## A survey on Hallucination in LLMs

**Abstract:**

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP) but face challenges such as hallucinations, where models generate incorrect or nonsensical information. This paper comprehensively reviews the causes, phenomena, and mitigation strategies of hallucinations in LLMs. Various approaches, including self-reflection, retrieval-augmented generation, and knowledge-enhanced models, are discussed to address hallucination issues. The study aims to inform future research and practical implementations in the field of LLMs.

**Keywords:**

Large Language Models, Hallucinations, Self-reflection, Retrieval-Augmented Generation, Knowledge-Enhanced Language Models, Natural Language Processing, Mitigation Strategies.

## Introduction

###### Background

The advent of LLMs has marked a pivotal milestone in the fields of artificial intelligence and NLP. These models, distinguished by their extensive parameter count and vast knowledge bases, have significantly influenced research, industry, and society by generating coherent and contextually relevant text. However, a critical challenge faced by LLMs is the phenomenon of "hallucination," where models produce plausible but factually incorrect or nonsensical information[1]. As LLMs are increasingly deployed across various applications, the issue of hallucination has raised substantial safety and ethical concerns. These inaccurate outputs can mislead users and lead to unpredictable consequences, underscoring the necessity to address this problem comprehensively. Investigating and mitigating hallucination in LLMs is crucial for ensuring their reliability and safety, thereby enhancing their practical utility and contributing to the advancement of NLP and AI as a whole[2].

###### Related Work

According to the prior exploration, hallucinations in LLMs arise from multiple sources throughout their development process, including data issues (insufficient quantity, poor quality, statistical biases), model-related factors (weak vision models, language model dominance, alignment interface weaknesses), training challenges (inappropriate loss functions, lack of reinforcement learning), and inference problems (diluted attention to visual content during generation)[2,3]. These factors collectively contribute to generating unfaithful or nonsensical outputs. In this survey[2], categorizes LLM hallucinations into three types: input-conflicting, context-conflicting, and fact-conflicting hallucinations. Input-conflicting hallucinations occur when generated content deviates from user input. Context-conflicting hallucinations arise when generated content contradicts previously generated information. Fact-conflicting hallucinations involve content that contradicts established world knowledge. While input- and context-conflicting hallucinations have been extensively studied, recent research focuses primarily on fact-conflicting hallucinations due to their complexity and significant impact on practical applications[5,6,7,8]. Various benchmarks have been proposed to evaluate hallucinations in LLMs, focusing on the ability to generate and distinguish factual statements. These benchmarks involve tasks like question-answering and text completion, often with human annotation for quality assurance, such as designing questions to elicit falsehoods (TruthfulQA) or manually validating generated data (FACTOR)[9,10]. However, these benchmarks face limitations: hallucinated content is often induced by manually designed prompts rather than naturally generated by LLMs[11]; most focus solely on detecting factuality hallucinations, neglecting faithfulness hallucinations[7,2]; and many address only sentence-level and passage-level hallucinations, overlooking the importance of dialogue-level detection, crucial for dialogue systems[8,12].Model development approaches aim to enhance LLMs' internal mechanisms by leveraging pre-existing knowledge to improve reasoning and factual accuracy, incorporating uncertainty-based loss functions to reduce hallucinations, and embedding factual knowledge directly into the model's architecture during training. New decoding strategies, such as innovative prompt refinement techniques and context-aware generation methods, have also been introduced to prevent the occurrence of hallucinated content[13,14,15,16].

###### Objective

The main objective of this paper is to provide a comprehensive review of hallucinations in LLMs, covering the causes, phenomena, definitions, classifications, evaluation benchmarks, and mitigation strategies. This research will systematically outline and summarize the relevant concepts and background, analyze the fundamental principles and methodologies of core technologies for detecting and mitigating hallucinations, and evaluate the experimental performance of key technologies and benchmarks. Furthermore, it will discuss the advantages, limitations, and future development prospects of these technologies, offering a forward-looking perspective on their evolution. By thoroughly reviewing current and promising techniques to mitigate hallucinations, this paper aims to inform and guide future research and practical implementations in the field of LLM.

Figure 1: the structure of this survey

## Definitions

###### Large Language Model

Before delving into the hallucination issues in LLMs, it is essential to understand their overall training process and architecture, as these factors are closely tied to the occurrence of hallucinations. LLMs typically undergo three primary training stages: pre-training, supervised fine-tuning (SFT), and reinforcement learning from human feedback (RLHF)[2].

* Pre-training. Pre-training is a foundational stage for LLMs to acquire extensive knowledge and capabilities[18]. During this stage, language models are trained in an unsupervised manner on large textual corpora, aiming to predict the next token in a sequence autoregressively. This process allows the models to learn language syntax, world knowledge, and reasoning abilities, forming a robust foundation for further fine-tuning tasks. Recent studies suggest that predicting subsequent words is akin to compressing significant information, which is crucial for the model's nuanced understanding of language and the world.
* Supervised Fine-Tuning. While pre-training equips LLMs with substantial knowledge, it primarily optimizes the models for text completion tasks[19]. This often results in a misalignment between the model's objective of predicting the next word and the user's objective of obtaining specific, desired responses. Supervised fine-tuning addresses this gap by training LLMs on annotated datasets containing pairs of instructions and desired responses. This process enhances the model's capabilities and controllability, allowing it to perform better on unseen tasks and align more closely with user preferences. Instruction tuning (IT) further refines this alignment, making the model's behavior more predictable and controllable, and enabling rapid adaptation to specific domains without extensive retraining.
* Reinforcement Learning from Human Feedback. Although SFT significantly improves LLM performance, there remains spare for better alignment with human preferences[20]. RLHF is a method that integrates human feedback to fine-tune LLMs further. This approach uses a preference model trained to predict human preference rankings for given prompts and responses. The LLM is then optimized to generate outputs that maximize the rewards from this preference model, often using reinforcement learning algorithms such as Proximal Policy Optimization (PPO)[21]. RLHF effectively enhances LLM alignment, guiding the models to produce high-quality, user-aligned responses.

## **2.2 Large Language Model in Formalized contexts**

Understanding the complete training process of LLMs still leaves us with challenges in defining hallucination in practical scenarios. The term "hallucination" remains abstract and difficult to quantify. In real-world applications, due to the complexity of model inputs and outputs, it is challenging to define what constitutes a hallucination. Typically, hallucination refers to outputs that do not align with factual information, but defining "factual" itself can be complex. The discrepancies and inconsistencies need to be discussed based on the specific contexts of inputs and outputs. To address this, we move away from the complex notion of "correctness" in the real world and instead define hallucination within a formalized, mathematical framework. By doing so, we can analyze hallucinations precisely. In this context, we establish a benchmark for hallucinations by using a subset of real-world corpora as input samples. These samples are fed into the LLM, which is trained and updated until it meets the stopping criteria. The trained LLM is then deployed to generate outputs for unknown strings. Hallucination is defined by comparing the LLM's output with the ground truth value . The difference between these values indicates the presence of hallucination. It is important to note that the occurrence and manifestation of hallucinations can vary depending on the specific tasks that the LLM is performing.

## **2.3 How it generate and perform**

**Common Causes of Hallucinations Across Modalities**

Hallucinations in LLMs and multimodal models arise from several common factors across different modalities (text, images, videos, and audio)[17]. These include:

**Hallucinations from Data Stage in LLMs**

Pre-training data serves as the foundation for large language models (LLMs), enabling them to develop general capabilities and acquire factual knowledge (Zhou et al., 2023a). However, this stage can inadvertently become a source of hallucinations due to two main issues: flawed data sources and poor utilization of factual knowledge.

* **flawed data sources.** Expanding pre-training data enhances LLMs' abilities, but maintaining consistent data quality is challenging, often introducing misinformation and biases. This can result in two primary issues: misinformation and biases, and knowledge boundary limitations. Misinformation can be reinforced during training, leading to misleading outputs. Additionally, LLMs tend to memorize duplicated data, leading to duplication bias, where the model over-prioritizes repeated information and deviates from the desired content. Social biases, such as associating certain professions with specific genders, can also be inadvertently learned from biased internet texts, resulting in contextually inconsistent outputs. Furthermore, the lack of domain-specific knowledge and up-to-date facts limits the model's knowledge, causing hallucinations when confronted with specialized or current information.
* **poor utilization of factual knowledge.** Even when factual knowledge is present in the training data, LLMs may not use it effectively, leading to hallucinations. This poor utilization can manifest as inadequate integration of available facts, resulting in fragmented or misleading outputs. Contextual misapplication occurs when models use facts inappropriately, such as referencing historical events inaccurately in modern contexts. Additionally, LLMs often rely heavily on learned patterns, which can overshadow the accurate application of factual knowledge and result in plausible-sounding but factually incorrect content.

**Hallucinations from Training Stage in LLMs**

The training process of LLMs encompasses two primary stages: pre-training and alignment. During pre-training, LLMs learn general-purpose representations and world knowledge through transformer-based architectures, while the alignment stage involves adapting these models to better meet user instructions and preferences. Any deficiencies in these stages can inadvertently lead to hallucinations.

* **Pre-Training Issues.** Pre-training serves as the foundational stage for LLMs, utilizing a transformer-based architecture for causal language modeling on vast corpora. However, hallucinations may arise due to architectural flaws and exposure bias. The transformer architecture, while effective, can suffer from inadequate unidirectional representation and attention glitches. Unidirectional modeling, which predicts tokens based only on preceding tokens, limits the ability to capture complex contextual dependencies, increasing the risk of hallucinations. Additionally, transformer-based models, despite their capability to handle long-range dependencies, can exhibit unpredictable reasoning errors due to limitations of soft attention, where attention becomes diluted across positions as sequence length increases. Exposure bias also contributes to hallucinations. This occurs due to the discrepancy between training and inference phases. During training, models use ground truth tokens as input (teacher-forcing), but during inference, they rely on their own generated tokens. This inconsistency can cause a cascade of errors, where one erroneous token leads to further inaccuracies in subsequent predictions.
* **Alignment Issues.** Alignment involves supervised fine-tuning and RLHF, crucial for enhancing LLM capabilities and aligning them with human preferences. However, alignment can also introduce hallucinations through capability misalignment and belief misalignment. Capability misalignment occurs when the demands from alignment data exceed the model's inherent capability boundaries established during pre-training. This can force LLMs to produce content beyond their knowledge scope, increasing the risk of hallucinations. Belief misalignment refers to the discrepancy between the model's internal beliefs about the truthfulness of its statements and the outputs generated after alignment with human feedback. Despite refinements through human feedback, LLMs can sometimes produce outputs that diverge from their internal beliefs, often attempting to satisfy human evaluators at the expense of truthfulness. This phenomenon highlights the model's tendency to favor user opinions over factual accuracy. Studies indicate that RLHF-trained models exhibit significant sycophantic behavior, choosing clearly incorrect answers to align with perceived user preferences, driven by biases in both human feedback and preference models.

**Hallucination from Inference Stage in LLMs**

The decoding process is crucial for realizing the full potential of LLM post-training and alignment. However, certain shortcomings during this stage can result in hallucinations. Two primary factors contributing to hallucinations during decoding are inherent sampling randomness and imperfect decoding representation.

* **Decoding Strategy Flaws.** Inherent sampling randomness in stochastic sampling methods, designed to avoid the "likelihood trap" of high-probability but low-quality text, can lead to hallucinations. Higher sampling temperatures make token distributions more uniform, increasing the likelihood of selecting less frequent tokens, thus producing hallucinated outputs.
* **Imperfect Representation**. Insufficient context attention, where models focus too much on the partially generated text at the expense of broader context, leads to faithfulness hallucinations. Additionally, the softmax bottleneck limits the expressiveness of output probability distributions, making it difficult for models to handle complex distributions and prioritize the correct next words, thereby increasing the risk of hallucinations.

**Specific Causes of Hallucinations**

For text, hallucinations often stem from misalignment of context and knowledge during training and an imperfect understanding of contextual nuances, leading to irrelevant information generation. In images, inaccuracies arise due to alignment issues between image and text data and flawed visual representations[22,23]. Videos face challenges with temporal inconsistencies and maintaining consistent information across frames[24,25]. In audio, hallucinations are primarily caused by difficulties in accurately encoding complex audio features and generating text that inaccurately reflects the audio content[26].

## **2.4 Classification of Hallucination**

**Factual Hallucinations**

* **Intrinsic Hallucination.** This type of hallucination occurs when the generated content contradicts the source content. For example, in Cookie Run: Kingdom, if the source content states, "Gingerbread Cookie is one of the strongest characters," but the generated description says, "Gingerbread Cookie is one of the weakest characters," this is an intrinsic hallucination.
* **Extrinsic Hallucination.** This occurs when the generated content cannot be verified from the source content, meaning the output can neither be supported nor contradicted by the source. Even though such hallucinations may sometimes include factually correct information from external sources, they should be treated cautiously due to their unverifiable nature. For instance, if the source content in Cookie Run: Kingdom does not mention a new wizard character, but the generated task states, "A new wizard character is joining the game," this is an extrinsic hallucination.

**Faithfulness Hallucinations**

* **Input-Conflicting Hallucination.** This hallucination happens when the generated content deviates from the user-provided input. User input typically includes task instructions (e.g., summarization prompt) and task input (e.g., the document to be summarized). For example, in a Cookie Run: Kingdom guide, if a user requests, "Please provide information on how to upgrade buildings," but the generated response says, "You can upgrade your characters by completing battle tasks," this is an input-conflicting hallucination.
* **Context-Conflicting Hallucination.** This type of hallucination occurs when the generated content conflicts with previously generated information. This issue usually arises in long or multi-turn responses when the model fails to maintain contextual consistency. For instance, in a storyline generation in Cookie Run: Kingdom, if the initial text mentions, "Chocolate Cookie joined Gingerbread Cookie's team," but later generated text says, "Chocolate Cookie decided to leave the team," this inconsistency is a context-conflicting hallucination.

## **2.5 Benchmark**

The necessity of hallucination detection lies in identifying nonsensical or untruthful content generated by models, which is crucial for maintaining the credibility of model outputs. Early hallucination detection benchmarks were primarily created through manual methods or traditional language models, such as FactCollect and HADES, but these benchmarks did not fully reflect the natural usage of LLMs[26,27]. In recent years, researchers have developed modern detection benchmarks that focus more on content directly generated by models. These include evaluating the factual accuracy of model responses through question-answering tasks or using knowledge graphs and natural language text datasets to detect factual hallucinations, such as FactCHD. While most benchmarks focus on factual hallucinations, some newer benchmarks also include the detection of faithfulness hallucinations, which assess the coherence and relevance of generated content. Existing benchmarks evaluate hallucinations in different application tasks for LLMs, such as question answering, biography generation, and text completion. Each task format provides samples to demonstrate potential hallucinations in generated content. Many benchmark datasets involve human annotation to ensure data quality, such as TruthfulQA, which designs questions and employs human annotators to verify answer consistency[9]. Evaluation methods for hallucinations include human evaluation, model-based automatic evaluation, and rule-based automatic evaluation. Although human evaluation is reliable and interpretable, it is costly and may be subjective due to the need for extensive human annotation. Model-based automatic evaluation uses trained models to act as proxies for human evaluation, assessing factual consistency with pre-existing models. Rule-based automatic evaluation employs rule-based classification metrics to evaluate the model's ability to distinguish factual from non-factual statements[28]. Additionally, while many benchmarks focus on sentence-level and passage-level hallucination detection, there are currently no benchmarks specifically for dialogue-level hallucination detection. DiaHalu addresses this gap by focusing on dialogue-level hallucination detection, emphasizing the importance of different types of hallucination detection and their application in dialogue generation[5]. In summary, the development of hallucination detection benchmarks has evolved from manual methods to more sophisticated modern approaches that reflect the natural usage of LLMs. These benchmarks play a critical role in ensuring the factual accuracy, coherence, and relevance of model-generated content, with ongoing research continuously improving these methods to address new challenges[5].

## Alleviation of Hallucination

Currently, numerous survey articles categorize methods for mitigating hallucinations from various angles. These include approaches based on prompt engineering and model optimization, as well as classification methods that address data processing before, during, and after training. Various research papers have proposed their own improved strategies for mitigating hallucinations, such as multimodal contrastive decoding, over-trust penalty, retrospection reallocation strategy, and induction-contrast decoding strategies utilizing reinforcement learning principles. Moreover, different fields have developed specific solutions to address hallucinations, whether for broad tasks or specialized applications. In this chapter, we will organize the existing survey articles, presenting a clearer and more comprehensive overview of hallucination mitigation methods[30-35].

**3.1 Prompt Engineering**

**3.1.1 Retrieval Augmented Generation**

Retrieval-Augmented Generation (RAG) refers to the optimization of LLMs outputs by allowing them to reference authoritative knowledge bases outside of their training data before generating responses[33]. Trained on vast amounts of data with billions of parameters, LLMs generate outputs for tasks such as answering questions, translating languages, and completing sentences. RAG enhances these already powerful models by enabling access to specific domain or organizational knowledge bases without the need for retraining. This approach cost-effectively improves LLM outputs, ensuring relevance, accuracy, and practicality across various contexts. We can categorize different types of tasks that use retrieval-augmented generation, and the diagram below illustrates the basic working principle.

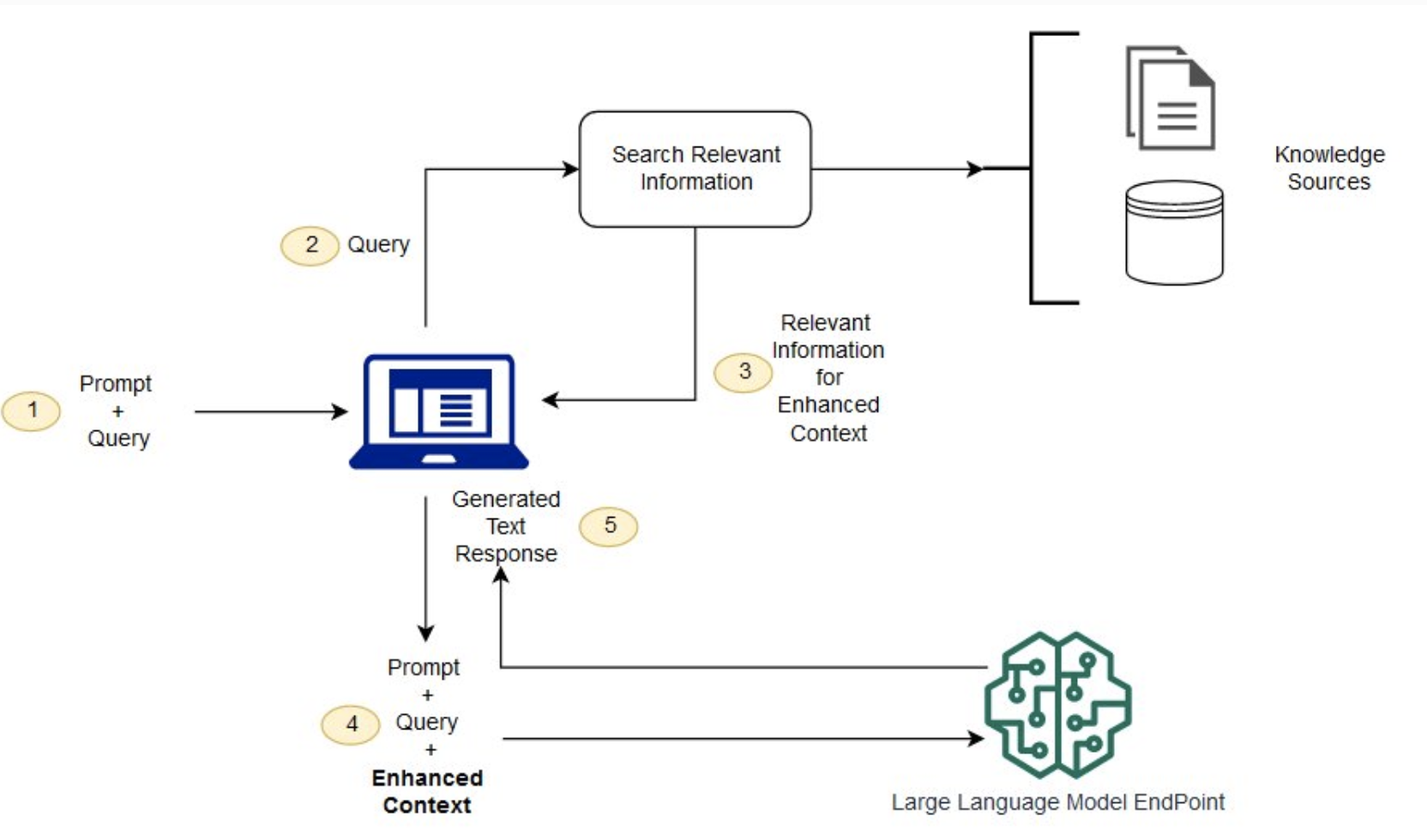


Figure: Basic working principle of RAG.

Additionally, we categorize RAG models into different types and applicable functions for various periods, as illustrated in the figure below.

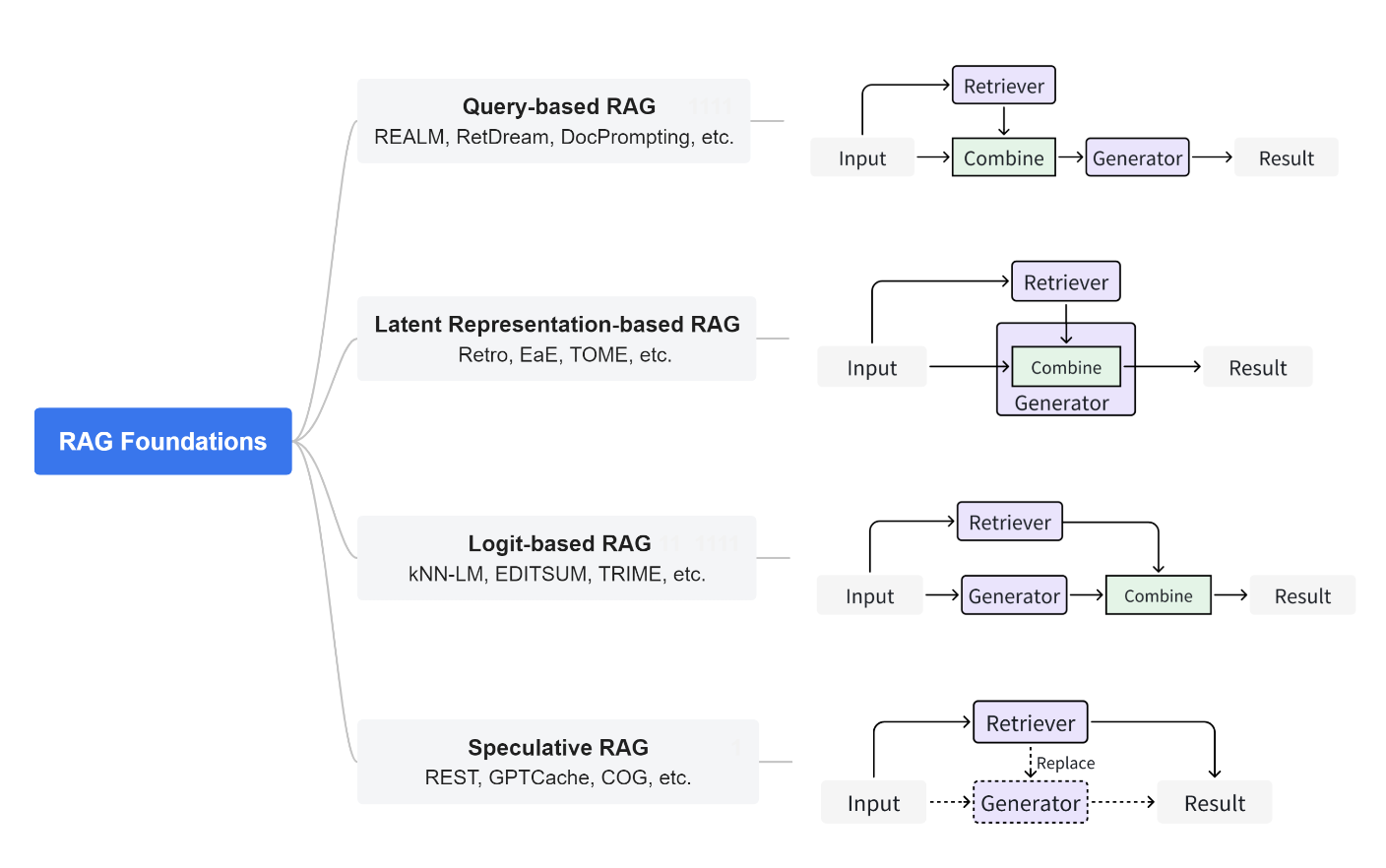


Figure: Different types of RAG

**Query-based RAG.** Also known as prompt augmentation, this model integrates user queries with information retrieved from documents at the initial stage of the language model input. This approach is widely adopted in RAG applications. Once documents are retrieved, their contents are merged with the user's original query, creating a combined input sequence. This enhanced sequence is then fed into a pre-trained language model to generate responses.

**Latent Representation-based RAG.** In this framework, retrieved objects are integrated into the generation model as latent representations, enhancing the model's understanding and the quality of generated content. This approach shows significant potential and adaptability in handling code, structured knowledge, and multimodal data. Particularly in code-related domains, techniques like EDITSUM, BASHEXPLAINER, and RetrieveNEdit use the Fusion-in-Decoder (FiD) method to facilitate integration through encoder processing. Methods like Re2Com and RACE also employ multiple encoders designed for different types of inputs.

**Logit-based RAG.** In this model, the generation process incorporates retrieved information through logit fusion during decoding. Typically, logits are summed or combined to produce stepwise generation probabilities. In code-to-text conversion tasks, Rencos generates multiple summary candidates for retrieved code and normalizes them using edit distance to compute final probabilities, selecting the summary that best matches the original code. In code summarization tasks, EDITSUM improves summary quality by integrating prototype summaries at the probability level. For text-to-code tasks, the kNN-TRANX model combines confidence networks and meta-knowledge to merge retrieved code snippets, utilizing seq2tree structures to generate target code that closely matches the input query. This method is particularly suitable for sequence generation tasks, focusing on generator training and effectively utilizing probability distributions for subsequent tasks.

**Speculative RAG.** This approach aims to save resources and speed up response times by using retrieval instead of pure generation. REST technology replaces small models in speculative decoding with retrieval to achieve draft generation. GPTCache addresses the high latency issue of using LLM APIs by constructing a semantic cache to store LLM responses. Speculative RAG is primarily applicable to sequence data, decoupling generators and retrievers so that pre-trained models can be directly used as components. This paradigm allows for the exploration of broader strategies to effectively utilize retrieved content.

**3.1.2 Self-reflection**

Self-Reflection is a framework that reinforces language-based agents through linguistic feedback. "Self-reflection is a novel paradigm of 'verbal' reinforcement, pairing the agent’s memory encoding with parameter selection by the LLM." At a high level, self-reflection transforms feedback from the environment (either free-form language or scalar) into linguistic feedback, providing context for the LLM agent in the next iteration. This process helps agents learn quickly and effectively from past mistakes, enhancing performance across various advanced tasks.

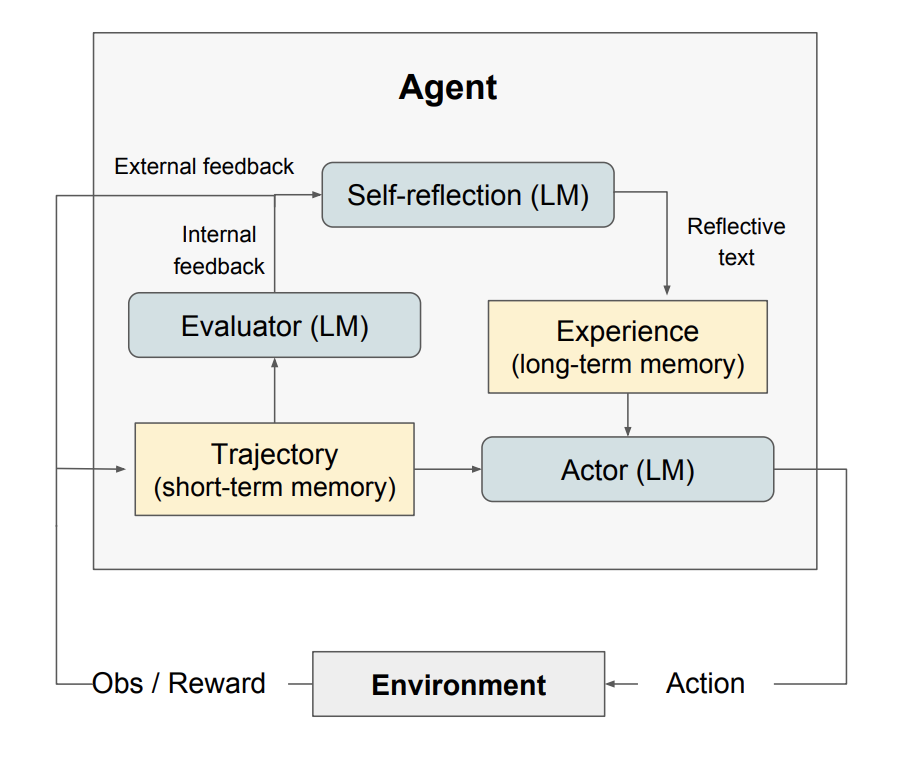


Figure: Basic working principle of self-reflection[36]

As illustrated, self-reflection consists of three distinct models:

**Actor.** Generates text and actions based on state observations. The actor takes actions in the environment and receives observations, forming trajectories. Chain-of-Thought (CoT) and ReAct are used as actor models. Additionally, a memory component is included to provide the agent with extra contextual information.

**Evaluator.** Assesses the outputs of the actor. Specifically, it takes the generated trajectories (also known as short-term memory) as input and outputs a reward score. Different reward functions (LLM-based and rule-based heuristics) are used depending on the character and the task.

**Self-Reflection.** Produces linguistic reinforcement cues to help the actor improve. This role is fulfilled by a large language model that provides valuable feedback for future iterations. The self-reflection model uses reward signals, current trajectories, and its long-term memory to generate specific and relevant feedback, which is stored in the memory component. The agent uses these experiences (stored in long-term memory) to rapidly improve decision-making.

Overall, the critical steps of self-reflection are(a) defining the task (b) generating trajectories (c) evaluation (d) executing self-reflection (e) generating the next trajectory.

Self-reflection mitigates hallucinations in LLMs by integrating feedback from the environment into the model's memory, enabling the retention of relevant context across iterations. It employs an evaluative mechanism to assess outputs and provide targeted feedback, guiding the model to correct errors and refine responses. This process allows LLMs to learn quickly from past mistakes, improving decision-making and reducing the likelihood of generating inaccurate or hallucinatory content.

**3.1.3 Prompt Tuning**

**Zero-Shot Prompting.** Modern LLMs, trained on extensive data and fine-tuned with instructions, are capable of performing zero-shot tasks. Simply adjusting instructions has been shown to improve zero-shot learning. Instruction tuning involves fine-tuning the model on datasets described by instructions. Additionally, RLHF extends instruction tuning by adjusting the model to better align with human preferences. However, when zero-shot learning is ineffective, demonstrations or examples in the prompt are necessary, leading to few-shot prompting[38].

**Few-shot prompting.** enables contextual learning by providing examples. Here are some tips for effective few-shot learning[41,42]:

1. The distribution of the label space and the input text specified by the demonstrations are crucial, regardless of the accuracy of the labels for individual inputs.
2. The format used plays a key role in performance; even using random labels is significantly better than having no labels.
3. Other findings suggest that selecting random labels from the true label distribution (rather than a uniform distribution) is beneficial.

**Chain-of-Thought Prompting.** This approach teaches the model to learn from context by presenting correct examples or related knowledge. The goal is to make the model think like a human and find simple, effective solutions. For instance, when calculating Fibonacci sequences, using dynamic programming is much faster than recursion. By providing hints, we guide the model to learn human-preferred methods, reducing errors. However, this method is effective only for specific problems[39,40].

**Automatic Chain-of-Thought Prompting.** Manual creation of effective and diverse examples for chain-of-thought prompting can lead to suboptimal solutions. This paper proposes an automated method using LLMs to generate reasoning chains step-by-step with prompts like "let's think step by step." This process, though automated, can still produce errors. To mitigate these errors, diversity in demonstrations is crucial. The proposed Auto-CoT samples diverse problems and generates reasoning chains to create demonstrations[37].

Auto-CoT consists of two main stages:

Stage 1: Problem Clustering, which divides the given problems into several clusters.

Stage 2: Demonstration Sampling, which selects a representative problem from each cluster and uses Zero-Shot-CoT with simple heuristics to generate its reasoning chain.

Simple heuristics might include the problem's length (e.g., 60 tokens) and the number of reasoning steps (e.g., 5 steps). This encourages the model to use simple yet accurate demonstrations.

## **3.2 Model development**

**3.2.1 New decoding strategies**

New decoding strategies, through the design of specific techniques during the generation phase, significantly reduce hallucinations in outputs and enhance the accuracy and contextual relevance of the generated content.

**Context-Aware Decoding (CAD).** Effectively overrides the model's prior knowledge by contrasting output distributions, making it particularly suitable for tasks involving knowledge conflicts. The workflow is illustrated below, showing how this approach encourages the language model to focus on its context during generation. CAD samples from a new output distribution, amplifying the differences in output probabilities with and without contextual documents. This introduces a new form of contrastive decoding, effectively down-weighting prior knowledge when more relevant contextual information is provided. CAD can be used with off-the-shelf pre-trained language models without any additional training[43].

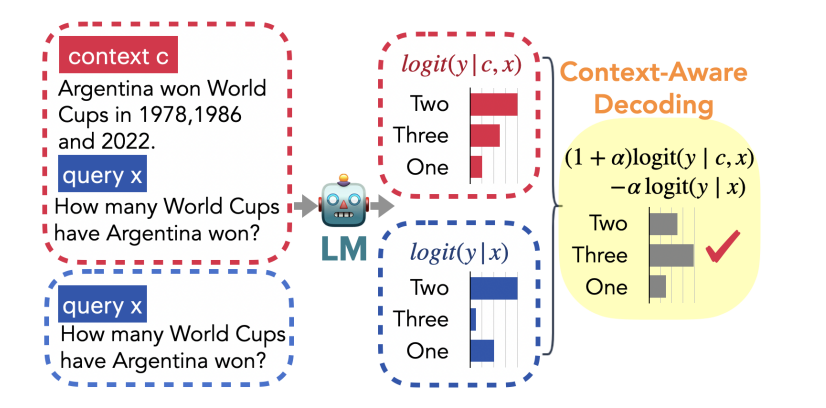
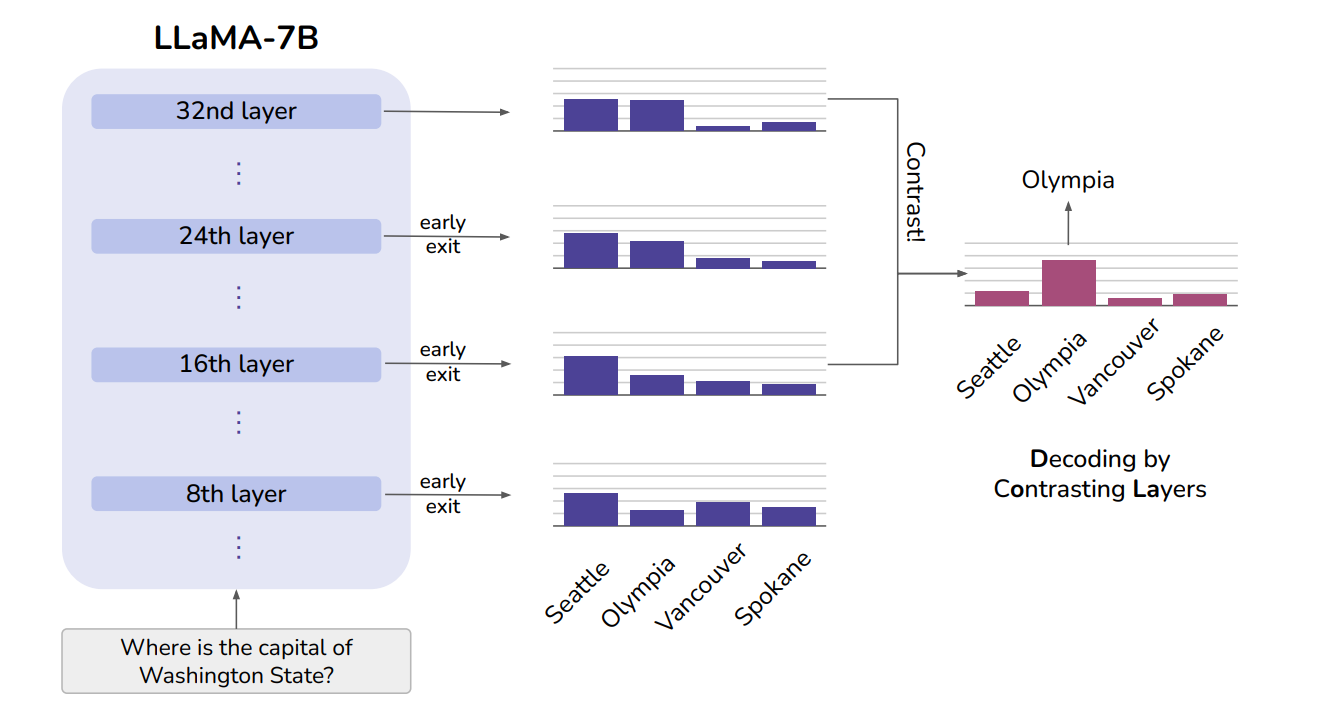


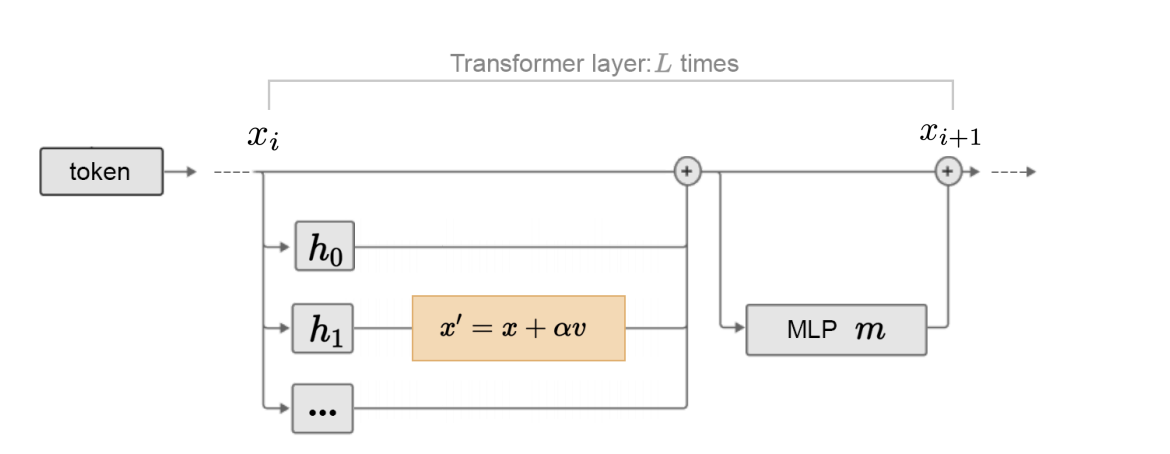
Figure: Basic working principle of CAD

**Decoding by Contrasting Layers (DoLa).** which leverages the factual knowledge encoded in specific Transformer layers by contrasting the logit differences between different layers, thus enhancing the truthfulness of the generated text. This paper highlights that hallucinations arise when generated content is not based on training data or factual information. Various factors, such as imperfect learning and decoding processes, contribute to hallucinations. Initial strategies to reduce hallucinations involved reinforcement learning with human feedback. Recent methods include inference-time consistency checks and multi-agent debates. Some approaches guide LLMs using human labels for inference-time interventions. Although the exact reasons for LMs producing hallucinations are not fully understood, one potential cause is the objective of maximum likelihood language modeling, which minimizes the forward KL divergence between data and model distributions. This objective may lead to large-scale search behavior, causing LMs to assign non-zero probabilities to sentences not fully consistent with the knowledge in the training data. Empirical evidence suggests that LMs trained on next-word prediction objectives with limited data tend to model surface patterns in training examples using linguistic knowledge rather than recognizing and generating real-world facts extracted from the training corpus. From a model interpretability perspective, Transformer-based models have been broadly shown to encode "lower-level" information (e.g., discourse tags) in the earlier layers and more "semantic" information in the later layers. Recent findings indicate that "knowledge neurons" are distributed in the uppermost layers of pre-trained BERT models. This suggests utilizing this modular knowledge encoding by amplifying factual knowledge within LMs through contrastive decoding, where the next-word output probability is derived from the logit differences between higher and lower layers. By emphasizing high-layer knowledge and down-weighting lower or intermediate-layer knowledge, LMs can potentially align more closely with factual information, thereby reducing hallucinations.



Figure：From the diagram, we can see that although "Seattle" maintains a high probability across all layers, likely because it appears syntactically reasonable, the probability of the correct answer "Olympia" increases when more factual knowledge is injected at higher layers. Therefore, contrasting the differences between layers can reveal the correct answer in this case[44].

**Inference-Time Intervention (ITI).** The authors explore the internal generation and prediction accuracy of LLMs. Generation accuracy assesses the correctness of the model's output layer, while prediction accuracy evaluates the authenticity of activation values in the model's intermediate layers. These two measures overlap significantly, but experiments show a considerable gap between generation and prediction accuracy, contributing to hallucinations. The ITI method addresses this by identifying a set of sparse attention heads with high prediction accuracy. During inference, the model's activations are adjusted along directions correlated with truthfulness until a complete and correct answer is generated. Multi-head attention essentially updates the input stream through residual operations, with probes acting as logistic regression predictors using intermediate activation values. Experimental results indicate that truthful information in LLMs is not confined to a single fixed direction but exists within a subspace. A natural approach is to apply interventions during inference to steer activations towards more truthful directions, enabling LLMs to produce more accurate answers and thereby reducing hallucinations[45].

Figure: Basic working principle of ITI[45]

**3.2.2 Knowledge-Enhanced Language Models**

Knowledge-enhanced language models (KE-LLMs) are a type of language model that utilizes external knowledge sources, such as knowledge graphs (KGs) and databases, and symbolic reasoning methods, such as logic, to improve their performance. They achieve this by incorporating knowledge into the language model's workflow, either by modifying how information is extracted (e.g., through retrieval from knowledge databases) or by applying logical reasoning. As a result, the language model gains access to additional information about the real world beyond what it can learn from training data, which can potentially mitigate hallucinations in formal domains. However, LLMs are often criticized for their lack of explainability, as they are essentially black-box models that encode knowledge implicitly through their parameters. This makes it difficult to understand and verify the knowledge that LLMs acquire. Additionally, LLMs perform reasoning using probabilistic models, which is a non-deterministic process. As a result, it is difficult for humans to directly obtain details and explanations about the specific patterns and functions that LLMs use to derive predictions and make decisions. Although some LLMs have the ability to explain their own predictions using "chains of thought," the explanations they generate can still be hallucinatory. This can severely limit the application of LLMs in high-stakes scenarios, such as medical diagnosis and legal judgment.

One potential solution is to integrate knowledge graphs (KGs) into LLMs. KGs store vast amounts of factual information in the form of triples (head entity, relation, tail entity), making them a structured and declarative representation of knowledge. KGs are crucial for many applications because they provide accurate and unambiguous knowledge. Additionally, they are known for their powerful symbolic reasoning capabilities, which can lead to explainable results. Moreover, KGs can evolve over time as new knowledge is continuously added. Furthermore, by involving experts in the construction of domain-specific knowledge graphs, KGs can be equipped with the ability to provide accurate and reliable domain-specific knowledge. However, building KGs can be challenging, and existing KG methods often struggle to handle the incompleteness and dynamic nature of real-world knowledge graphs. These methods often fail to effectively model unseen entities and represent new knowledge. Additionally, the rich textual information in KGs is often overlooked. Moreover, existing KG methods are often tailored to specific knowledge graphs or tasks, limiting their generalizability. Therefore, there is also a need to use LLMs to address the challenges faced by KGs. This can both improve the hallucination problem in large language models and address the shortcomings of knowledge graphs, as illustrated in the following figure:

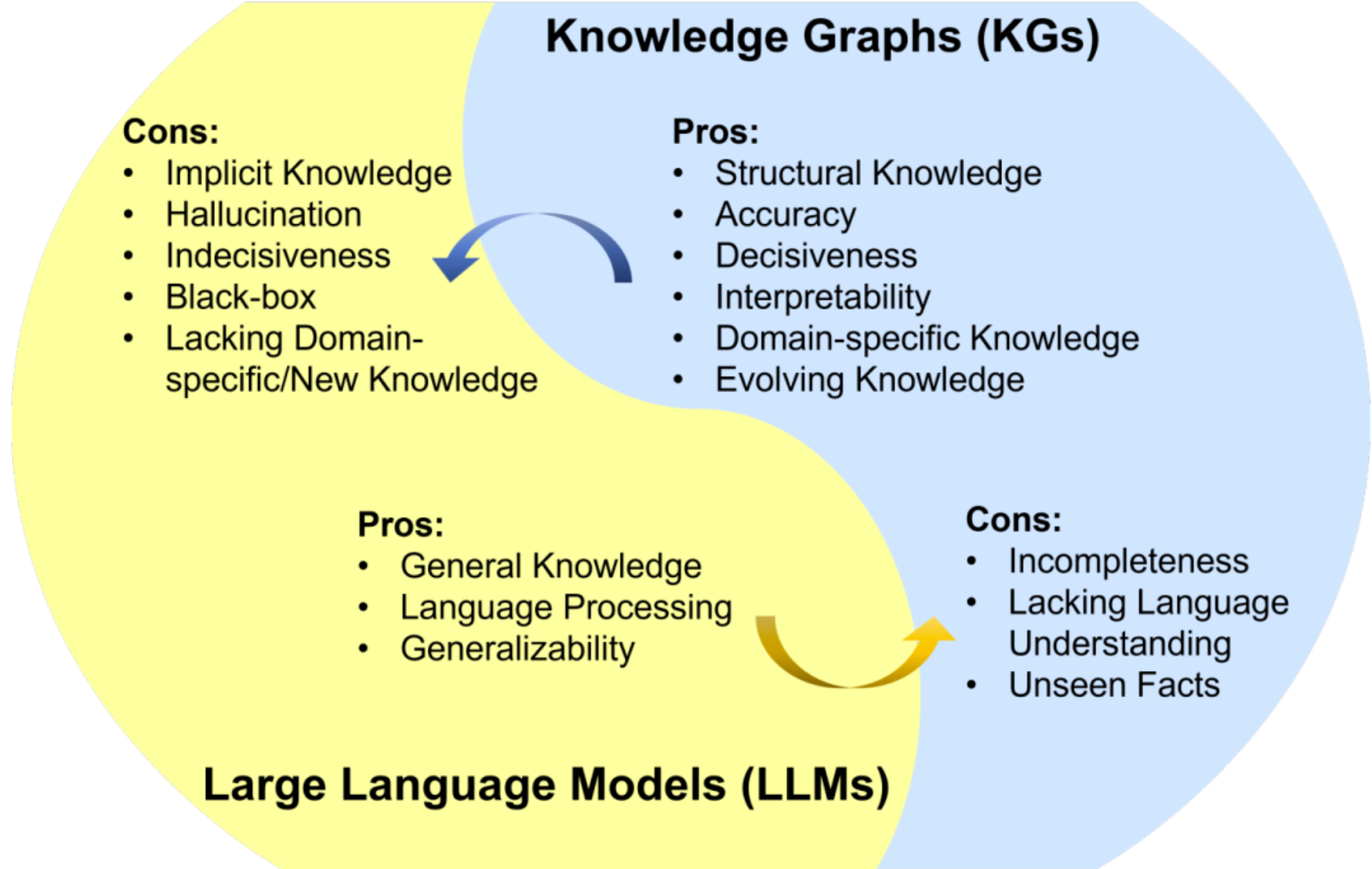


Figure:

**3.2.3 Fidelity-based Loss Function**

The fidelity-based loss function is a powerful technique for mitigating hallucinations in LLMs. It achieves this by comparing the LLM's output to real-world information and calculating the loss based on their similarity. This approach has proven effective in reducing hallucinations, as it encourages LLMs to generate outputs that are consistent with factual knowledge. However, the fidelity-based loss function also presents certain challenges. One major obstacle lies in acquiring accurate real-world information, especially for complex or nuanced tasks. This can be a time-consuming and resource-intensive process, limiting the applicability of this method. Additionally, computing the similarity between the LLM's output and real-world information can be computationally expensive, particularly when dealing with large volumes of data. We need to utilize high-quality training data. This involves collecting data from multiple sources and carefully curating it to ensure accuracy and minimize biases. Additionally, employing a diverse set of evaluation methods beacuse it involves assessing the LLM's performance on various datasets and tasks to ensure its generalization capabilities. Finally, incorporating regularization techniques can help prevent overfitting and promote more generalizable models. Despite its effectiveness in reducing hallucinations, the fidelity-based loss function may not be the most practical solution due to its inherent complexity and implementation challenges[46].

**3.2.4 Supervised Fine-Tuning**

Supervised fine-tuning (SFT) is a cornerstone technique in deep learning for model optimization. It involves leveraging a pre-trained deep learning model as the foundation and then fine-tuning it on a training set tailored to the specific target task. This process enables the model to effectively adapt to the task at hand.The SFT process unfolds in two distinct stages[46]. (a)Pre-training: A large-scale dataset is employed to train the pre-training model, empowering it to acquire representations of general features. (b) Fine-tuning: A smaller dataset, specifically curated for the target task, is used to fine-tune the pre-trained model, adapting it to the unique characteristics of the task or domain. Typically, fine-tuning is focused on lower layers, concentrating on the final few layers of the model, to better capture task-specific or domain-specific features. This approach essentially transitions the model from broad learning to deep learning, specifically tailored to the target domain task, thereby reducing the likelihood of hallucinations.SFT effectively combats hallucinations in large language models by, utilizing a pre-trained model as the foundation for task-specific fine-tuning; Imparting general feature representations through pre-training on a large-scale dataset; Adapting the model to the target domain through fine-tuning on a task-specific dataset; Focusing on lower layers during fine-tuning to capture task-specific features. These features help LLMs alleviate hallucinations.

## Conclusion

In conclusion, we have conducted a comprehensive examination of hallucinations in LLMs, encompassing their causes, phenomena, definitions, classifications, evaluation benchmarks, and mitigation strategies. Through a systematic overview and summary of relevant concepts and backgrounds, we have analyzed the fundamental principles and methodologies of core technologies for detecting and mitigating hallucinations, and evaluated the experimental performance of key technologies and benchmarks. Furthermore, we have discussed the advantages, limitations, and future development prospects of these technologies, offering a forward-looking perspective on their evolution. By summarizing relevant papers, we have observed that hallucination detection benchmarks have evolved from manual methods to more sophisticated modern approaches that better reflect the natural usage of LLMs. These benchmarks play a crucial role in ensuring the factual accuracy, coherence, and relevance of model-generated content, with ongoing research continuously improving these methods to address new challenges. Future research should continue to focus on enhancing hallucination detection benchmarks, optimizing mitigation strategies, reducing costs, and addressing specialized problems across multiple domains. This will require refined data, improved model performance, and the exploration of more effective methods to ensure the accuracy and reliability of model-generated content.