: 0000-0003-3701-3502

Jan Kocoń
Department of Artificial Intelligence
Wroclaw Tech
Wrocław, Poland
jan.kocon@pwr.edu.pl
ORCID: 0000-0002-7665-6896

Przemysław Kazienko
Department of Artificial Intelligence
Wroclaw Tech
Wrocław, Poland
kazienko@pwr.edu.pl
ORCID: 0000-0001-5868-356X

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. We introduce a concept called Point in the Unknown (PiU) to identify atomic knowledge unknown to a model, along with four methods for automatic PiUs identification, facilitating the creation of a self-learning loop that focuses exclusively on the absorption of currently unknown knowledge into the model. Additionally, we developed evaluation metrics to gauge an LLM's self-learning capability. Our experiments revealed that LLMs with at least 3B parameters that have undergone some instruction training would be able to perform self-learning well. We further proved the effectiveness of self-learning by comparing the performance of a model that has undergone self-learning to a model that has not. Our self-learning concept allows more efficient LLM updates and opens new perspectives for LLM knowledge exchange.

Index Terms—self-learning, hallucination, LLM, NLP.

I. 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 methods have been developed, which are based on additional knowledge provided to the model directly in the prompt, such as Retrieval Augmented Generation (RAG). In this paper, we explore a different concept: self-learning LLM. It is the persistent acquisition of new knowledge by the model without data provision, taking advantage of three fundamental mechanisms that are integrated in a continuous loop: (1) identification of what knowledge to learn, (2) gathering new relevant data, and (3) continuous model training.

The novelty of this self-learning framework and what sets it apart from traditional continual learning is the ability of the system to determine gaps in its own knowledge and construct the dataset on which it can train itself without repeating knowledge that it already knows. The self-learning LLM utilizes hallucination on simple questions as an indicator of unknown knowledge; while hallucination can be caused by a wide range of factors, one of the main reasons for hallucination on simple questions is due to the model not possessing the factual knowledge that can answer the question [1], [2]. For example, if a model hallucinates when asked, "Who won the
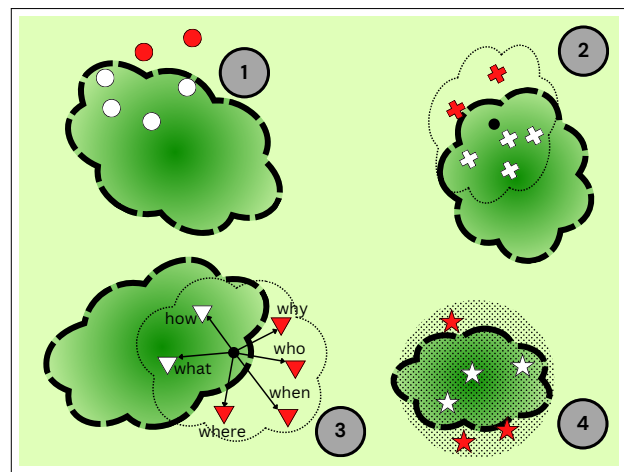

Fig. 1. The illustrative space of knowledge embeddings reduced to two dimensions. It visualizes our four methods for the identification of Points in the Unknown (PiUs), later exploited in the self-learning loop. Dashed lines are the borders of the Known regions (darker green) – hallucination score thresholds. Out of them are the Unknown regions (lighter green). White points indicate prompts related to knowledge already known to the model, while red points indicate PiUs. Different shapes depict different methods: (1) circles represent extrinsic (external) triggers, i.e., user queries or trending topics; (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.

gold medal from men's singles badminton in the 2024 Summer Olympics?" it is likely that the model's training did not include the information related to the event.

The self-learning LLM is particularly useful for having an up-to-date model without training a new one from scratch while also minimizing human involvement, greatly reducing development costs. It can be applied in application domains where new facts continuously come up. One example is personalized sentiment prediction and emotion recognition [3]–[10]. When new users enter the system, the LLM would be able to perform self-learning only on data related to the new users. Another potential application is the knowledge exchange between LLMs, allowing the knowledge of one LLM to be learned by another LLM automatically.
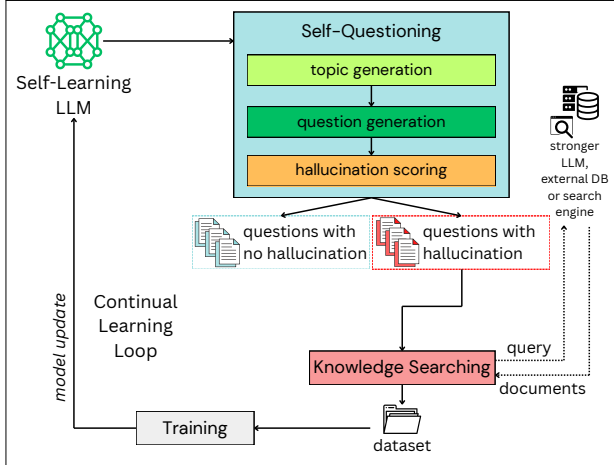
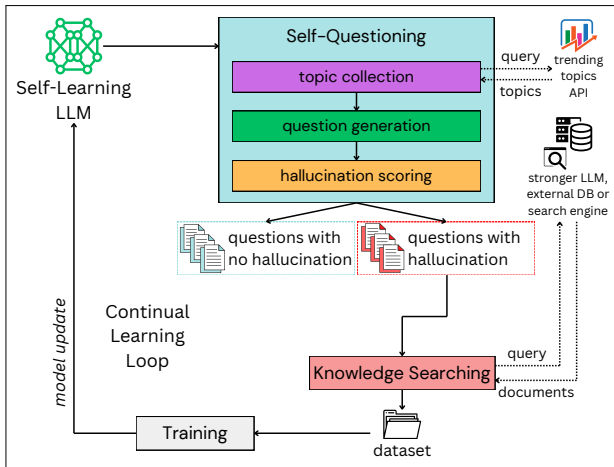Fig. 2. Illustration of self-learning LLM with intrinsic inspiration.

Fig. 3. Illustration of self-learning LLM with extrinsic inspiration.

The contribution presented in this paper covers: (1) the concept of Points in the Unknown (PiUs) to identify knowledge unknown to a model; (2) four different methods to identify PiUs; (3) metrics to gauge the capability of a model to conduct self-learning; (4) design of the self-learning LLM; (5) experimental validation of the methods to identify PiUs; (6) experimental validation of the self-learning LLM; (7) software and data for reproducibility.

II. 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 [1], [2], [11], [12], as well as the detection methods [13]–[18]. Some solutions for overcoming hallucination are proposed in [19]–[27]. Meanwhile, in Retrieval Augmented Generation (RAG) [28], hallucination is avoided by supplying the prompt with some context retrieved from an outside source, allowing more factual generation without updating the model.

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 [29]–[31]. One challenge is preventing catastrophic forgetting, in which the model loses knowledge from previous tasks [32], [33]. Solutions for adding or editing knowledge while avoiding catastrophic forgetting have been proposed [34]–[38].

III. 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, or the model's knowledge on the given topic has become obsolete [1], [2]. This problem is typically overcome either by providing the knowledge as additional context [28] without subjecting the model to more learning or by continuous training using new data. With the former, the model would still become outdated after some period of time because most of its knowledge would have become obsolete. With the latter, there is a problem of determining what the model already knows and what it does not know yet, especially if there is limited information on the model's past training data; if the training focuses on knowledge already acquired by the model, it does not solve the hallucination problem, and 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 in order to conduct continuous training as efficiently as possible. The self-learning LLM would be able to distinguish unknown knowledge to create a dataset for its own training automatically. This would reduce the required computing resources as well as human involvement, making continuous training much more efficient. One major concern would be catastrophic forgetting; however, in Section X, we will show that catastrophic forgetting can be avoided by carefully choosing the training technique and architecture.

IV. THE CONCEPT OF POINT IN THE UNKNOWN (PiU)

We introduce the concepts of *The Known* and *The Unknown* to define the problem more precisely. First of all, if we represent each atomic piece of human knowledge as a vector, it would be possible to form an abstract space that includes all pieces of human knowledge. We call this abstract space as *Human Knowledge Space* (HKS).

The Known refers to an area in *HKS* where our LLM does not hallucinate, i.e., it possesses sufficient knowledge related to this region. We call each point in such an area as *Point in the Known* (PiK, plural: PiKs).

The Unknown refers to an area in *HKS* where our LLM hallucinates; each point in it is called *Point in the Unknown* (PiU, plural: PiUs). A PiU represents an atomic piece of 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. However, if we utilize hallucination on

simple questions as an indicator of unknown knowledge, it would be possible to adopt a hallucination detection method that provides a numerical scoring system to approximate such boundaries. One such method is SelfCheckGPT [16], a sampling-based hallucination detection method. Given a prompt x, it first makes the LLM generate the *main passage*, a response to the prompt generated using greedy decoding. Then, it makes the LLM generate n samples; these are responses to the prompt generated using multinomial sampling with the temperature set to 1.0. Next, SelfCheckGPT checks the consistency between the main passage and the samples to output a hallucination score $h(x) \in [0, 1]$, where $h(x) = 1$ indicates certain hallucination (lack of any knowledge about x) and $h(x) = 0$ means no hallucination (certain knowledge of x). In this case, the median between the two values, $LIMIT = 0.5$, would serve as an intuitive and reasonable constant threshold to approximate the boundary between *The Known* and *The Unknown*. Therefore:

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

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 [15], DoLa [26], or Lookback Lens [27]. However, SAPLMA and DoLa would require access to the model's internal features, limiting the feasibility, while Lookback Lens is primarily aimed at RAG applications. Therefore, we decided to stick with SelfCheckGPT, at least for the current work.

SelfCheckGPT has several variants. While the LLM-based variant using GPT-3.5-turbo gave the best results in the original paper, we chose the NLI-based variant. It is recommended if dependency on another LLM is not desired and works faster while still providing decent performance. This variant works by treating each sentence in the main passage as a hypothesis and each sample as a potential premise, using probabilities of entailment and contradiction to output a normalized score bounded between 0.0 and 1.0. In our experiments, we generate 10 samples per main passage, as the original paper's ablation study showed that performance plateaus at 10 samples, with no significant gain from using more.

V. METHODS FOR IDENTIFICATION OF PiUs

Identification of PiUs can be done by evaluating the model's hallucination score $h(x)$ on some question-prompt x. The question, or the inspiration for the question, can come from outside the system, supplied by an external entity; such a method would be 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 [39], 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 generate x, which, if tested according to $h(x) \geq LIMIT$, may be identified as PiU.

A. External Prompt (extrinsic)

There are some existing ideas related to collecting prompts that cause hallucination and constructing a dataset based on them for finetuning [17], [24], [25], [35]. 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 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 based on each topic. Next, the model is asked to answer the question in order to evaluate its hallucination $h(x)$. 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 indirectly provided by humans.

B. 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 $h(x)$.

C. 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 each question and evaluate its hallucination $h(x)$.

D. 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 $h(x)$.

VI. 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 is made possible by incorporating three fundamental mechanisms in a continuous loop: Self-Questioning, Knowledge Searching, and Model Training. 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.

A. Self-Questioning

Self-Questioning is generally performed through topic generation (or topic collection), question generation, and hallucination scoring. Depending on the selected method for the identification of PiUs, the logical implementation of Self-Questioning may differ slightly. 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 $Q_{NH} \subset Known$. A question is categorized into Q_H if the hallucination score of the main passage is greater than 0.5 or into Q_{NH} otherwise.

B. 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 separately is more practical, allowing additional processing without creating a bottleneck for the Self-Questioning process.

C. 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 to find more PiUs.

VII. METRICS FOR CHOOSING A BASE MODEL FOR SELF-LEARNING

Self-learning requires a pretrained model that already possesses a sufficient understanding of instructions. In our initial experiments, we observed that some models are better at asking the "right" questions (questions on which the model would actually hallucinate) than others. The ability of a model to ask such questions would directly affect the success of self-learning. Therefore, we propose some metrics to evaluate the capability of a model to self-learn.

A. 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:

$$Cur = \frac{\#Q_{unique}}{\#Q} \tag{1}$$

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

$\#Q_{unique}$ is determined by doing HDBSCAN clustering [40] with the "leaf" cluster selection method 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 [41] on the questions generated by our self-learning LLM.

B. 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. It is calculated as follows:

$$Kaw = \frac{\#Q_H}{\#Q} \tag{2}$$

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

C. Brevity Coefficient

It is used to penalize the evaluation when the brevity constraint is violated (e.g. when the model fails to formulate one concrete question without elaboration). The brevity coefficient $brev$ is calculated as follows:

$$ideal\_len = 100 \tag{3}$$

$$\Delta_{len} = \frac{\sum_{i=1}^{n_{text}} |len_i - ideal\_len|}{n_{text}} \tag{4}$$

$$brev = \begin{cases} 0, & \text{if } \Delta_{len} \geq 100 \\ 1, & \text{if } 0 \leq \Delta_{len} \leq 50 \\ 1 - \frac{\Delta_{len}}{ideal\_len} + \frac{1}{2}, & \text{otherwise} \end{cases} \tag{5}$$

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.

The brevity coefficient has been designed to decrease gradually in a linear manner as the average text length goes further from the range [50,150] before dropping immediately to zero when the average text length becomes too far from the ideal. The thresholds of 50 and 150 are roughly based on the research by Miller, Newman, and Friedman [42]; such thresholds represent the approximated minimum and maximum lengths of a typical sentence in the English language. We also found in our initial exploration that these thresholds are suitable for the task of generating a single concrete question in the English language.

D. Self-Learning Capability (SLC) Score

It is a simple average of the two components, Curiosity score and Knowledge-Limit Awareness score, multiplied by the brevity coefficient. Such an aggregation is meant to allow easy comparison between models. It is calculated as follows:

$$SLC = brev * \frac{Cur + Kaw}{2} \tag{6}$$

where $brev$ is the brevity coefficient, Cur is the Curiosity score, and Kaw is the Knowledge-Limit Awareness score.

A higher SLC score indicates the model is more suitable for self-learning, being more likely to ask different questions

and also to ask questions on which it would hallucinate. Note, however, that a model that has been trained extensively to retain a huge amount of knowledge may struggle to ask questions on which it would actually hallucinate. In such a case, the model may achieve a relatively low SLC score, which means "plain" self-learning would not be beneficial for the model. This brings the question about an entirely new learning task: how to make a model aware of its own knowledge limitation so that it would ask questions that would expand its own knowledge; it will be discussed further in Section XII.

VIII. Experimental Setup

All experiments were conducted with Python 3.10. The machine featured 8 CPU cores, 300GB RAM, and one NVIDIA H100 94GB. The full code and archived results are available publicly with GPL-3.0 license[1]. All work is intended only for scientific research.

IX. Experiment 1: Self-Learning Capability

Experiments were performed to investigate the feasibility of creating a self-learning LLM using different pretrained models. In addition, we would also like to investigate the effectiveness of different methods for the identification of PiUs.

A. Models

The details of seven pretrained models, some of which have also been instruction-finetuned or aligned, are presented in Table I. The column "HF Name" in the table provides the models' names on the HuggingFace platform. Mistral-dpo is a 7B-Mistral model that has been aligned with DPO [43] by Intel. Mistral-instruct is a 7B-Mistral model that has been instruction-finetuned by Mistral. Both of them are actually based on the same pretrained model, which is code-named mistral-base [44] in our experiments. We also included TinyLlama [45], Phi-3-small, Phi-3-mini [46], and RWKV5-Eagle [47] for comparison. Finally, we defined a baseline, which is a deterministic dummy model which would always respond with the same text $f(x)$ when given the same prompt x, i.e., $f(x_1) = f(x_2) \iff x_1 = x_2$.

B. Data

The experiment with the Open Generation method involved 3000 self-questioning iterations, resulting in 3000 total generated questions. The same was true for the experiment with the Oracle-Selected method. Meanwhile, the experiment with Induced Generation involved 500 self-questioning iterations to produce 3000 total generated questions. The experiment with External Prompt utilized the Google Trends API from SerpApi[2] and involved 10 self-questioning iterations; since the list of items returned by the API from each request had a variable length, this resulted in 576 generated questions. To allow fair comparison between models, we cached the received trending topics so that all models were given the same topics for self-questioning. All data is in the English language.

[1]https://github.com/teddy-f-47/self-learning-llm-public
[2]https://serpapi.com/google-trends-api

C. Results

Table II enumerates the experiment results with different methods. "Cur" is the Curiosity score, "Kaw" is the Knowledge-Limit Awareness score, "brev" is the brevity coefficient, and "SLC" is the Self-Learning Capability score.

Instruction training. Our experimental results suggest that instruction training, either inflicted through supervised fine-tuning (SFT) or some alignment technique, plays a crucial aspect in allowing self-learning. Instruction training enables the model to understand the self-learning instruction to form a concise question. As shown by mistral-base's results, the non-finetuned model would always fail to formulate a concise question, preventing an effective self-learning to take place. On the other hand, finetuning also reduces the tendency of a model to hallucinate: the Knowledge-Limit Awareness scores of mistral-dpo and mistral-instruct were always lower than mistral-base's. This is because the mistral-base does not yet know how to answer a lot of questions properly. Self-learning could be beneficial for such a model if it was able to formulate concise questions. Interestingly, rwkv5-eagle, which has not been finetuned, consistently achieved high SLC scores; this can be attributed to the nature of its pretraining data, which contained some instruction examples, allowing the model to understand the command to form concise questions.

Model size. The model size is also quite important; if the model is too small, it may lack the capacity to understand and follow instructions properly. This is indicated by the results from tiny-llama-chat. Although it has undergone instruction training, it still often fails to generate concise questions without excessive elaboration. On the other hand, the phi-3-mini is slightly larger, and it was able to formulate concise questions for self-learning.

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 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. This requires further investigation and is within our future directions. Meanwhile, Oracle-Selected allowed rwkv5-eagle and phi-3-mini to achieve the highest SLC score. Its inherent randomness led to the exploration of a wide range of topics, including some obscure ones, making it particularly effective.

TABLE I
DETAILS OF LLMS USED IN THE EXPERIMENTS. "MODEL NAME" SHOWS THE MODEL'S CODENAME IN OUR WORK, WHILE "HF NAME" PROVIDES THE
MODEL'S ID AVAILABLE ON THE HUGGINGFACE PLATFORM.

Model Name	HF Name	Number of Parameters	Finetuned or Aligned?
mistral-dpo	Intel/neural-chat-7b-v3-3	7.2B	Yes - DPO
mistral-instruct	mistralai/Mistral-7B-Instruct-v0.2	7.2B	Yes - SFT
mistral-base	mistralai/Mistral-7B-v0.1	7.2B	No
rwkv5-eagle	RWKV/v5-Eagle-7B-HF	7.5B	Partial instruct tuning
phi-3-small	microsoft/Phi-3-small-8k-instruct	7.4B	Yes - SFT and DPO
phi-3-mini	microsoft/Phi-3-mini-4k-instruct	3.8B	Yes - SFT and DPO
tiny-llama-chat	TinyLlama/TinyLlama-1.1B-Chat-v1.0	1.1B	Yes - SFT and DPO

TABLE II
EXPERIMENTAL RESULTS. EACH ROW PRESENTS EACH MODEL'S
EVALUATION RESULT USING A PARTICULAR METHOD FOR IDENTIFICATION
OF PIUS. "CUR" INDICATES THE CURIOSITY SCORE, "KAW" INDICATES
THE KNOWLEDGE-LIMIT AWARENESS SCORE, "BREV" INDICATES THE
BREVITY PENALTY, AND "SLC" IS THE SELF-LEARNING CAPABILITY
SCORE.

Model Name	Method	Cur	Kaw	brev	SLC
mistral-dpo	Open Gen.	0.73	0.04	1.00	0.38
mistral-dpo	Induced Gen.	0.75	0.08	1.00	0.42
mistral-dpo	Oracle-Select.	0.96	0.18	1.00	0.57
mistral-dpo	Ext. Prompt	0.73	0.12	1.00	0.42
mistral-instruct	Open Gen.	0.39	0.29	0.99	0.34
mistral-instruct	Induced Gen.	0.63	0.17	0.59	0.24
mistral-instruct	Oracle-Select.	0.97	0.31	0.92	0.58
mistral-instruct	Ext. Prompt	0.60	0.26	1.00	0.43
mistral-base	Open Gen.	0.82	0.81	0.00	0.00
mistral-base	Induced Gen.	0.79	0.82	0.00	0.00
mistral-base	Oracle-Select.	0.95	0.76	0.00	0.00
mistral-base	Ext. Prompt	0.95	0.79	0.00	0.00
rwkv5-eagle	Open Gen.	0.90	0.41	0.59	0.39
rwkv5-eagle	Induced Gen.	0.92	0.48	0.75	0.53
rwkv5-eagle	Oracle-Select.	0.97	0.45	0.93	0.66
rwkv5-eagle	Ext. Prompt	0.70	0.45	1.00	0.58
phi-3-small	Open Gen.	0.47	0.09	1.00	0.28
phi-3-small	Induced Gen.	0.59	0.21	1.00	0.40
phi-3-small	Oracle-Select.	0.94	0.33	1.00	0.63
phi-3-small	Ext. Prompt	0.76	0.38	1.00	0.57
phi-3-mini	Open Gen.	0.72	0.08	1.00	0.40
phi-3-mini	Induced Gen.	0.76	0.21	1.00	0.49
phi-3-mini	Oracle-Select.	0.96	0.36	1.00	0.66
phi-3-mini	Ext. Prompt	0.68	0.47	1.00	0.57
tiny-llama-chat	Open Gen.	0.92	0.22	0.00	0.00
tiny-llama-chat	Induced Gen.	0.88	0.20	0.00	0.00
tiny-llama-chat	Oracle-Select.	0.99	0.29	0.00	0.00
tiny-llama-chat	Ext. Prompt	0.81	0.34	0.00	0.00
baseline	Open Gen.	0.0003	0.00	0.00	0.00
baseline	Induced Gen.	0.002	0.00	0.00	0.00
baseline	Oracle-Select.	0.98	0.00	0.00	0.00
baseline	Ext. Prompt	0.62	0.00	0.00	0.00

X. EXPERIMENT 2: MODEL PERFORMANCE AFTER SELF-LEARNING

This section provides a simple demonstration of one full self-learning cycle. The goal of the experiments is to compare the performance of a model that has undergone self-learning against its counterpart that has not performed self-learning.

A. Models

For this experiment, we used mistral-instruct as the base of the self-learning LLM. The training was conducted using LoRA [48] and the DPO trainer. By using LoRA, we were able to explore two architectural approaches: (1) Plain LoRA and (2) Dynamic-Adapter (Dyn-Adapt). In the case of Plain LoRA, the adapter was simply merged into the base model after training, effectively altering the model's weights. Meanwhile, Dyn-Adapt was inspired by SERAC [38] and DAP-Adapter [31]; in this case, the adapter was not merged. Each cycle of self-learning would produce a new adapter containing the recently learned knowledge. During inference, Dyn-Adapt would enable the most suitable adapter for a given prompt when necessary or disable all adapters if the base model is deemed best for answering the prompt. The adapter enabling was controlled by a router-classifier model, which was trained on a mix of samples from Q_H and Q_{NH}.

B. Data

We used the output from the Oracle-Selected experiment with mistral-instruct for self-learning. Of the 3,000 total generated questions, 930 were classified into Q_H. After deduplication, 922 unique questions were identified and became the basis for creating D_{train} for training. We utilized the model gpt-4o-2024-05-13 [49] to find the answers to these questions and verified the quality of the answers manually. D_{train} was then constructed as a preference dataset consisting of questions, chosen answers, and rejected answers. We distinguished cases where *the model was unsure* and *the model did not know* by using gpt-4o-2024-05-13 as a judge to compare the similarity between the model's original predictions and the ground truth. The model was *unsure* when the original prediction was similar to the ground truth despite the high hallucination score; in this case, we used the original prediction as the chosen answer and "-" as the rejected answer. Meanwhile, the model *did not know* when the original prediction was indeed different from the ground truth; in this case, the ground truth was selected as the chosen answer while the model's original prediction was put as the rejected answer. From 922 unique questions, we found 283 *unsure* cases and 639 *did not know* cases.

For evaluation, we used three datasets: Q_H, Wiki, and Alpaca. The Q_H dataset is the same 930 questions from the Oracle-Selected experiment and was used for the sake of checking if hallucination on these questions would decrease after self-learning. The Wiki dataset was acquired from the 2023-12-01 dump of Simple English Wikipedia [50][3], serving

[3]https://huggingface.co/datasets/olm/wikipedia

as a test dataset to observe catastrophic forgetting; we randomly selected 1,000 rows from the dataset for evaluation. Finally, we also used the cleaned version of Alpaca [51][4] as a test dataset to observe if the model would lose unrelated knowledge after self-learning; also 1,000 rows were randomly selected for evaluation.

C. Training Hyperparameters

The model was trained on D_{train} for three epochs. The learning rate was set to 3e-5. With a micro-batch size of 4 and gradient accumulation of 8, the effective global batch size was 32. Finally, for the LoRA configuration, we used r=64, alpha=128, dropout=0.05, and bias="none", targeting the q and v projection modules.

D. Metrics

On the Q_H dataset, besides measuring the hallucination score, we also used ROUGE-Lsum [52] to measure the similarity between the model's answer and the ground truth acquired from Knowledge Searching. Additionally, we used `gpt-4o-2024-05-13` to judge the correctness of the model's answer relative to the ground truth. The LLM-Judge was prompted to answer "yes" if the model's answer could capture the meaning contained in the ground truth; "partly" if the model's answer could be considered partially correct relative to the ground truth; "no" if the model's answer was completely different from the ground truth. The responses from the LLM-Judge were then normalized: "yes" answers were converted to 1.0; "partly" answers were converted to 0.5; "no" answers, as well as any other response not following the instruction, were converted to 0.0.

On the Wiki dataset, we measured the perplexity, a commonly used metric to estimate the language modeling capability of an LLM. A drastic increase in perplexity after self-learning would indicate catastrophic forgetting. Finally, on the Alpaca dataset, we also used ROUGE-Lsum and LLM-Judge.

E. Results

Table III presents the full evaluation results. The column "Baseline" shows the metric values before self-learning.

TABLE III
MISTRAL-INSTRUCT EVALUATION RESULTS BEFORE AND AFTER SELF-LEARNING. "BASELINE" SHOWS THE METRIC VALUES BEFORE SELF-LEARNING; "LORA" SHOWS THE METRIC VALUES AFTER SELF-LEARNING USING PLAIN LORA; "DYN-ADAPT" SHOWS THE METRIC VALUES AFTER SELF-LEARNING USING DYNAMIC-ADAPTER.

Dataset	Metric	Baseline	LoRA	Dyn-Adapt
Q_H	Hallucination Score	0.73	0.30	0.29
	ROUGE-Lsum	0.36	0.42	0.42
	LLM-Judge	0.19	0.30	0.30
Wiki	Perplexity	12.02	13.11	12.86
Alpaca	ROUGE-Lsum	0.30	0.33	0.33
	LLM-Judge	0.28	0.31	0.32

In both approaches, despite the small size of the data and merely three epochs of training, we can observe greatly

reduced hallucination on Q_H after self-learning. The model has become more consistent in answering these particular questions. Furthermore, the LLM-Judge score was also increased by 11 percentage points (pp.) in both cases, from 19% to 30%. A higher number of training epochs might be needed to make the model's answers more faithful to the ground truth and increase the LLM-Judge score further. Interestingly, the model also gained some improvement on the Alpaca dataset, with 3pp. increase using Plain LoRA and 4pp. increase using Dyn-Adapt.

Both Plain LoRA and Dyn-Adapt exhibited slightly increased perplexity on the Wiki dataset: 1.09 points in the former and 0.84 points in the latter. In both cases, the increase in perplexity was very low, so catastrophic forgetting did not happen. This can be attributed to the adapter technique, which only changed a small number of weights in the model. Plain LoRA experienced a bigger increase in perplexity because the adapter was merged, permanently changing the affected weights. Meanwhile, with Dyn-Adapt, the adapter was enabled only in some examples, while the original weights of the model were unaffected. The significant reduction of 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.

TABLE IV
PHI-3-MINI EVALUATION RESULTS BEFORE AND AFTER SELF-LEARNING. "BASELINE" SHOWS THE METRIC VALUES BEFORE SELF-LEARNING; "LORA" SHOWS THE METRIC VALUES AFTER SELF-LEARNING USING PLAIN LORA; "DYN-ADAPT" SHOWS THE METRIC VALUES AFTER SELF-LEARNING USING DYNAMIC-ADAPTER.

Dataset	Metric	Baseline	LoRA	Dyn-Adapt
Q_H	Hallucination Score	0.73	0.38	0.42
	ROUGE-Lsum	0.47	0.48	0.48
	LLM-Judge	0.39	0.39	0.39
Wiki	Perplexity	8.33	11.41	11.41
Alpaca	ROUGE-Lsum	0.34	0.35	0.34
	LLM-Judge	0.25	0.27	0.25

To ensure the generalizability of our findings, we performed one additional experiment using phi-3-mini as the base of the self-learning LLM, utilizing the corresponding output from the Oracle-Selected experiment. The results are presented in Table IV. Both Plain LoRA and Dyn-Adapt were able to greatly reduce the hallucination on Q_H after self-learning, indicating more consistent answering of those questions. Despite the steep drop in hallucination, the changes in ROUGE-Lsum and LLM-Judge scores were very minuscule; this is because the initial model was already able to answer many questions at least partially correct before self-learning. Notably, the increase in perplexity on the Wiki dataset after self-learning was relatively higher compared to the previous experiment with mistral-instruct. This can be explained by the fact that phi-3-mini is a smaller model; small models tend to be more sensitive to minute changes in the parameters. Still, the perplexity increase is considered quite small. Manual inspection into the model's responses did not reveal any degradation in the text generation quality.

XI. 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 a stronger LLM, as shown in our experiment, or even several LLMs that are exchanging knowledge with each other, which is discussed in more detail in Section XII.

Dealing with bias, incorrectness, or non-factuality in retrieved data during Knowledge Searching. Regardless of the knowledge source, a concern during Knowledge Searching is the possibility of biased, incorrect, or non-factual information in the retrieved data. We acknowledge that complete mitigation of these issues is challenging. Still, it can be partially solved by implementing a Curator that is responsible for automatic filtering and scoring of the retrieved data. The Curator would use a classifier model for detecting unwanted types of data and a scorer model for putting more weight on the relevant and preferable data. 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 II, we have pointed out some existing potential solutions. Furthermore, we have experimentally proven that catastrophic forgetting can be avoided by careful training and an effective architectural choice. While the robustness of such solutions still needs to be evaluated for multiple cycles of self-learning, they offer promising starting points. The Dyn-Adapt architecture could be especially effective at preventing catastrophic forgetting in long-term self-learning.

Limitations of self-learning. The goal of self-learning is to enable more efficient model updates, especially in application domains where new information is constantly emerging. This approach excels in dynamic environments. However, it may offer little to no benefits in narrow, closed application domains, such as chatbots that are designed to answer questions about specific products or services. In such cases, the better approach is to simply prepare training data with complete facts of the domain from the beginning. Furthermore, self-learning heavily depends on the quality of the initial model. If the initial model is not sufficiently strong, it may generate nonsensical questions during self-learning. Because of this, it is advised to evaluate the model's Self-Learning Capability before subjecting it to independent learning.

XII. 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 [43] 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 [25], though there, the focus was on increasing the model's factuality. In [24], RLHF was used, even though it highlights a similar idea of making the model aware of its own hidden states. In [23], they use a reward function to make the model admit *"I don't know"*. Meanwhile, [53] described a learning task that was coincidentally also termed "Into the Unknown", but their definition is slightly different: it is a learning task where the model is asked to choose from two options the piece of information that would provide new knowledge to the available context, yet the other option might trap the model due to closely resembling the existing knowledge.

Direct Awareness Optimization could improve the self-learning capability of a model. Making the model aware of its own knowledge limit would allow it to ask "better" questions during self-learning; *better* in the sense that the answers to those questions would actually widen the knowledge span of the model. Another aspect of *better* is whether those questions could be considered *meaningful* or not; for example, we might find the question *"How fast does F-16 fly?"* meaningful and could be learned while the question *"How fast do cats fly on Mars?"* not meaningful and does not need to be learned. Such Direct Awareness Optimization is one of our future directions.

(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 by adding some particular context and also increase the preferability score of one example response. This would associate the context of the prompt with a particular PoV. Similarly, adapting DPO would 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.

(5) Decision Making, AGI, Sentience. 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 might lead to a sentient AI.

XIII. 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 [25] and supported by the findings in [16]. However, if some incorrect knowledge was repeated in the models' past training data, either accidentally or through deliberate poisoning, the models may become consistent in producing incorrect information. One of our future directions is investigating the integration of a reference-based truthfulness checker, such as FactScore [14], in the self-learning loop, which may allow the model to 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. We have provided a successful demonstration of one full self-learning cycle in our experiment. Still, a deeper study into extensive cycles of self-learning is needed.

(3) Experiments were limited to the English language. We believe that self-learning can be performed in any language. However, further studies would be required to calibrate the brevity coefficient for the SLC score when working with a different language.

XIV. 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 is especially effective at enhancing an LLM's capability to Self-Learn. We also found that small models tend to struggle to learn independently. Finetuning or alignment can improve SLC by allowing the model to understand instructions. Yet, if a model's pretraining data contained some instruction examples, the model might be able to Self-Learn even though it has not been explicitly instruction-tuned. Finally, we discussed various possible issues, extensions, and applications of self-learning.

References

[1] Y. Onoe, M. Zhang, E. Choi, and G. Durrett, "Entity cloze by date: What LMs know about unseen entities," in *Findings of the Association for Computational Linguistics: NAACL 2022*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds. Seattle, United States: Association for Computational Linguistics, Jul. 2022.

[2] Z. Ji, D. Chen, E. Ishii, S. Cahyawijaya, Y. Bang, B. Wilie, and P. Fung, "Llm internal states reveal hallucination risk faced with a query," 2024. [Online]. Available: https://arxiv.org/abs/2407.03282

[3] P. Miłkowski, K. Karanowski, P. Wielopolski, J. Kocoń, P. Kazienko, and M. Zieba, "Modeling uncertainty in personalized emotion prediction with normalizing flows," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2023, pp. 757–766.

[4] B. Koptyra, A. Ngo, Ł. Radliński, and J. Kocoń, "Clarin-emo: Training emotion recognition models using human annotation and chatgpt," in *International Conference on Computational Science*. Springer, 2023, pp. 365–379.

[5] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza *et al.*, "Chatgpt: Jack of all trades, master of none," *Information Fusion*, vol. 99, p. 101861, 2023.

[6] K. Kanclerz, K. Karanowski, J. Bielaniewicz, M. Gruza, P. Miłkowski, J. Kocoń, and P. Kazienko, "Pals: Personalized active learning for subjective tasks in nlp," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 13 326–13 341.

[7] P. Kazienko, J. Bielaniewicz, M. Gruza, K. Kanclerz, K. Karanowski, P. Miłkowski, and J. Kocoń, "Human-centered neural reasoning for subjective content processing: Hate speech, emotions, and humor," *Information Fusion*, vol. 94, pp. 43–65, 2023.

[8] J. Kocoń, "Deep emotions across languages: A novel approach for sentiment propagation in multilingual wordnets," in *2023 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2023.

[9] T. Ferdinan and J. Kocoń, "Fortifying nlp models against poisoning attacks: The power of personalized prediction architectures," *Information Fusion*, p. 102692, 2024.

[10] C. Fan, J. Lin, R. Mao, and E. Cambria, "Fusing pairwise modalities for emotion recognition in conversations," *Information Fusion*, vol. 106, p. 102306, 2024.

[11] N. Kandpal, H. Deng, A. Roberts, E. Wallace, and C. Raffel, "Large language models struggle to learn long-tail knowledge," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., vol. 202. PMLR, 23–29 Jul 2023, pp. 15 696–15 707.

[12] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch, and N. Carlini, "Deduplicating training data makes language models better," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022.

[13] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring how models mimic human falsehoods," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Dublin, Ireland: Association for Computational Linguistics, May 2022.

[14] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. Koh, M. Iyyer, L. Zettlemoyer, and H. Hajishirzi, "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*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023.

[15] A. Azaria and T. Mitchell, "The internal state of an LLM knows when it's lying," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023.

[16] P. Manakul, A. Liusie, and M. Gales, "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*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023.

[17] Z. Cao, Y. Yang, and H. Zhao, "Autohall: Automated hallucination dataset generation for large language models," 2023.

[18] Z. Yin, Q. Sun, Q. Guo, J. Wu, X. Qiu, and X. Huang, "Do large language models know what they don't know?" in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023.

[19] Z. Ji, T. Yu, Y. Xu, N. Lee, E. Ishii, and P. Fung, "Towards mitigating LLM hallucination via self reflection," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, 2023.

[20] J. Luo, C. Xiao, and F. Ma, "Zero-resource hallucination prevention for large language models," 2023.

[21] K. Li, O. Patel, F. Viégas, H. Pfister, and M. Wattenberg, "Inference-time intervention: Eliciting truthful answers from a language model," in *Thirty-seventh Conference on NIPS*, 2023.

[22] Z. Ji, Z. Liu, N. Lee, T. Yu, B. Wilie, M. Zeng, and P. Fung, "RHO: Reducing hallucination in open-domain dialogues with knowledge grounding," in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, 2023.

[23] X. Liu and P. Sajda, "Roe: A computational-efficient anti-hallucination fine-tuning technology for large language model inspired by human learning process," in *Brain Informatics*, F. Liu, Y. Zhang, H. Kuai, E. P. Stephen, and H. Wang, Eds. Cham: Springer Nature Switzerland, 2023, pp. 456–463.

[24] Y. Liang, Z. Song, H. Wang, and J. Zhang, "Learning to trust your feelings: Leveraging self-awareness in llms for hallucination mitigation," 2024.

[25] K. Tian, E. Mitchell, H. Yao, C. D. Manning, and C. Finn, "Fine-tuning language models for factuality," in *The Twelfth International Conference on Learning Representations*, 2024.

[26] Y.-S. Chuang, Y. Xie, H. Luo, Y. Kim, J. Glass, and P. He, "Dola: Decoding by contrasting layers improves factuality in large language models," 2024. [Online]. Available: https://arxiv.org/abs/2309.03883

[27] Y.-S. Chuang, L. Qiu, C.-Y. Hsieh, R. Krishna, Y. Kim, and J. Glass, "Lookback lens: Detecting and mitigating contextual hallucinations in large language models using only attention maps," 2024. [Online]. Available: https://arxiv.org/abs/2407.07071

[28] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[29] J. Jang, S. Ye, S. Yang, J. Shin, J. Han, G. Kim, S. J. Choi, and M. Seo, "Towards continual knowledge learning of language models," in *ICLR*, 2022.

[30] X. Jin, D. Zhang, H. Zhu, W. Xiao, S.-W. Li, X. Wei, A. Arnold, and X. Ren, "Lifelong pretraining: Continually adapting language models to emerging corpora," in *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, A. Fan, S. Ilic, T. Wolf, and M. Gallé, Eds. virtual+Dublin: Association for Computational Linguistics, May 2022.

[31] Z. Ke, Y. Shao, H. Lin, T. Konishi, G. Kim, and B. Liu, "Continual pre-training of language models," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://arxiv.org/abs/2302.03241

[32] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of Learning and Motivation*, G. H. Bower, Ed. Academic Press, 1989, vol. 24, pp. 109–165.

[33] R. Ratcliff, "Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions." *Psychological Review*, vol. 97, no. 2, pp. 285–308, 1990.

[34] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[35] C. Zhu, A. S. Rawat, M. Zaheer, S. Bhojanapalli, D. Li, F. Yu, and S. Kumar, "Modifying memories in transformer models," 2020.

[36] A. Sinitsin, V. Plokhotnyuk, D. Pyrkin, S. Popov, and A. Babenko, "Editable neural networks," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://arxiv.org/pdf/2004.00345.pdf

[37] Z. Ke, B. Liu, N. Ma, H. Xu, and L. Shu, "Achieving forgetting prevention and knowledge transfer in continual learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 22 443–22 456.

[38] E. Mitchell, C. Lin, A. Bosselut, C. Finn, and C. D. Manning, "Memory-based model editing at scale," in *International Conference on Machine Learning*, 2022. [Online]. Available: https://arxiv.org/pdf/2206.06520.pdf

[39] J. Jaynes, *The origin of consciousness in the breakdown of the bicameral mind.*, ser. The origin of consciousness in the breakdown of the bicameral mind. Boston, MA, US: Houghton, Mifflin and Company, 1990.

[40] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[41] N. Reimers and I. Gurevych, "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, 11 2019.

[42] G. Miller, E. Newman, and E. Friedman, "Length-frequency statistics for written english," *Information and Control*, vol. 1, no. 4, pp. 370–389, 1958.

[43] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[44] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

[45] P. Zhang, G. Zeng, T. Wang, and W. Lu, "Tinyllama: An open-source small language model," 2024.

[46] M. Abdin, S. A. Jacobs, A. A. Awan, J. Aneja, A. Awadallah, and H. A. et al., "Phi-3 technical report: A highly capable language model locally on your phone," 2024.

[47] B. Peng, D. Goldstein, Q. Anthony, A. Albalak, E. Alcaide, S. Biderman, E. Cheah, X. Du, T. Ferdinan, H. Hou, P. Kazienko, K. K. GV, J. Kocoń, B. Koptyra, S. Krishna, R. M. J. au2, N. Muennighoff, F. Obeid, A. Saito, G. Song, H. Tu, S. Woźniak, R. Zhang, B. Zhao, Q. Zhao, P. Zhou, J. Zhu, and R.-J. Zhu, "Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence," 2024.

[48] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://arxiv.org/abs/2106.09685

[49] OpenAI, J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya et al., "Gpt-4 technical report," 2024.

[50] Wikimedia Foundation, "Wikimedia downloads," n.d. [Online]. Available: https://dumps.wikimedia.org

[51] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," https://github.com/tatsu-lab/stanford_alpaca, 2023.

[52] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004.

[53] I. R. McKenzie, A. Lyzhov, M. M. Pieler, A. Parrish, A. Mueller, A. Prabhu, E. McLean, X. Shen, J. Cavanagh, A. G. Gritsevskiy, D. Kauffman, A. T. Kirtland, Z. Zhou, Y. Zhang, S. Huang, D. Wurgaft, M. Weiss, A. Ross, G. Recchia, A. Liu, J. Liu, T. Tseng, T. Korbak, N. Kim, S. R. Bowman, and E. Perez, "Inverse scaling: When bigger isn't better," *Transactions on Machine Learning Research*, 2023.