Advanced Methods in Natural Language Processing

Session 10: Hallucinations & Agent

Arnault Gombert

June 2025

Barcelona School of Economics

Introduction

Today's session will explore:

- 1. Main Limitations of LLMs
 - Static Data / Outdated Knowledge: LLMs cannot update themselves in real-time, leading to outdated or incomplete answers.
 - Hallucinations: Confidently generating information that is false or misleading.

Note: We will treat outdated knowledge as part of the hallucinations.

2. From Hallucinations to Agents

- How multi-step reasoning and tool use (agents) can reduce hallucination risks.
- The ReAct framework: enabling LLMs to reason and act for more robust and grounded outputs.

Understanding Hallucinations in LLMs

Defining the Phenomenon:

Hallucinations in LLMs refer to instances where models generate incorrect, fabricated, or nonsensical information that is not supported by their input data or real-world facts.

Why It Matters:

- Impacts model reliability and trustworthiness.
- Challenges in deploying LLMs for critical applications.
- Raises concerns about misinformation dissemination.

Objective: To explore the underlying causes of hallucinations and introduce strategies to mitigate these effects, enhancing the fidelity and dependability of LLM outputs.

Examples of Hallucinations

- 1. **Outdated Information:** The model confidently states outdated facts about a country's current president or economic status, reflecting training data that is not up-to-date.
- 2. **Factual Inaccuracies:** A model generates a detailed biography of a fictional scientist, including their contributions to a non-existent field of study.
- 3. **Nonsensical Outputs:** In response to a prompt about climate change, the model conflates unrelated historical events, creating a confusing narrative.
- 4. **Misinformation:** The model asserts a false cause-and-effect relationship between unrelated global health events, potentially spreading misinformation.

Insight: These examples underscore the challenge of ensuring that LLMs produce accurate and sensible outputs, particularly when dealing with complex or nuanced topics.

who is the president of the united states

The current President of the United States is **Joe Biden**. He took office on January 20, 2021, as the 46th president. If you want more details about him or the presidency, just let me know!

日 む 匂 ら い い~



ChatGPT Screenshot from May 20th, 2025

Examples of Hallucinations

Ρ

summarise this article https://www.nytimes.com/2023/03/11/technology/chatgpt-prompts-to-avoid-content-filters.html

The article discusses how ChatGPT, an AI language model, is being used to circumvent content filters that are designed to detect and block certain types of content. By generating prompts that are less likely to trigger these filters, ChatGPT can be used to create content that might otherwise be censored. While this technology could be useful for those seeking to express themselves freely, it also raises concerns about the potential misuse of AIgenerated content, as well as the difficulty of regulating such content in the future. The article concludes by noting that there is still much to learn about the impact of AI language models like ChatGPT on the way we communicate online.

Wikipedia

ChatGPT summarizing a non-existent New York Times article based on a fake URL.

Examples of Hallucinations

Default (GPT-3.5)

 Translate the Lojban into English: i lo cevzda cu bapli lo nalcurnu i lo cerni cu zvati lo donri i lo cerni cu se cmene lo cevzda lo tcika. i lo cevzda cu cusku lu i. i voi li re no li .i lo cevzda cu cusku lu i. i voi li no li solo .o voi lo voi li mu li .i lo cevzda cu cusku lu i. i avoi in no li solo. voi lo voi li mu li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo mu li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no li .i lo cevzda cu cusku lu i.i solo no li solo no solo no li no li mu li .i lo cevzda cu cusku lu i.i .e lo no solo no li .i lo no solo no li no li mu li .i lo cevzda cu cusku lu i.i .e lo no solo no li no li .i no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cu cusku lu i.i .e lo no solo no cevzda cusku lu i.i .e cusku lu i.i .e cusku lu i.i .e lo no solo no cevzda cusku lu i.i .e cusku lu i.i .e cusku lu i.i .e cusku lu i.i .e cusku lu i.i .e

Please note that Lojban is a constructed language with a unique grammar and structure, so the translation may not perfectly capture the nuances of the original text.

Wikipedia

A translation on the Vicuna LLM test bed of English into the constructed language Lojban, and then back into English in a new round

Navigating the Impacts:

- Erosion of User Trust: Reliance on LLMs diminishes as output reliability comes into question.
- Operational Risks: Inaccurate model outputs can lead to flawed decision-making in critical applications, such as healthcare or legal advice.
- Ethical and Societal Concerns: The spread of misinformation can have widespread consequences, affecting public opinion, policy making, and individual behavior.

Call to Action: Addressing hallucinations in LLMs is not just a technical challenge but a necessary step towards responsible AI development and deployment.

Why Do LLMs Hallucinate?

Data-Related Causes:

- Source-Reference Divergence: Heuristic data collection or inherent nature of NLG tasks can promote ungrounded text generation.
- Outdated or Incomplete Knowledge: LLMs are trained on static data and cannot update themselves, leading to outdated or incomplete responses.
- Examples:
 - Asserting false cause-and-effect links in global health events.
 - Stating outdated facts about political or economic developments.

Insight: Addressing these hallucinations requires grounding LLM outputs with updated, external knowledge sources.

Model-Related Causes:

- Statistical Inevitability: Imperfect models that maximize training likelihood can produce hallucinations, especially without active learning corrections.
- Novelty vs. Usefulness Tradeoff: Focusing too much on novelty can create original yet inaccurate responses, while prioritizing usefulness might lead to repetitive outputs.
- Encoding and Decoding Errors: Incorrect correlations or attention patterns in encoders and decoders can lead to hallucinations.
 Example: Top-k sampling or high temperature can introduce more hallucinated content.
- Examples:
 - Generating detailed biographies for fictional scientists.
 - Creating elaborate but untrue stories about climate change.

Mitigating Hallucinations

Introduction: Mitigating Hallucinations in LLMs

1. Prompt Engineering

- Crafting prompts that guide the LLM to generate grounded and accurate answers.
- Using structured prompts, examples, and constraints to reduce ambiguity.

2. Retrieval-Augmented Generation (RAG)

- Incorporating external, up-to-date knowledge sources to ground responses.
- Fusing retrieval and generation to produce factually accurate answers.

3. Multi-LLM Approaches: LLM as a Judge

 Using a second (or more !) LLM to validate or critique outputs, enhancing quality control.

11

Enhanced Robustness and Reliability:

 Multi-agent systems can cross-verify information among different models, reducing the likelihood of accepting hallucinated content as accurate.

Diverse Perspectives and Specializations:

 Leveraging specialized agents for different domains or tasks can provide a broader range of knowledge and viewpoints, mitigating the risk of content fabrication.

Dynamic Feedback and Correction:

 Agents can provide real-time feedback and corrections to each other's outputs, ensuring more accurate and grounded responses.

LLM-as-a-Judge: Concept Overview

- Using a LLM to evaluate other LLM outputs.
- A scalable, consistent alternative to human evaluations.

Key Insights:

- Zheng et al. (2023) shows LLM judges can align with human judgments over 80% of the time.
- Effective for assessing *helpfulness*, *coherence*, *factual accuracy*.

Applications:

- Evaluating chatbot responses.
- Comparing outputs from different LLMs.
- Improving model performance through automated feedback.





LLM-as-a-Judge: Evaluating a Single Output

Scenario:

- Input Prompt: "Explain the significance of the Higgs boson discovery."
- LLM Response: "The Higgs boson discovery confirmed the existence of the Higgs field, which gives mass to particles."

Evaluation Prompt:

 "Rate the accuracy and completeness of the following response on a scale from 1 (poor) to 5 (excellent)."

LLM-as-a-Judge Assessment:

- Score: 4
- Justification: Accurate but lacks detail on experimental methods and implications.

[System]

Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin your evaluation by providing a short explanation. Be as objective as possible. After providing your explanation, please rate the response on a scale of 1 to 10 by strictly following this format: "[[rating]]", for example: "Rating: [[5]]".

[Question] {question}

[The Start of Assistant's Answer] {answer} [The End of Assistant's Answer]

Credit: Zheng et al. (2023)

LLM-as-a-Judge: Comparing Two Outputs

Scenario:

- Input Prompt: "Summarize the causes of World War I."
- Response A: "World War I was caused by a complex set of alliances and the assassination of Archduke Franz Ferdinand."
- Response B: "The war began due to imperialism, militarism, alliances, and nationalism, triggered by the assassination of Archduke Franz Ferdinand."

Evaluation Prompt:

 "Which response provides a more comprehensive summary of the causes of World War I?"

LLM-as-a-Judge Decision:

- Preferred Response: B
- Justification: Includes broader range of causes, offering a more complete overview.

LLM-as-a-Judge: Comparing Two Outputs

[System]

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user question displayed below. You should choose the assistant that follows the user's instructions and answers the user's question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible. After providing your explanation, output your final verdict by strictly following this format: "[[A]]" if assistant A is better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

[User Question] {question}

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]

[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]

Credit: Zheng et al. (2023)

Function Calling as a Guardrail

Motivation:

- LLMs are powerful text generators, but they may hallucinate or produce outdated or fabricated information.
- Retrieval-Augmented Generation (RAG) mitigates this by providing external data, but sometimes structured external interactions are needed.

Function Calling:

- LLMs call functions or APIs to retrieve precise information or perform specific tasks instead of guessing.
- This blends LLM's natural language generation with factual, real-time data—bridging the gap between static model knowledge and dynamic external sources.

What is Function Calling?

- Instead of generating final answers from scratch, the LLM can call an external function or API for reliable information.
- The LLM generates structured requests (e.g., JSON), which are sent to external systems.
- Results from the function are then integrated into the LLM's final response.

Key Advantage:

 Reduces hallucinations by relying on external data sources for factual accuracy. Ideal Use Cases:

- Factual Retrieval: E.g., weather updates, stock prices, or current events from APIs.
- Calculations and Conversions: LLM delegates complex math or unit conversion tasks to reliable calculators.
- Database Queries: Querying structured databases (SQL, NoSQL) for precise answers.
- Other External Tools: Any structured task requiring accurate external data (e.g., flight status, product inventory, custom APIs).

Outcome: Enables LLMs to deliver factually accurate, real-time responses grounded in reliable sources.

Scenario:

- Prompt: "What's the weather in Paris?"
- **Goal:** Provide real-time, accurate weather data.

Function Flow:

- 1. **LLM Response:** Generate a structured API call request to a weather service.
- 2. External API: Weather service processes the request and returns current data.
- 3. LLM Final Output: Incorporates the API response into a natural language answer.

Example Implementation

```
def get_weather(location: str, unit: str = "celsius") -> dict:
    # In real implementation, you'd use an API like OpenWeatherMap
```

```
mock_weather = {
    "Paris": {"celsius": "18°C, sunny",
        "fahrenheit": "64°F, sunny"},
    "New York": {"celsius": "22°C, partly cloudy",
        "fahrenheit": "72°F, partly cloudy"}
}
output = {"weather":
        mock_weather.get(location, {}).get(unit, "Data not available")}
return output
```

Integration: This function would be called by your external system or wrapper, and its output would be fed back to the LLM to generate a user-friendly response.

Implementation Example: Function Calling with OpenAI API

Code Snippet

```
from openai import OpenAI
client = OpenAI()
functions = [
    {
        "name": "get_weather",
        "description": "Retrieve current weather info for a city",
        "parameters": {
            "type": "object",
            "properties": {
                "location": {"type": "string", "description": "City name"},
                "unit": {"type": "string", "enum": ["celsius", "fahrenheit"]}
            },
            "required": ["location"]
        }
   }
```

Implementation Example: Function Calling with OpenAI API

Code Snippet

```
response = client.chat.completions.create(
   model="gpt-4o",
   messages=[
       { "role": "system".
          "content": "You can call APIs for accurate weather data."},
       {"role": "user".
         "content": "What's the weather in Paris?"}
   ],
   functions=functions,
   function_call={"name": "get_weather"}
print(response.choices[0].message)
```

Output: A structured JSON request for the external weather API, reducing guesswork and improving accuracy.

Agents

From Function Calling to Agents:

- Function Calling Limitations: When tasks involve multiple functions, LLMs need to plan which functions to call and in what order. - LLMs alone struggle to coordinate multiple external calls in a structured way.
- Beyond Single-Turn QA: Some queries require multi-step reasoning, intermediate steps, or tool usage (e.g., calculator, search API, database).

Agents:

- Extend LLMs to **plan**, **decide**, **and execute** actions in sequence.
- Enable dynamic interaction with external tools and data sources to produce more reliable and grounded answers.

Why Do We Need Agents?

LLMs Alone:

- Typically stateless and limited to single-turn responses.
- Cannot track progress or adapt plans based on intermediate outcomes.

Agents to the Rescue:

- **Multi-Step Reasoning:** Agents plan and adapt their steps to reach a final answer.
- **Tool Use:** Dynamically select and use external tools (APIs, calculators, knowledge bases).
- Memory Adaptation: Retain intermediate results to refine their final response.
- Hallucination Mitigation: By verifying each step, agents reduce error propagation.

Traditional LLMs:

- Stateless: Each query is independent.
- Direct Output: No explicit planning or verification.
- Single-Pass Generation: Prone to hallucinations in complex tasks.

Agents:

- Iterative Reasoning: Plan actions based on intermediate results.
- **Tool Integration:** Use external APIs, retrieval systems, calculators, etc.
- **Environment Interaction:** Can adapt to dynamic data and evolving tasks.

Takeaway: Agents extend the capabilities of LLMs to tackle more complex and realistic scenarios with better accuracy and reliability.

What is ReAct?

- ReAct = Reason + Act
- It's a framework where an LLM alternates between thinking and acting to solve complex queries.

Key Features:

- Self-Ask: LLM formulates sub-questions to refine its reasoning.
- Use Tools: LLM interacts with external APIs, databases, or calculators.
- Verification: Observes intermediate results and adapts its plan to ensure accurate answers.

Why It Matters: ReAct empowers LLMs to tackle complex tasks in an iterative, grounded, and structured manner.

User Query: "What's the capital of France's highest mountain?"

ReAct Agent Process:

- Step 1 Think: Recognizes that it needs to know the highest mountain in France first.
- Step 2 Act: Calls a knowledge API to find out: "What's the highest mountain in France?"
- 3. Step 3 Observe: Receives: "Mont Blanc."
- 4. Step 4 Think: Now find out the capital of Mont Blanc's region.
- Step 5 Act: Calls the API to find the capital of Haute-Savoie (Mont Blanc's department).
- Step 6 Compose Answer: The final answer: "The capital is Annecy, Haute-Savoie."

 $\begin{array}{c} \textbf{Outcome:} \ \text{An accurate and well-grounded answer through reasoning and} \\ \text{external tool use.} \end{array} \begin{array}{c} 29 \end{array}$

How ReAct Works:

- 1. **Think (Reason)** The LLM breaks down the task and identifies sub-questions or steps.
- 2. Act (Use Tool) The LLM decides to call an external tool (API, calculator, search engine) to get accurate data.
- 3. **Observe (Get Result)** Receives the result from the tool and integrates it into its reasoning.
- Repeat Until Robust Iteratively repeats reasoning and action until a comprehensive and verified answer is reached.

Insight: ReAct turns the LLM from a static responder into a dynamic **agent** that iteratively plans, executes, and verifies to minimize hallucinations.

Python Example:

Code Snippet (LangChain)

```
from langchain.agents import initialize_agent, AgentType
from langchain.tools import Tool
from langchain.llms import OpenAI
# Define your external tool (e.g., search engine)
tools = [
    Tool(
        name="search",
        func=lambda q: f"Mock search result for: {q}",
        description="Use to search the web or knowledge base."
)
```

Implementation Example: ReAct Agent with LangChain

Python Example:

Code Snippet (LangChain)

```
# Initialize the ReAct agent
llm = OpenAI(model="gpt-4o")
agent = initialize_agent(
    tools,
    llm,
    agent=AgentType.REACT,
    verbose=True
)
# Ask the agent a question
result = agent.run("What's the capital of France's highest mountain?")
print(result)
```

Outcome: The ReAct agent dynamically calls external tools to iteratively build an accurate answer.

Agents Can Fail in Many Ways:

Planning Failures:

- Invalid tools or incorrect parameters in tool calls.
- Plans that do not solve the intended task (goal failures).
- Errors in reflection (agent thinks it's done but isn't).
- Tool Failures:
 - Correct tool is used but it produces the wrong output.
 - Tool translation errors when generating executable commands.
 - Missing or inadequate tools for certain domains.

Efficiency Challenges:

• Agents may be slow, use too many steps, or be more expensive than simpler approaches.

Key Insight: Effective evaluation and domain-specific tool selection are essential for robust, efficient, and accurate agent-based solutions.

QA and **Takeaways**

Open Discussion

- Feel free to ask questions or share your thoughts about today's topics.
- Any insights, experiences, or perspectives you'd like to discuss are welcome.

Summary of Key Takeaways

- Hallucinations and Errors: LLMs can generate outdated or incorrect information, highlighting the need for robust mitigation.
- Function Calling: A practical way to reduce hallucinations by integrating real-world data sources (APIs, DBs).
- Agents: Go beyond single-turn QA with iterative planning, tool use, and verification to tackle complex tasks.
- ReAct Framework: Combines reasoning and acting to create adaptive, multi-step workflows.
- Limitations: Agents themselves can fail (planning, tool use, efficiency), so evaluation and continuous improvement are critical.
- **Outlook:** Agent-based approaches open new frontiers for LLMs, but they require careful design, testing, and domain adaptation.