



DeepLearning.AI

Module 4 introduction

LLMs and Text
Generation

Module Overview

LLM Foundations

Learn how transformers work and how to structure effective LLM calls.

Transformer Workflows and Grounding

Dive into the transformer architecture and build iterative LLM workflows grounded in retrieved information.

Advanced Techniques

Explore what works best in real-world RAG setups.

Hands-on project

Apply what you've learned to build a full RAG pipeline.

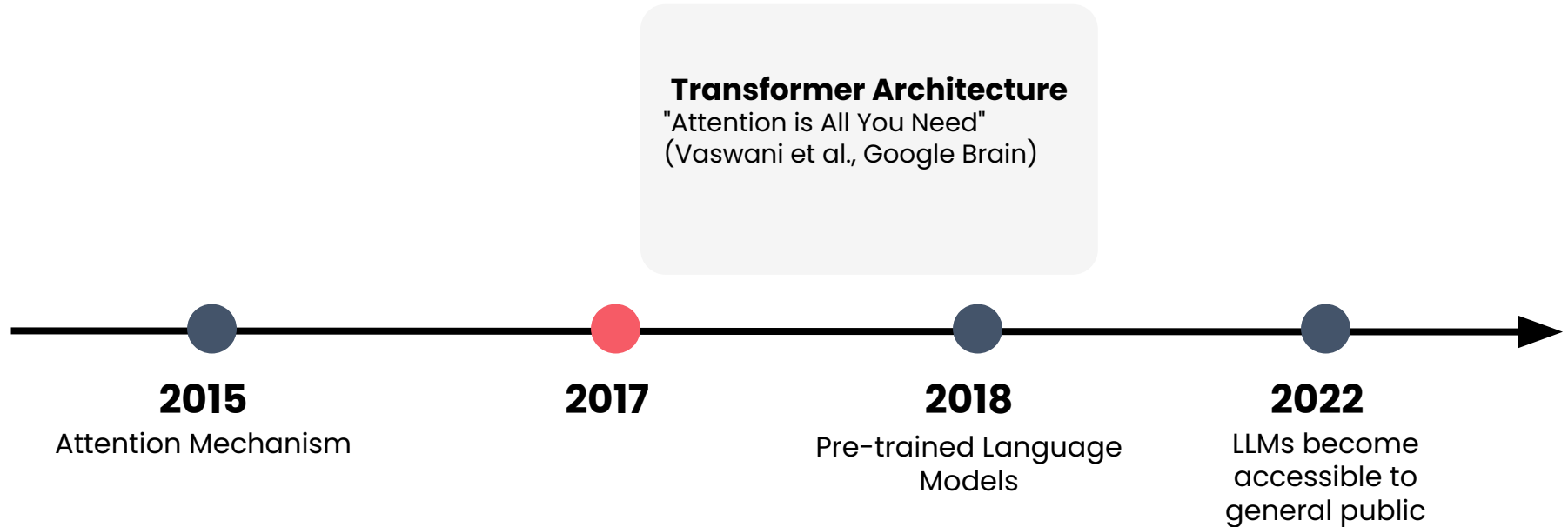


DeepLearning.AI

Transformer architecture

LLMs and Text
Generation

Origins of the Transformer



Encoder

- Processes the original text (e.g., German paragraph)
- Develops deep contextual understanding of the text's meaning

Used in embedding models for rich semantic representations

Decoder

- Uses the deep understanding from the encoder
- Generates new text in target language (e.g., English translation)

Most LLMs only include the decoder component, as they just care about text generation.

the brown dog sat next to the red fox

the brown dog sat next to the red fox

Dense Semantic Vector
"First Guess" of meaning

Position Vector
Position in prompt

the	[0.10, 0.21, 0.05, 0.12, 0.07]	[0.54, 0.19, 0.03, 0.17, 0.08]
brown	[0.55, 0.32, 0.91, 0.20, 0.44]	[0.28, 0.36, 0.82, 0.27, 0.51]
dog	[0.65, 0.40, 0.87, 0.22, 0.41]	[0.72, 0.34, 0.80, 0.29, 0.36]
sat	[0.11, 0.78, 0.23, 0.91, 0.65]	[0.33, 0.71, 0.26, 0.84, 0.59]
next	[0.05, 0.60, 0.18, 0.42, 0.87]	[0.88, 0.55, 0.23, 0.34, 0.82]
to	[0.09, 0.20, 0.04, 0.18, 0.15]	[0.11, 0.26, 0.06, 0.16, 0.19]
the	[0.10, 0.21, 0.05, 0.12, 0.07]	[0.52, 0.23, 0.07, 0.10, 0.05]
red	[0.57, 0.30, 0.89, 0.18, 0.43]	[0.08, 0.33, 0.88, 0.25, 0.46]
fox	[0.63, 0.38, 0.85, 0.21, 0.39]	[0.16, 0.42, 0.79, 0.19, 0.37]

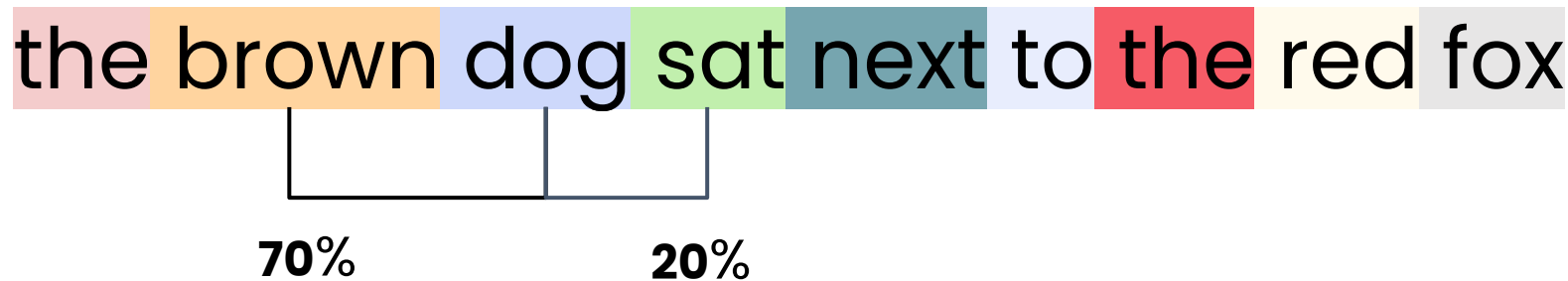
Semantic and **Position vectors** sent along together for processing

the brown dog sat next to the red fox

A diagram illustrating the concept of attention in a sequence model. The sentence "the brown dog sat next to the red fox" is shown with each word in a colored box. Below the words is a horizontal line with vertical tick marks corresponding to each word. From the word "the" (red box), lines connect to every other word in the sequence, representing its global attention.

Each token sees the **meaning** and the **position** of every other token

Attention is a fancy way of saying
"which other tokens should have the **biggest** impact on my
meaning"



The other 10% distributed across the other tokens.

Input Embeddings

The + pos_0	[0.12, 0.87, ..., 0.45]
Brown + pos_1	[0.33, 0.22, ..., 0.67]
Dog + pos_3	[0.50, 0.74, ..., 0.32]



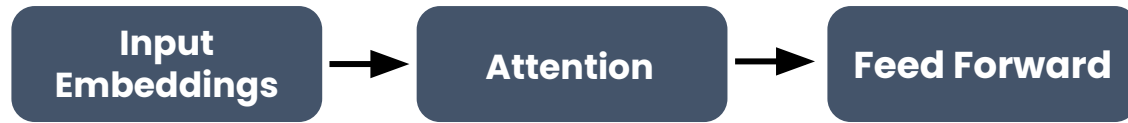
Transformer Layers



Smaller models may use 8 or 16 attention heads, but larger ones might use over 100.

Each head learns abstract patterns, not human defined rules

The Feed Forward Phase



the

brown

dog

[0.65, 0.40, 0.87, 0.22, 0.41]

sat

First Guess: "dog" is an animal

next

to

the

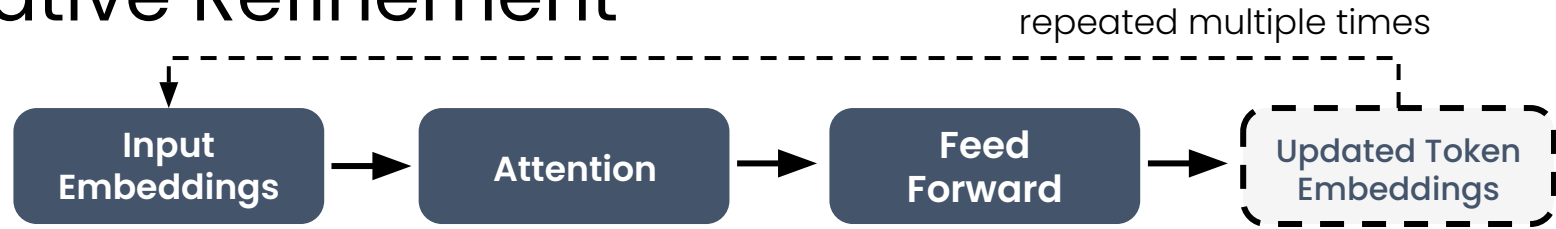
red

fox

[0.38, 0.19, 0.26, 0.41, 0.22]

Second Guess: "dog" is a brown animal that sat near a red fox

Iterative Refinement



First Guess

[0.65, 0.40, 0.87, 0.22, 0.41]

first attention + feed forward

**Second
Guess**

[0.38, 0.19, 0.26, 0.41, 0.22]

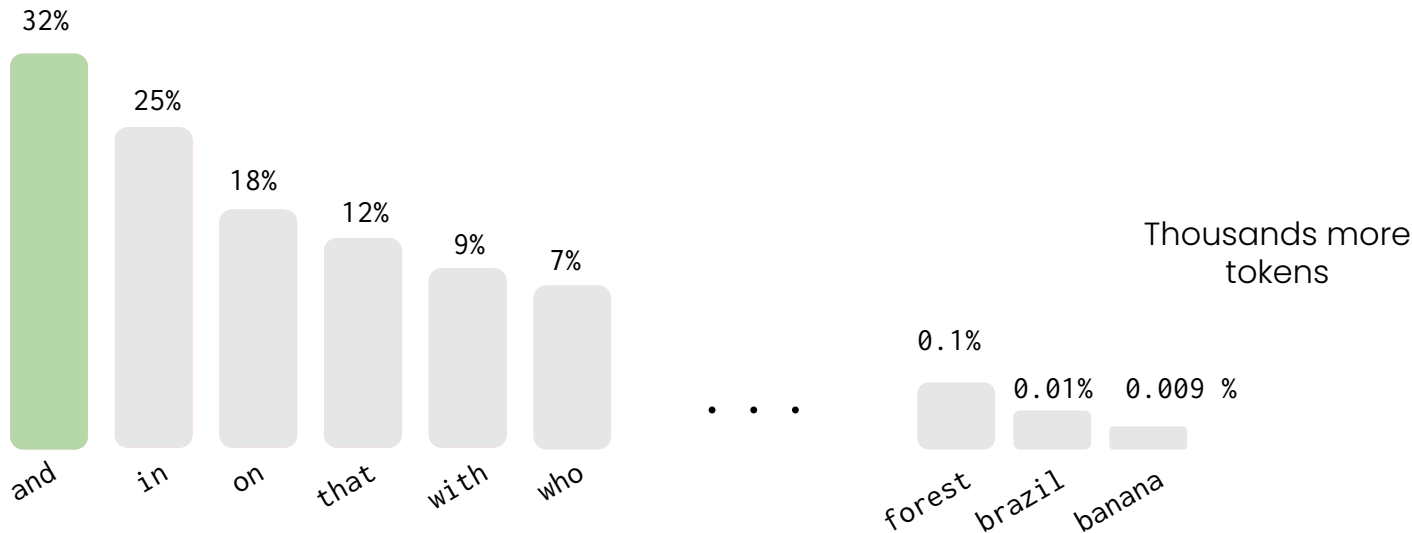
Understanding 'dog' as subject of sentence

[0.38, 0.22, 0.26, 0.58, 0.12]

What tokens are likely to come next?

Refined Embeddings

Model's
Vocabulary



the brown dog sat next to the red fox

and

the brown dog sat next to the red fox and...[EOS]

- To generate the next token, the model repeats the entire process from scratch
- This loop continues until it hits either token limit or end of completion token

The LLM's output tokens are de-tokenized and returned to the user as the final response.

the brown dog sat next to the red fox and wagged its tail

Conclusion

Why RAG works: LLMs can deeply understand information added to prompts through

- Attention mechanism processing
- World knowledge in feed forward layers

Inherent randomness remains

- LLMs may randomly ignore injected information
- Need to control randomness
- Must confirm LLM grounds answers in retrieved information

Computational expense

- Generating single tokens requires extensive processing
- Costs grow with prompt/completion length
- Each token must examine all others for context
- Most RAG system costs come from running transformers

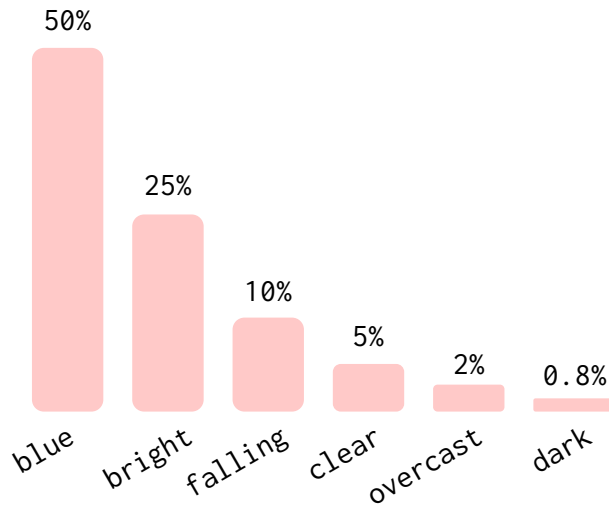


DeepLearning.AI

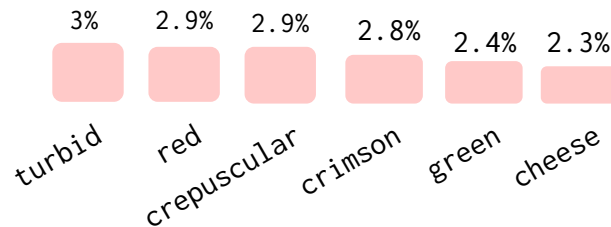
LLM sampling strategies

LLMs and Text
Generation

The sky is



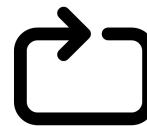
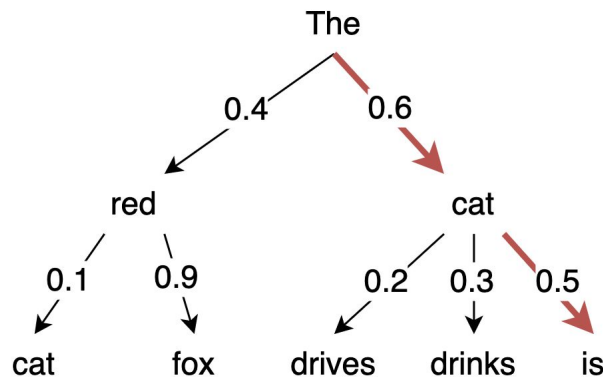
Peaked distribution
Model is "confident"



Flat distribution
Model is "uncertain"

Greedy Decoding

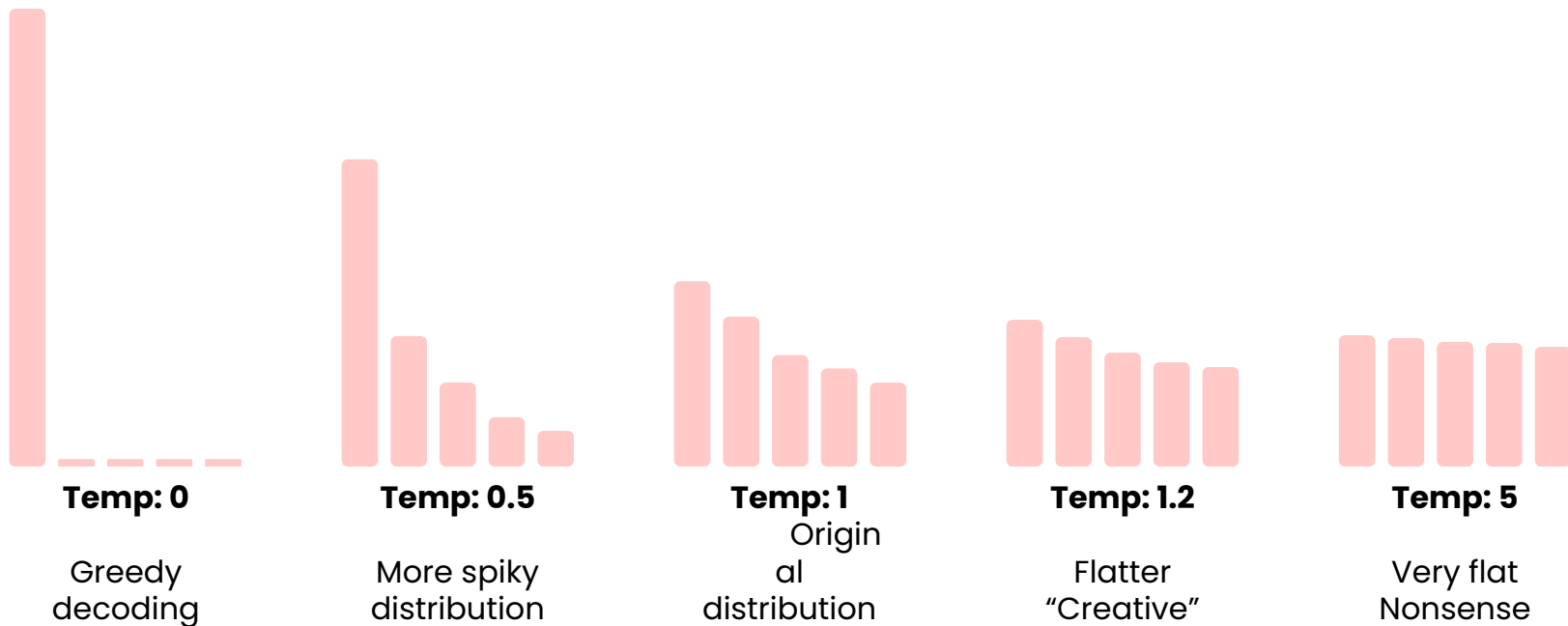
- Sampling strategy that always selects the token with the highest probability at each step of text generation.
- Deterministic
- Generic-sounding text
- Can get stuck in a loop
- Useful in code completion or debugging



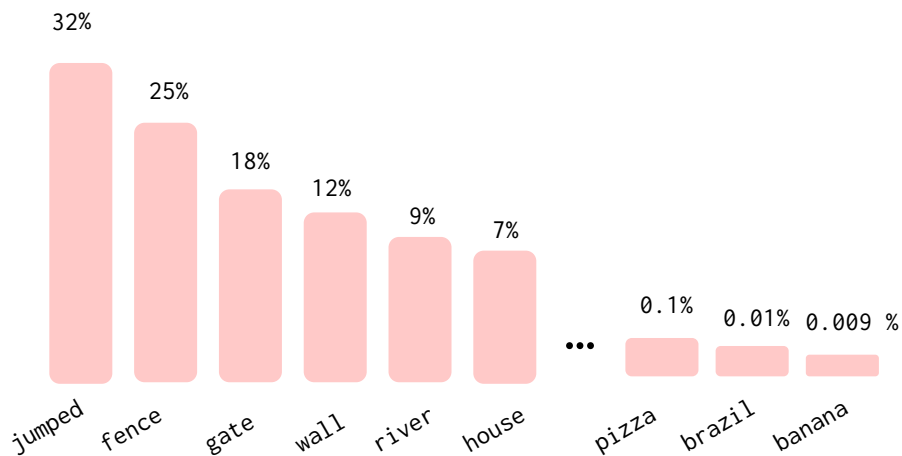
“which the data confirms,
which the data confirms,
which the data confirms...”

Temperature

Parameter that changes the shape of the distribution generated by the LLM



Advanced Token Sampling

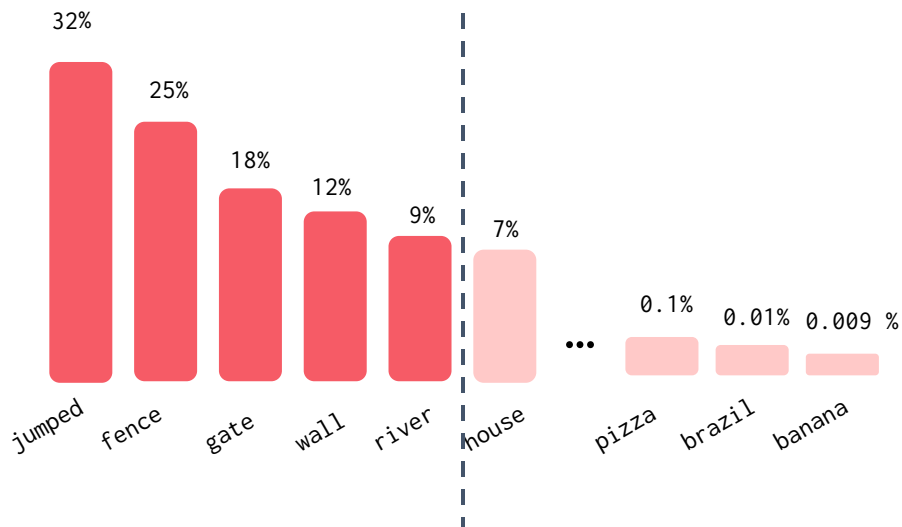


Top-K

Picks only from the k most likely tokens, ignoring the rest.

Advanced Token Sampling

Top-k: 5

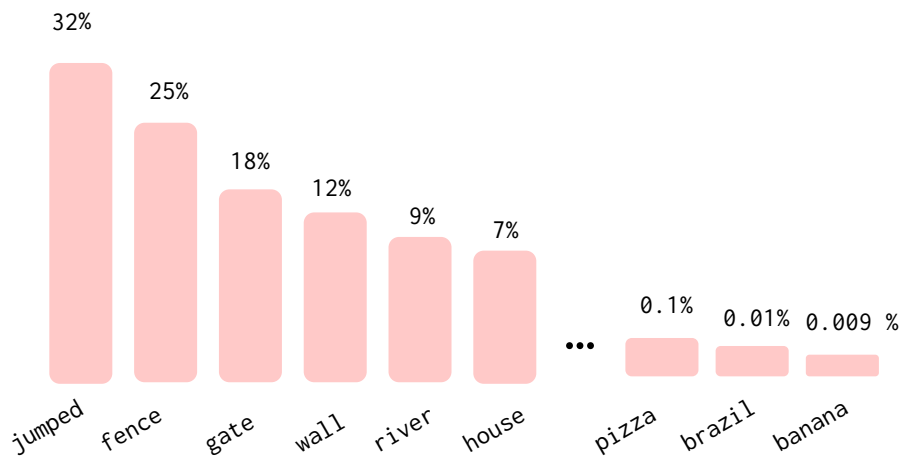


Top-K

Picks only from the k most likely tokens, ignoring the rest.

Top-K of 5 means only the 5 most likely tokens can be chosen

Advanced Token Sampling

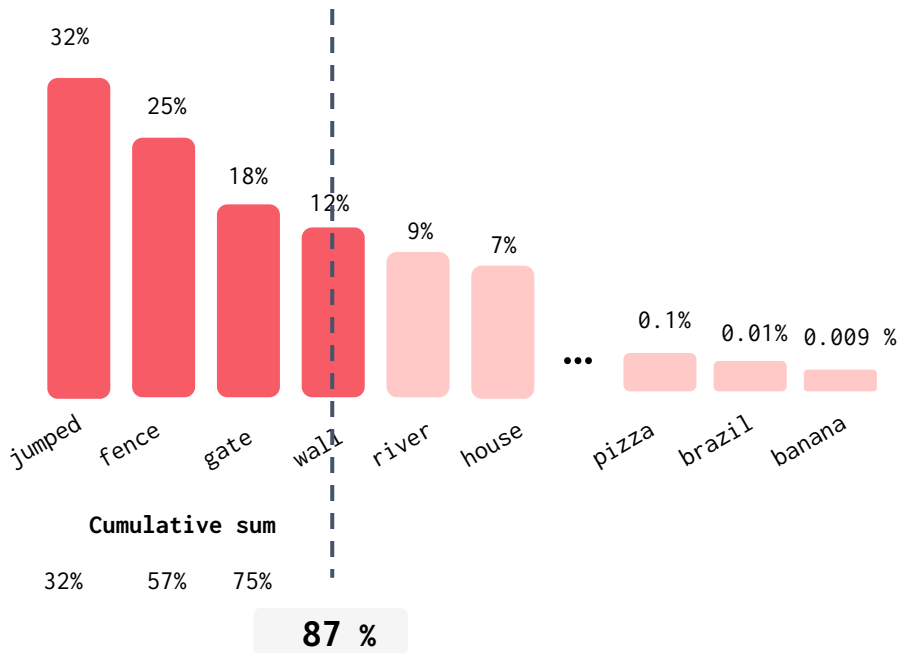


Top-P

Picks from tokens whose cumulative probability is below some threshold

Advanced Token Sampling

Top-p: 85%



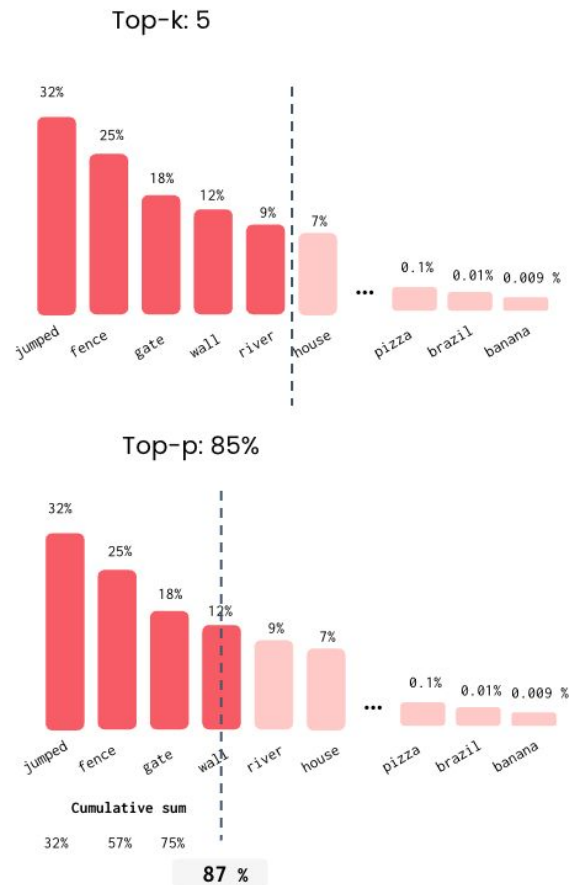
Top-P

Picks from tokens whose cumulative probability is below some threshold

Top-p of 85% includes tokens until their probability is above 85%

Top-k vs. Top-p

- Top-p is more dynamic than top-k
- Top-k always includes the same number of tokens regardless of the shape of the distribution
- Top-p includes more or fewer tokens depending on how “certain” the model is



Token Specific Strategies

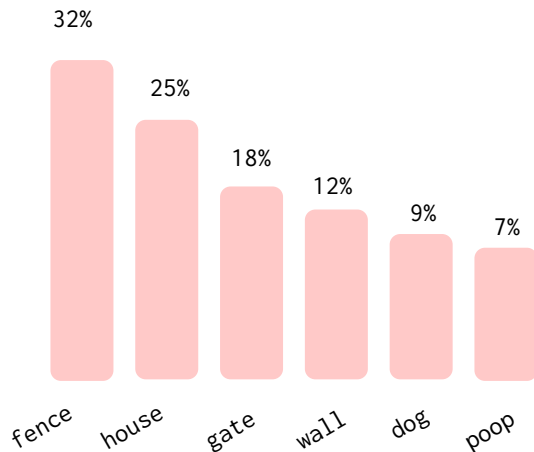
Repetition Penalties

Reduce the probability of already used tokens, discouraging repetition.

Helps prevent:

- Loops (e.g., "I think, I think...")
- Redundant sentence patterns
- Overuse of specific words

The brown **dog** jumped over the...



Token Specific Strategies

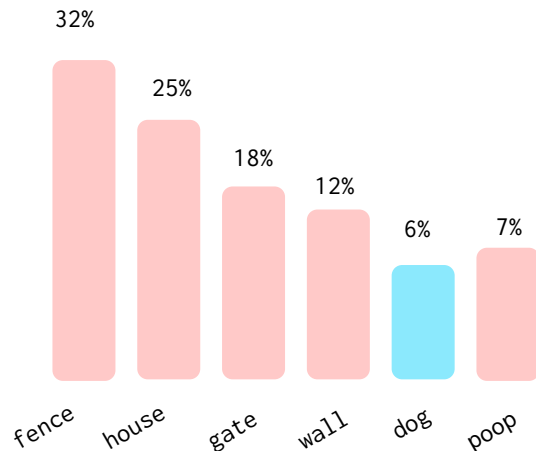
Repetition Penalties

Reduce the probability of already used tokens, discouraging repetition.

Helps prevent:

- Loops (e.g., "I think, I think...")
- Redundant sentence patterns
- Overuse of specific words

The brown **dog** jumped over the...



New probability
after penalty

Token Specific Strategies

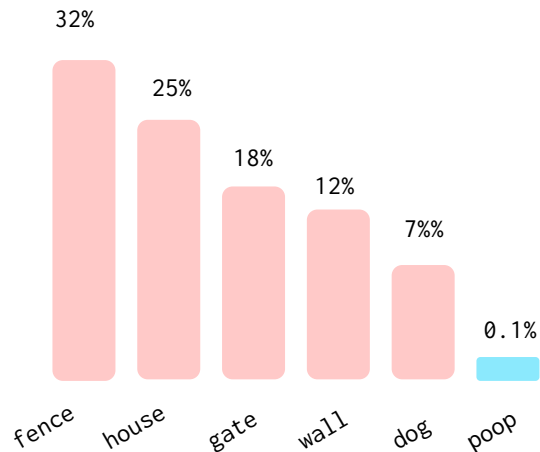
Logit Biases

Allow direct manipulation of token probabilities by adding or subtracting values from the model's raw calculated probabilities.

Biases can:

- Filter profanity
- Boost categories in a classifier LLM

The brown dog jumped over the...



"poop" permanently biased down

```
payload = {  
    "model": "meta-llama/Llama-2-70b-hf",  
    "temperature": 0.8,    "Slightly conservative in token choice"  
    "top_p": 0.9,    "Avoid choosing from far tail of distribution"  
    "repetition_penalty": 1.2    "Lightly penalize repeated tokens"  
}
```

Using Sampling Strategies

- Set temperature at top-p that fits your application's needs
- Code or factual domain → low temperature and top-p
- Creative domain → explore higher temperature and top-p
- Then add repetition penalties, logit biases, etc. as needs arise



DeepLearning.AI

Choosing your LLM

LLMs and Text
Generation

Important LLM Characteristics

Model Size

- Small models: 1 – 10 billion parameters
- Large Models: 100 – 500 billion and beyond
- Large models can be more capable, always more expensive

Cost

- Fixed cost per million tokens, sometimes different for input vs output
- New and larger models usually cost more

Context Window

- Maximum number of tokens an LLM can process, both prompt and completion
- You still pay per token

Latency and Speed

- Time to first token, tokens per second

Training Cutoff Date

- Last point in time in the model's training data
- Later is usually preferable

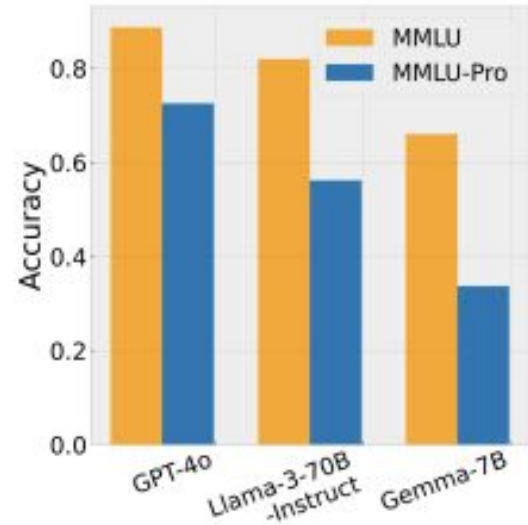
LLM Quality Metrics

- There are many types of quality
- There are many benchmarks that try to measure quality
- There's no single authoritative list
- Three categories: Automated, Human-scoring, LLM-as-a-judge



Automated Benchmarks

- Evaluated with code
- Common format is multiple choice test(s) on various subjects
- Example: MMLU covers 57 subjects from STEM to humanities
- Many automated benchmarks exist for wide variety of domains
















Human Evaluated Benchmarks

- Anonymous LLMs respond to a prompt, humans choose preferred response
- Uses ELO algorithm to create comparative leaderboards
- Example: LLM Arena is a popular host of human-graded rankings
- Captures nuanced quality factors automated benchmarks miss

Text Arena
View rankings across various LLMs on their versatility, linguistic precision, and cultural context across text.

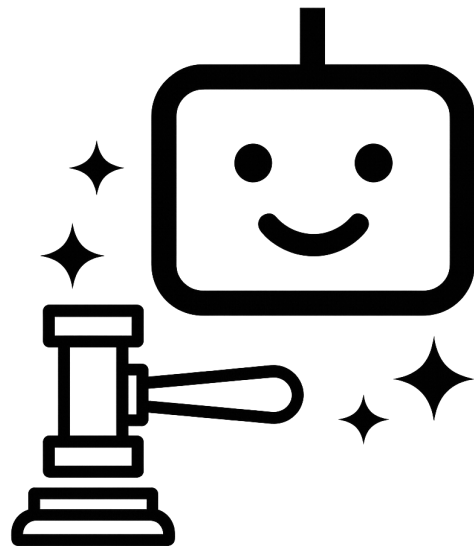
Last Updated: Jul 1, 2025
Total Votes: 3,144,445
Total Models: 254

Overall Default

Rank (UB) ↑	Model ↓	Score ↓	95% CI (±) ↓	Votes ↓	Organization ↓	License ↓
1	 gemini-2.5-pro	1463	+6/-5	14,062	Google	Proprietary
2	 o3-2025-04-16	1449	+3/-4	20,095	OpenAI	Proprietary
2	 chatgpt-4o-latest-20250326	1441	+4/-4	23,599	OpenAI	Proprietary
3	 gpt-4.5-preview-2025-02-27	1436	+4/-5	15,271	OpenAI	Proprietary
5	 claude-opus-4-20250514	1417	+4/-4	19,708	Anthropic	Proprietary
5	 gemini-2.5-flash	1415	+4/-5	19,451	Google	Proprietary
5	 deepseek-r1-0528	1413	+4/-5	12,396	DeepSeek	MIT
5	 gpt-4.1-2025-04-14	1411	+4/-5	17,456	OpenAI	Proprietary
6	 grok-3-preview-02-24	1408	+4/-3	25,067	xAI	Proprietary
10	 o1-2024-12-17	1399	+3/-4	29,038	OpenAI	Proprietary
10	 o4-mini-2025-04-16	1398	+4/-6	17,150	OpenAI	Proprietary
10	 deepseek-v3-0324	1396	+5/-4	20,171	DeepSeek	MIT
10	 qwen3-235b-a22b-no-thinking	1396	+5/-5	14,090	Alibaba	Apache 2.0

LLM-as-a-judge Benchmarks

- One LLM rates another's responses against reference answers
- Produces "win rate" for comparing LLM performance
- Upside: Cheap and flexible evaluation method
- Downside: Judges prefer their own model family (bias)



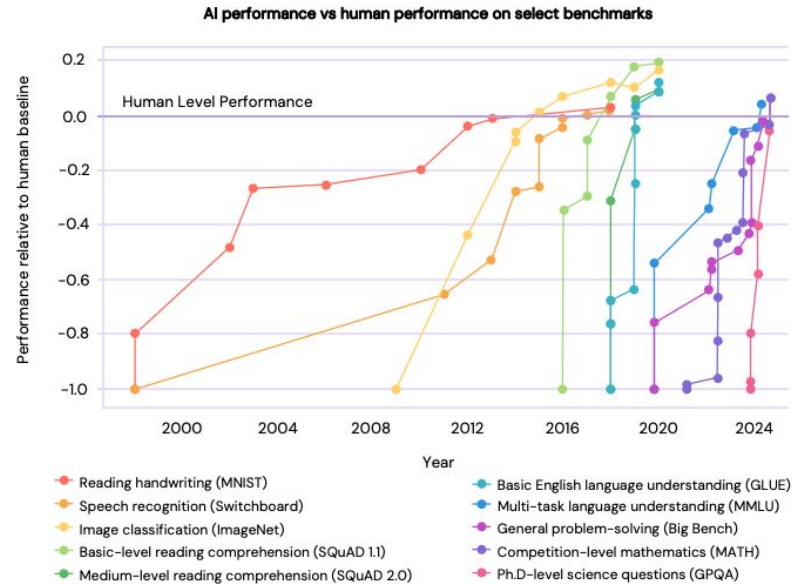
Good Benchmark Qualities

- **Relevant** to your project
- **Difficult** to help distinguish between high and low performing models
- **Reproducible** with stable outcomes between test runs and verifiable results
- **Align** with real-world performance
- Look out for data contamination



Benchmarks Over Time

- Benchmark scores start low
- Scores rapidly rise to be on par with human experts
- “Saturated” metrics no longer differentiate models
- New benchmarks need to be repeatedly introduced
- Main takeaway – Newer models usually outperform older ones



Source: International AI Safety Report – January 2025
Contains public sector information licensed under the Open Government Licence v3.0.



DeepLearning.AI

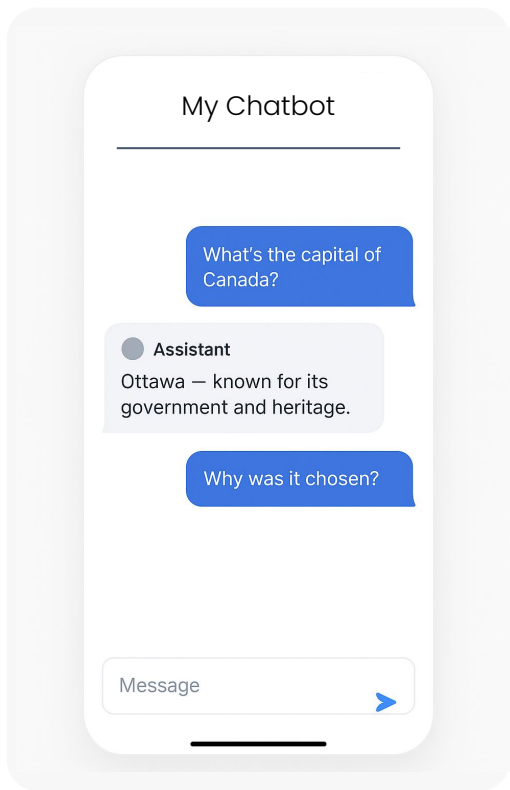
Prompt engineering: building your augmented prompt

LLMs and Text Generation

Messages Format

- **Content:** The text of the message
- **Role:** either "system", "user", or "assistant"
 - System: provides high-level instructions to influence LLM behavior
 - User: records prompts sent by users
 - Assistant: records responses previously generated by the LLM

```
{
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful assistant that answers questions. Provide factual, educational responses."
    },
    {
      "role": "user",
      "content": "What is the capital of Canada?"
    }
  ]
}
```

```
{  
  "messages": [  
    {  
      "role": "user",  
      "content": "What's the capital of Canada?"  
    },  
    {  
      "role": "assistant",  
      "content": "Ottawa — known for its government and heritage."  
    },  
    {  
      "role": "user",  
      "content": "Why was it chosen?"  
    }  
  ]  
}
```

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
```

```
You are a helpful AI assistant that provides accurate facts about countries<|eot_id|>
```

```
<|start_header_id|>user<|end_header_id|>
```

```
What's the capital of Canada?<|eot_id|>
```

```
<|start_header_id|>assistant<|end_header_id|>
```

```
Ottawa – known for Parliament Hill and its bilingual culture.<|eot_id|>
```

System Prompt

- System prompt provides high-level instructions on how the LLM should behave
- Include desired tone and procedures the LLM should follow

Example system prompt:
approximately **2,100 words**. About 4-5
pages

RELEASE NOTES

System Prompts

Claude 3.7 Sonnet

▾ Feb 24th, 2025

The assistant is Claude, created by Anthropic.

The current date is {{currentDateTime}}.

Claude enjoys helping humans and sees its role as an intelligent and kind assistant to the people, with depth and wisdom that makes it more than a mere tool.

Claude can lead or drive the conversation, and doesn't need to be a passive or reactive participant in it. Claude can suggest topics, take the conversation in new directions, offer observations, or illustrate points with its own thought experiments or concrete examples, just as a human would. Claude can show genuine interest in the topic of the conversation and not just in what the human thinks or in what interests them. Claude can offer its own observations or thoughts as they arise.

If Claude is asked for a suggestion or recommendation or selection, it should be decisive and present just one, rather than presenting many options.

High-Level Instructions

Fundamental behavior and knowledge cut-off

Claude's knowledge base was last updated at the end of October 2024. It answers questions about events prior to and after October 2024 the way a highly informed individual in October 2024 would....

Tone & Personality

Define how the LLM should communicate

Claude reasons through answers step by step.

Claude does not help with potentially harmful requests.

Claude responds in markdown.

Claude is intellectually curious and enjoys hearing what humans think.

Your own system prompt

Use these principles to construct your own system prompt

- Instruct your LLM to respond in great detail OR answer questions succinctly
- For RAG applications, tell the language model to:

Use only retrieved documents to answer prompts

Judge whether a document is relevant

Cite sources in its response

- System prompts are added to every prompt your LLM processes

Prompt Templates

You're ready to build your augmented prompt

- Use a well-considered template since prompts include many pieces of information
- Templates provide structure and decide where content gets injected

```
# System Instructions
{system_prompt} # behavioral guidance

# Conversation History
# if conversation_history exists
User: {message_1}
Assistant: {response_1}
User: {message_2}
Assistant: {response_2}
...

# Retrieved Information
[DOCUMENT 1]
{chunk_1_text}
Source: {chunk_1_source}

...

# User Prompt

User: {user_query}
```

Prompt Template

System Instructions

You are an useful assistant for geographic information . Only use retrieved documents to answer.

- Use only retrieved documents
- Cite sources as [DOC X]
- Admit if information is missing

Conversation History

User: What is the largest city in Canada?

Assistant: Toronto is Canada's largest city with a population of 2.9 million [DOC 1]

Retrieved Documents

[DOC 1] Ottawa is the capital city of Canada, located in the province of Ontario

[DOC 2] Ottawa became Canada's capital when Queen Victoria chose it in 1857

[DOC 3] The Parliament of Canada is located in Ottawa on Parliament Hill

Current Query

What is the capital of Canada?



DeepLearning.AI

Prompt engineering: advanced techniques

LLMs and Text
Generation

In Context Learning

Including example question-response pairs within your prompt to help the LLM learn a structure and tone to respond in.

Customer service bot example:

- Include previous customer requests
- Include high-quality responses to those request

Pattern learned from examples

Request:

How do I return an item?

Response:

Hi [Customer Name], for order #[Order ID]:
Print return label from your email and ship to us.
Questions? Contact help@mapleshop.ca



How do I return an item?



Hi Zain, for order #78234:
Print return label from your email and ship to us.
Questions? Contact help@mapleshop.ca

In Context Learning

You're adding extra information to the prompt

- Many examples it's called few-shot learning
- One example it's called one-shot learning
- A common way to implement this is by hard-coding examples

System Instructions

You are a helpful customer support assistant.

Example 1

Customer: How do I reset my password?

Agent: Click "Forgot Password" on the login page.

Example 2

Customer: Can I cancel my subscription?

Agent: Yes, from your account settings.

Index successful customer chats

Q: "How do I track my order?"

A: "To track your order, click the 'Order Status' button in your account dashboard."

Retrieve relevant information



Refund Policy



Customer Chats

Inject examples into prompt

`<refund policy>` `<chats>` How to track a refund from a cancelled order?

In context learning prompt template

```
{  
  "messages": [  
    {"role": "system", "content": "You are a helpful customer service assistant. "},  
    {"role": "system", "name": "example_user", "content": "How do I track my order?"},  
    {"role": "system", "name": "example_assistant", "content": "To track your order,  
click 'Order Status' in your dashboard. "},  
    {"role": "system", "content": "REFUND POLICY: Refunds take 5-7 days. Check status  
in 'Refunds' tab. "},  
    {"role": "user", "content": "How to track a refund from a cancelled order? "}   
  ]  
}
```

Encouraging Reasoning

Another powerful technique encourages the LLM to reason through prompts **step by step**.

Tell the LLM to '**think aloud**' about how to approach the problem before providing a final answer.



```
<scratchpad>
```

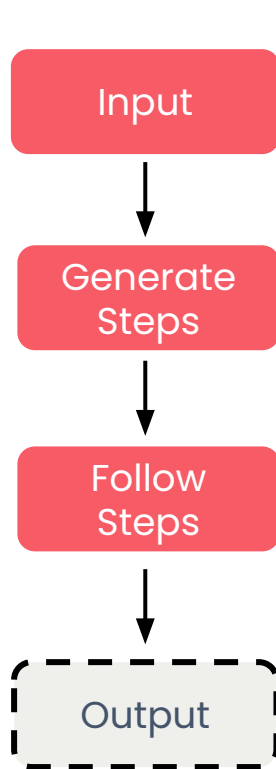
```
Option 1: Could be X because...
```

```
Option 2: Might be Y if...
```

```
Actually, Z makes most sense  
because...
```

```
</scratchpad>
```

Chain Of Thought



User Query

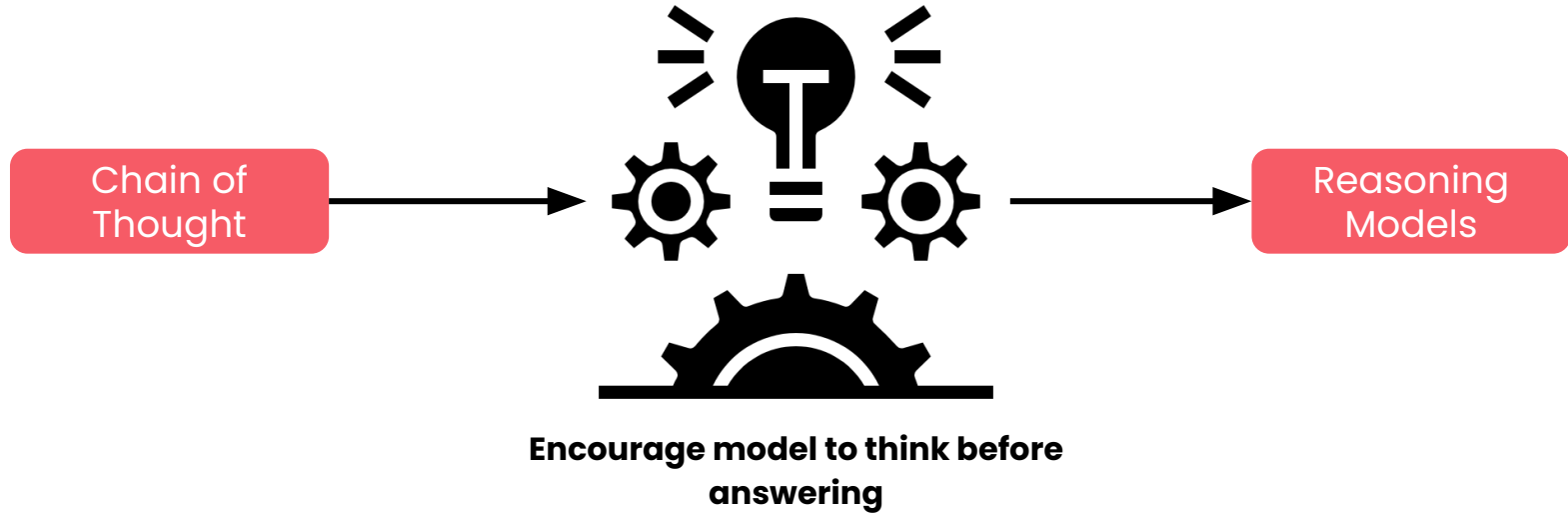
What's the capital of Canada? **Think step by step**

LLM response

1. I need to identify Canada's capital city
2. Canada is a North American country
3. The capital is located in Ontario province
4. Ottawa is the capital city of Canada

Therefore, **the capital of Canada is Ottawa.**

Reasoning Models



Reasoning Models

- Reasoning tokens boost accuracy but add cost
- These models run slower and are more expensive
- Worth it in RAG for better relevance and integration



Why do we wear sunglasses in the snow?

REASONING TOKENS:

Snow = white → reflects sunlight

Sunlight = bright → includes UV rays

Reflected light = intense on the eyes

Sunglasses reduce glare & UV → protect vision



We wear sunglasses in the snow to protect our eyes from the intense reflected sunlight.

Reasoning Models Warnings

Many prompting techniques don't work well on reasoning models.

- Struggle with in-context learning and example mixing
- Perform best with clear goals and strict formats
- Work well with full context and high-level guidance
- LLM providers will suggest the best ways to prompt new models

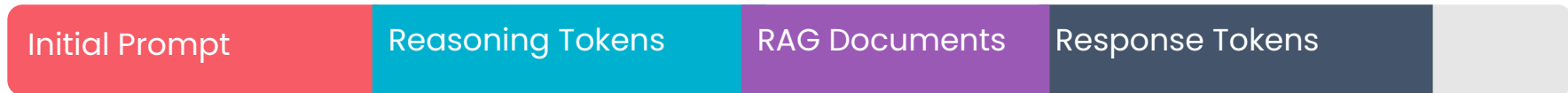
Context Window Management

Regular LLM use



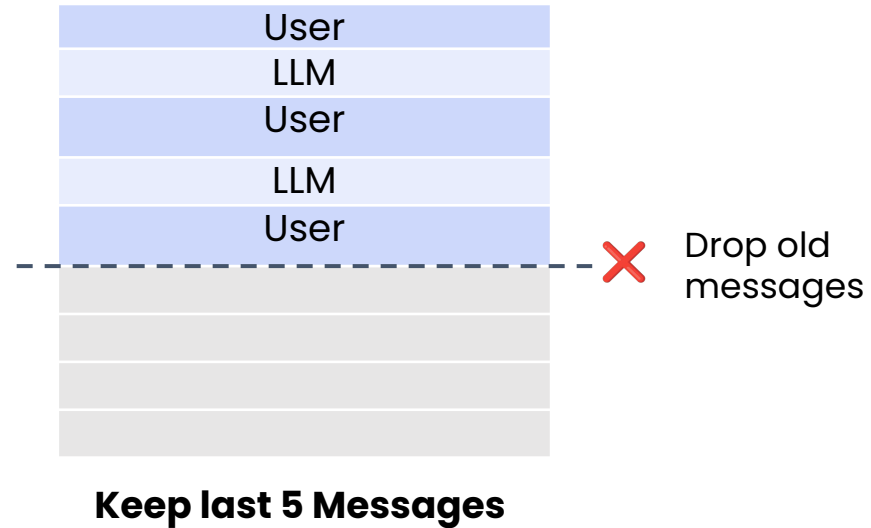
Advanced techniques **consume more context**

Reasoning Model + RAG



Management Strategies – Context Pruning

- With single-turn conversation skip prompt techniques if they add no value
- Use context pruning to manage long multi-turn prompts
- With reasoning models, drop reasoning tokens from chat history



Management Strategies

Include only chunks relevant to the latest question.

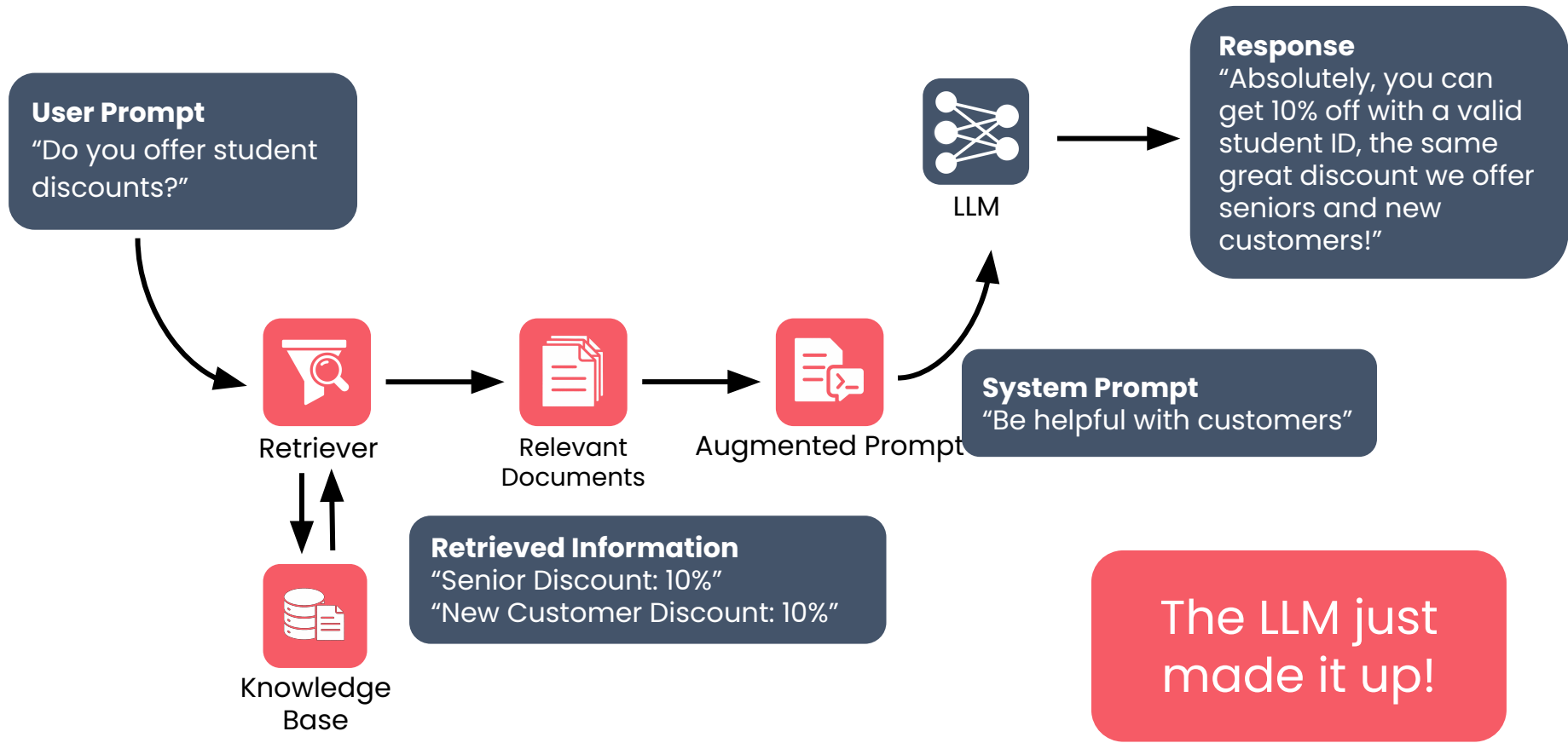
- Use long-context models for deeper, multi-turn conversations.
- Even with big context, keep prompts efficient.



DeepLearning.AI

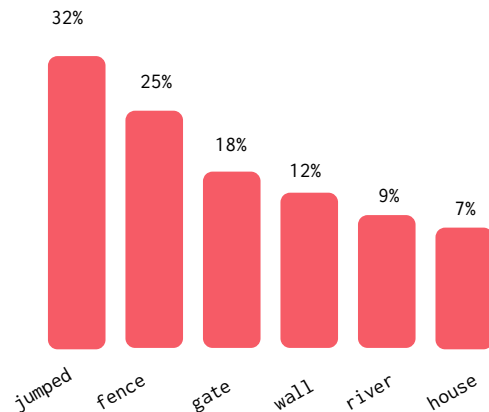
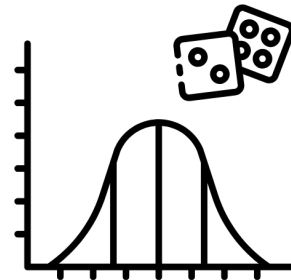
Handling hallucinations

LLMs and Text
Generation



Why LLMs Hallucinate

- LLMs produce probable text sequences
- Probable text is not always accurate, LLMs can't tell the difference



Why Hallucinations Matter

- **Inaccurate Information**

Bad on it's own

- **Hard to detect**

Sounds more plausible than nonsense

- **Erode Trust**

Occasional hallucinations detract even from mostly accurate systems

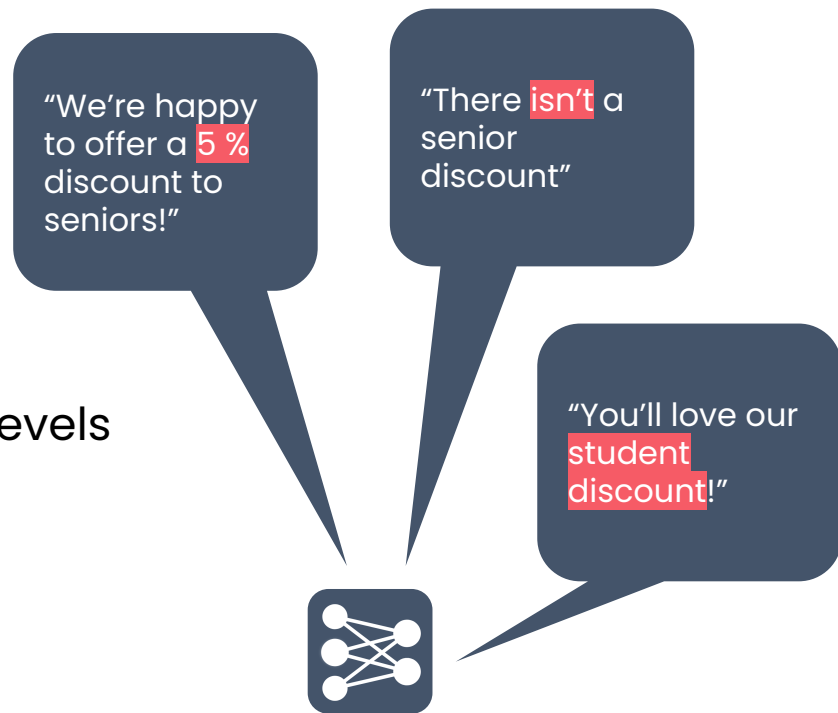
- **RAG can help**

Helps ground responses in retrieved information, but hallucinations still possible



Types of Hallucinations

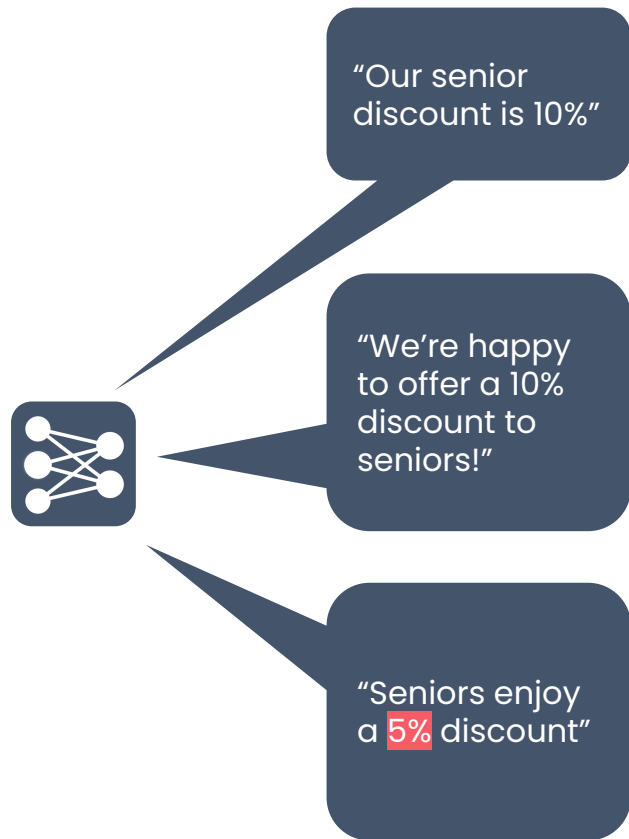
- Sometimes mess up small details
- Other times entirely invent facts
- Need to evaluate LLM output on many levels to ensure accuracy



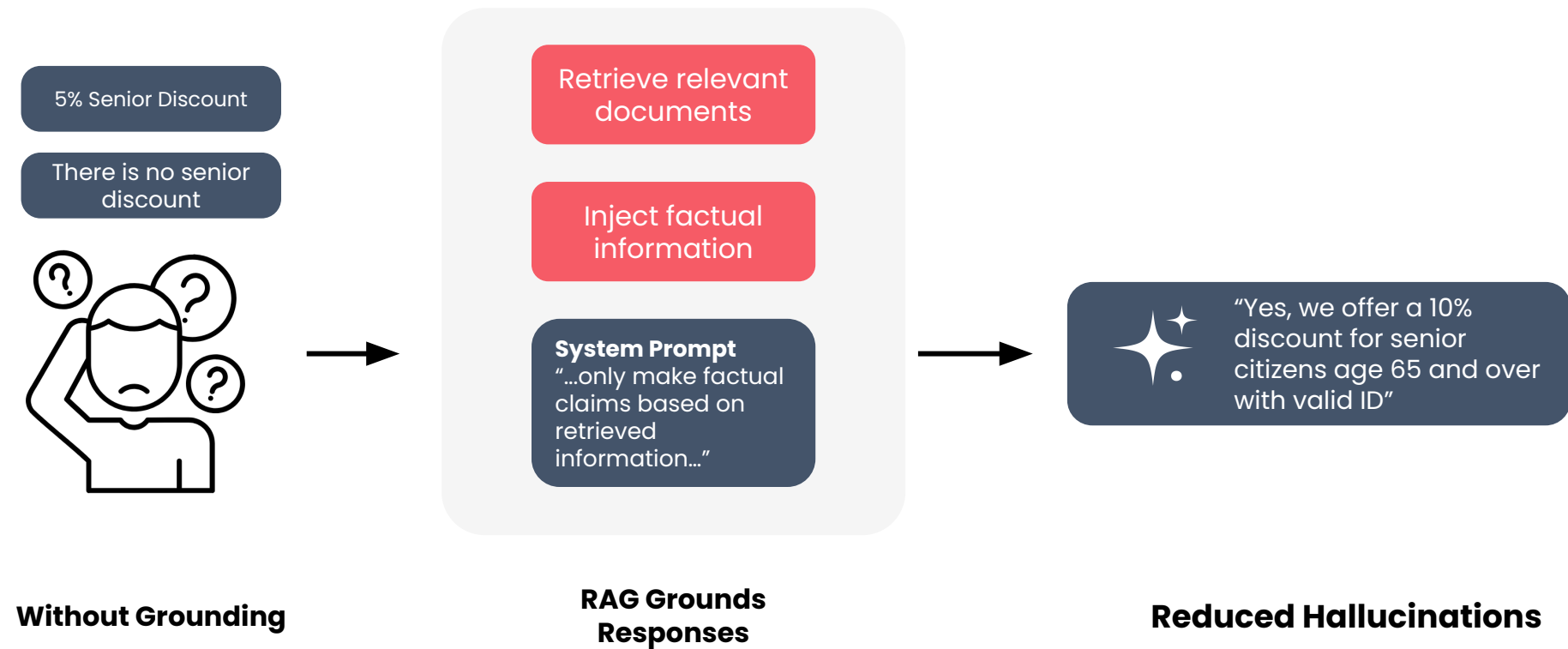
There's no perfect solution for hallucinations

Self-Consistency Methods

- Repeatedly generate responses to the same prompt and confirm consistency
- Factual inconsistencies can indicate hallucinations
- In practice, costly and unreliable



Reducing Hallucinations with RAG



Citation Generation

- Instruct the LLM to cite sources after each sentence or paragraph
- Increases likelihood of grounding and makes human verification easier
- LLMs can just hallucinate citations!

User Prompt

"Tell me about Lionel Messi's career achievements."

System Prompt

"Cite your sources at the end of each sentence using [1], [2], etc."

LLM response

Messi has won a record 8 Ballon d'Or awards [1]. He scored 672 goals for Barcelona [2]. Messi won the World Cup with Argentina in 2022 [3].

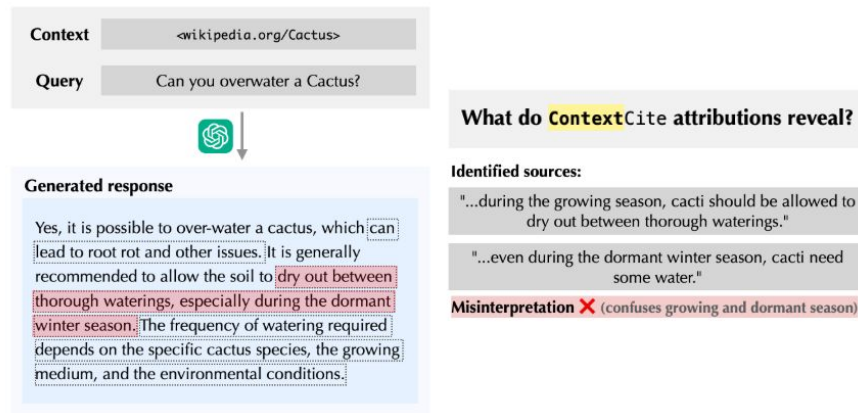
[1] Source: FIFA.com, "Ballon d'Or History"

[2] Source: FCBarcelona.com, "Club Records"

[3] Source: FIFA.com, "World Cup 2022"

ContextCite

- Attributes sentences in response to retrieved documents
- Tags sentences with supporting document
- Can be used to generate citations or as part of evaluation



Source: <https://gradientscience.org/contextcite/>

Evaluating Citation Quality in LLMs

ALCE Benchmark

- Pre-assembled knowledge bases + sample questions
- Tests RAG system responses on prepared prompts
- Evaluates responses across three metrics

Key Evaluation Metrics

Fluency

How clear and well-written is the generated text?

Correctness

How factually accurate is the generated content?

Citation Quality

How well do citations align with correct sources?



DeepLearning.AI

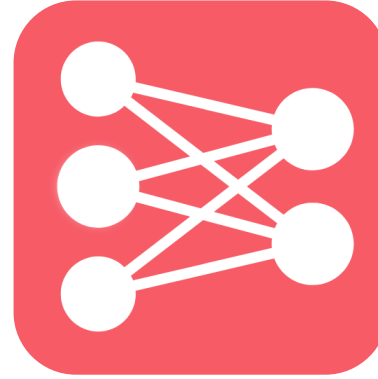
Evaluating your LLM's performance

LLMs and Text Generation



Retriever

Finds relevant
information



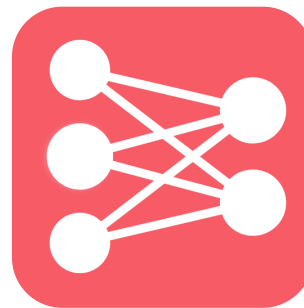
LLM

Constructs
response

LLM metrics should focus on the LLM's role

LLM Responsibilities

- Assume the retriever finds relevant information, possibly with a few irrelevant documents
- LLM should respond clearly, incorporate relevant information, cite sources, ignore irrelevant information
- These responsibilities are somewhat subjective
- LLM evals usually use other LLMs, e.g. those found in the RAGAS library



LLM



Response Relevancy

- Measures where response is relevant to user prompt, regardless of accuracy
- Evaluator LLM generates several new “sample prompts” that could have lead to the response
- Embed original and sample prompts to vectors and calculate cosine similarity
- Average similarity scores for final relevancy measure
- Doesn't check if information is factual, just if you can “work backwards” from generated response to original user prompt

ResponseRelevancy()



Faithfulness

- Measures whether response is consistent with retrieved information
- LLM identifies all factual claims in response
- More LLM calls to determine if claims are factually supported by retrieved information
- Percentage of supported claims is the “faithfulness”

Faithfulness()



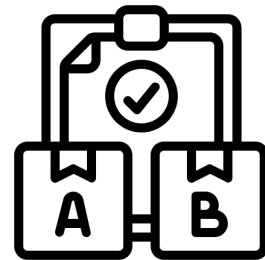
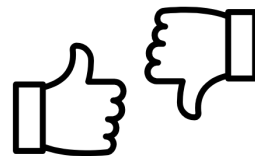
Other RAGAS metrics

- Other RAGAS metrics capture noise sensitivity, citations, etc.
- All metrics rely on LLM-as-a-judge at some point since the LLM's role is complex



Measuring LLM Performance Indirectly

- Measure LLM performance by measuring overall system performance.
- Collect system-wide feedback (e.g. thumbs-up/down) and A/B test changes to LLM
- Important to isolate changes to LLM to ensure you're capturing impact of only those changes

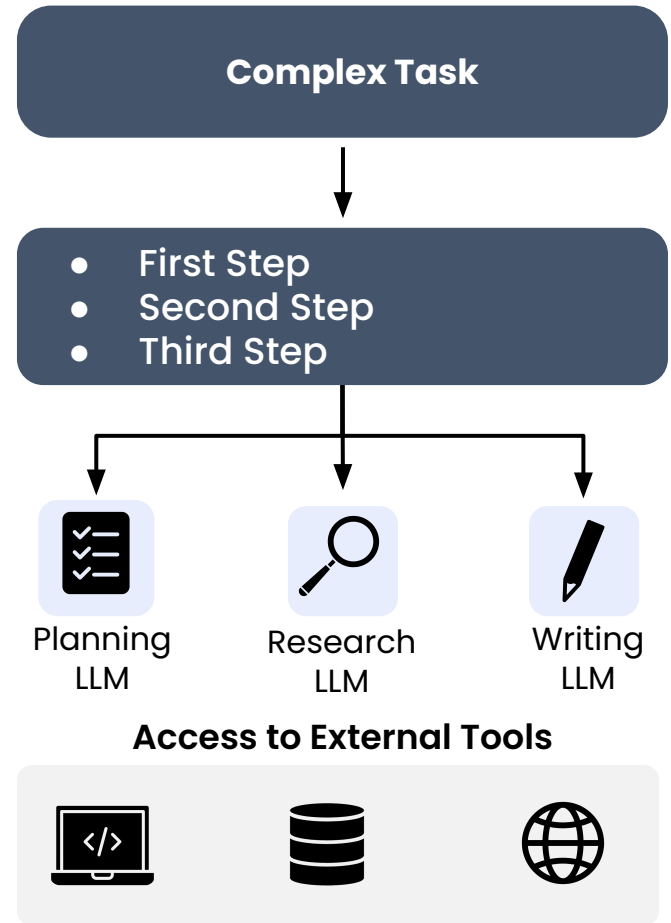




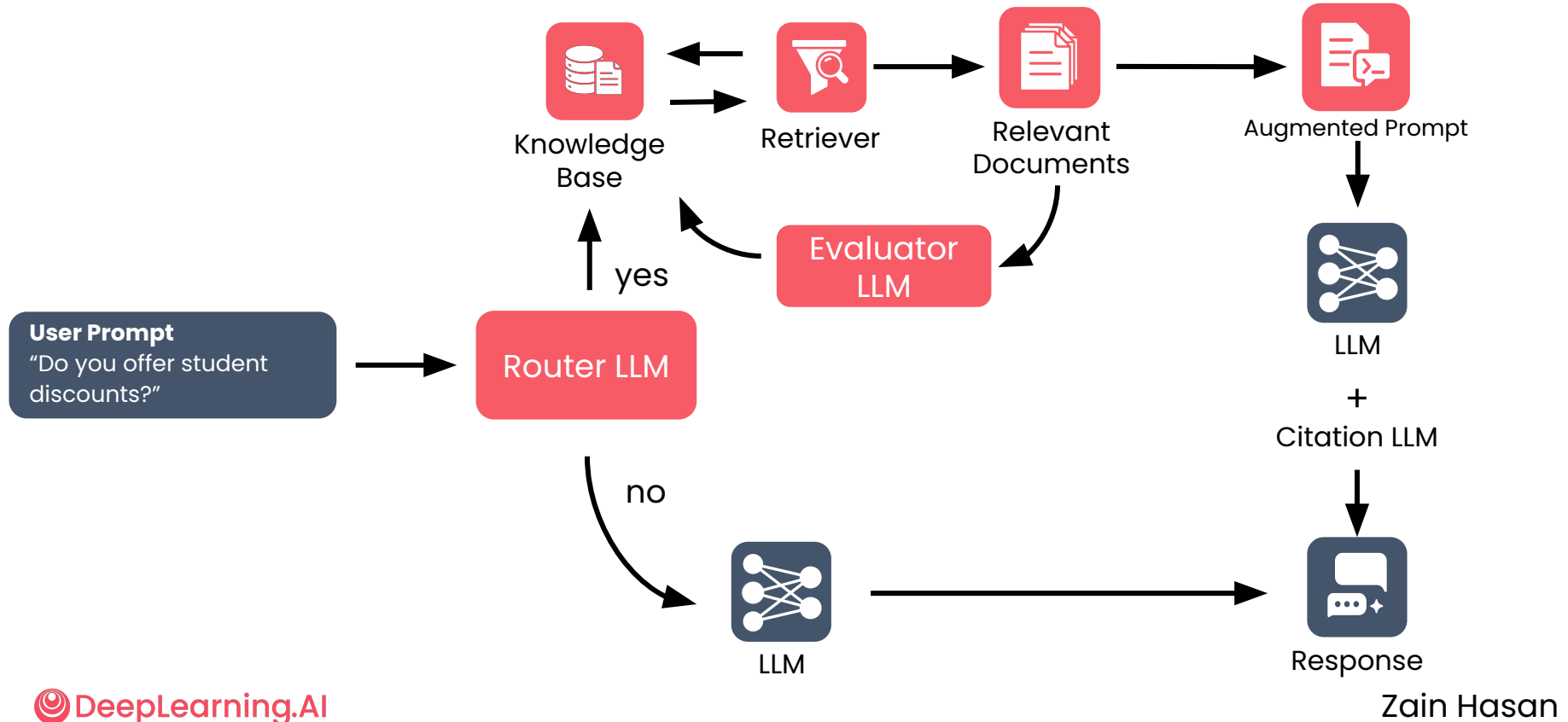
DeepLearning.AI

Agentic RAG

LLMs and Text
Generation



Example Agentic RAG System



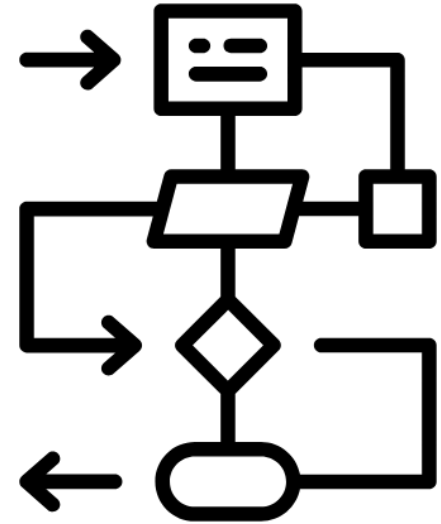
Agentic Systems As Flowcharts

- **Agentic systems as flowcharts**

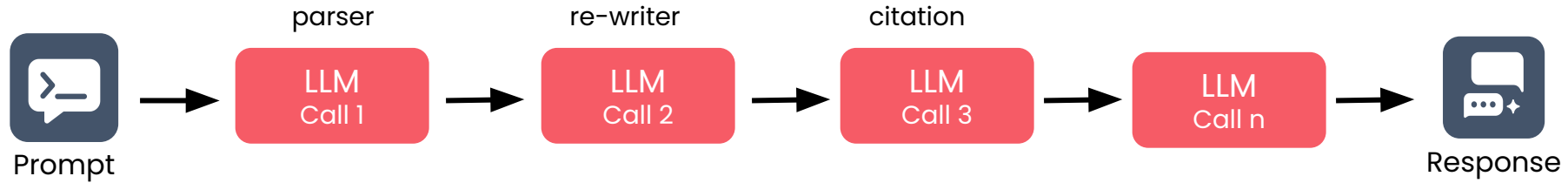
Each LLM completes one specific task in the prompt's journey, taking text input and generating text output at each step

- **Different LLMs for different tasks**

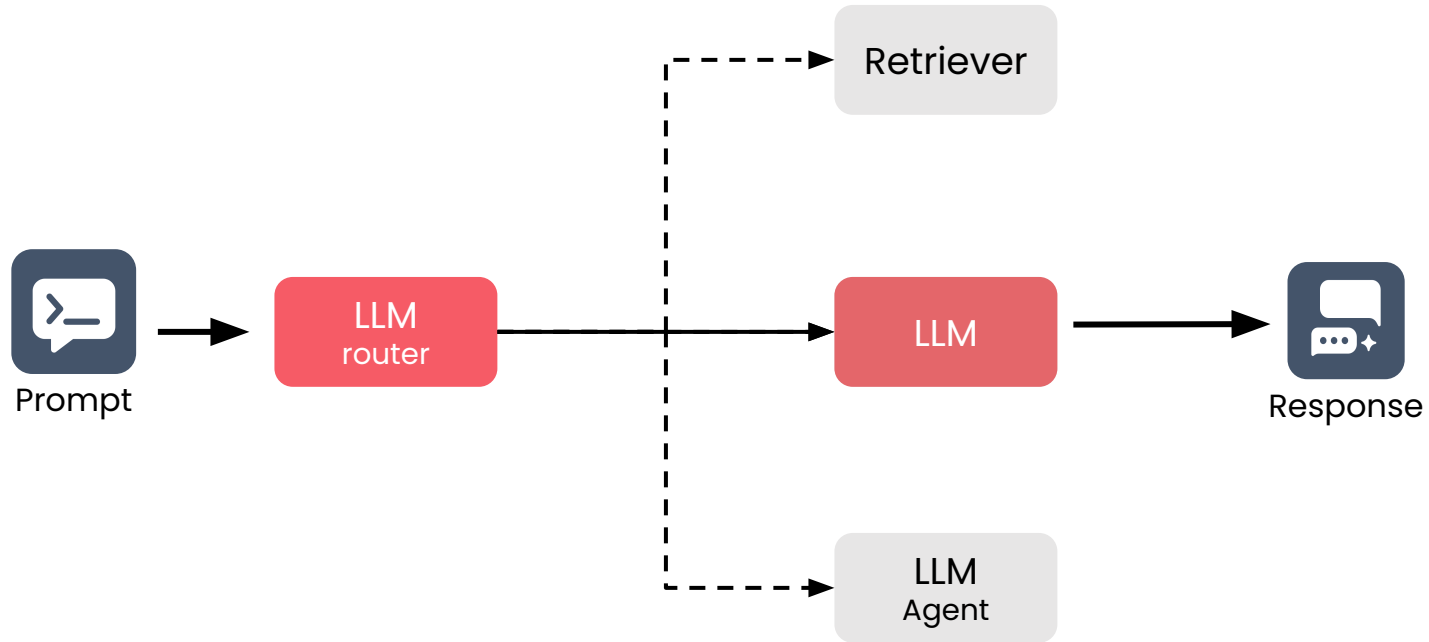
Use lightweight models for routing and evaluation, larger models for response generation, and specialized models for citations



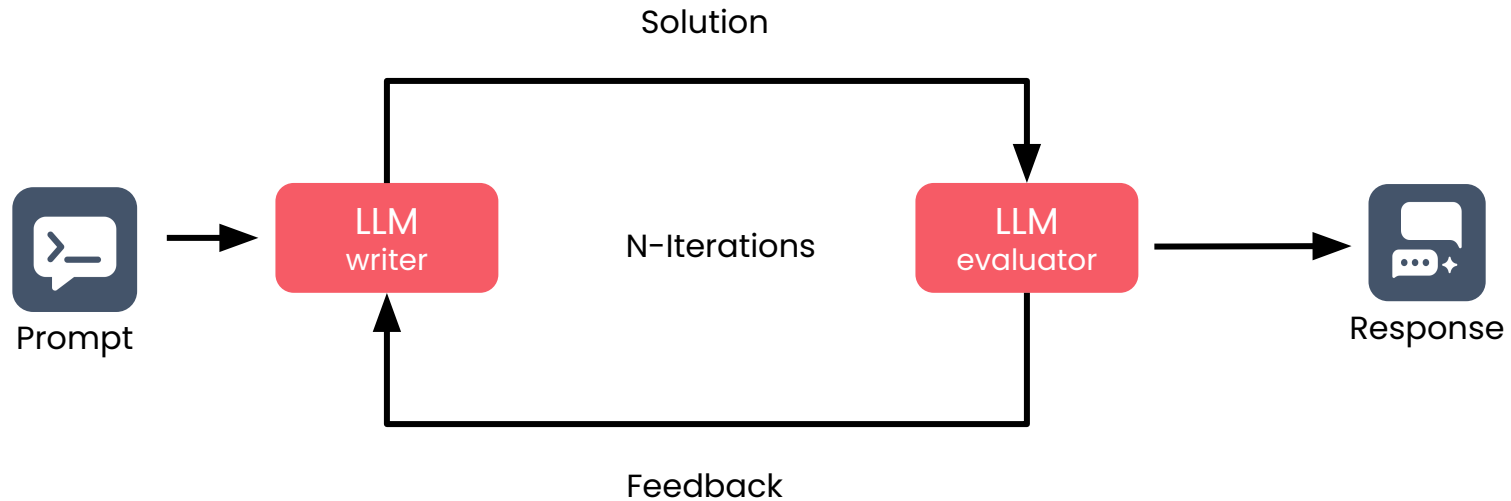
Sequential Workflow



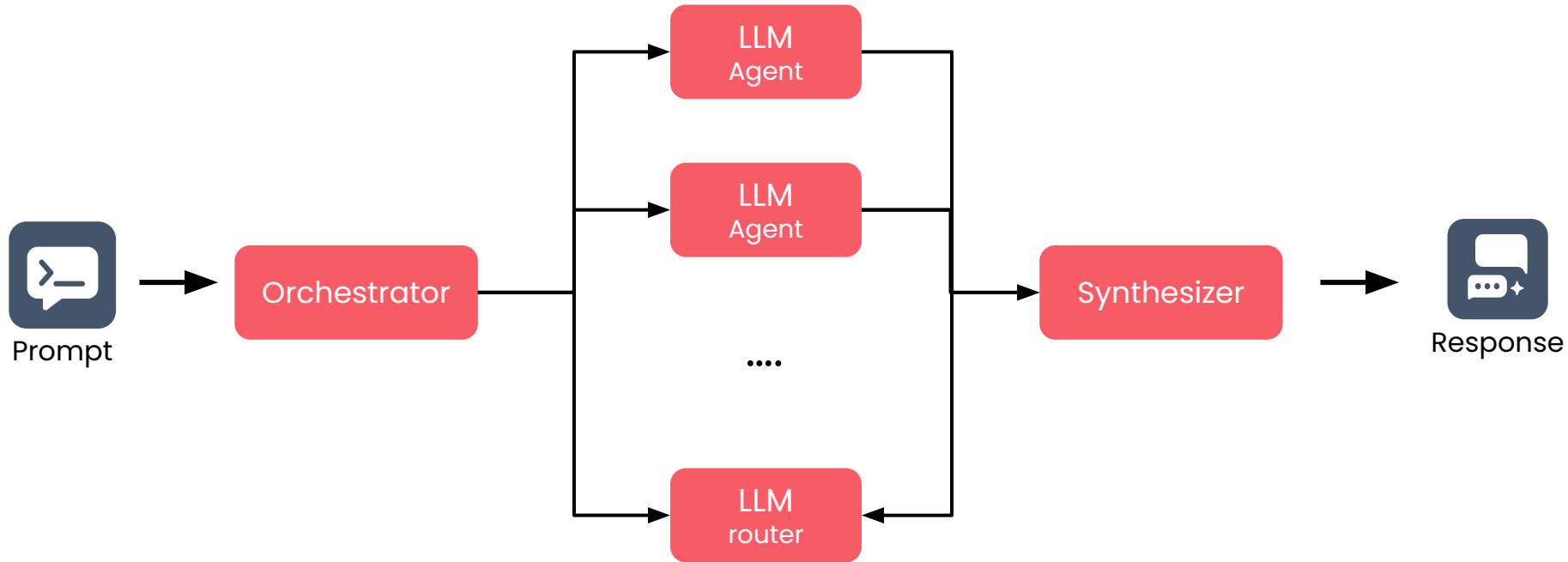
Conditional Workflow



Iterative Workflow



Parallel Workflow



Implementing Agentic Systems

- For simple agentic systems, you can manually implement the **workflow logic yourself**.
- As complexity increases, consider using **tools, libraries, or platforms** to build and manage your agentic system.





DeepLearning.AI

RAG vs. Fine-Tuning

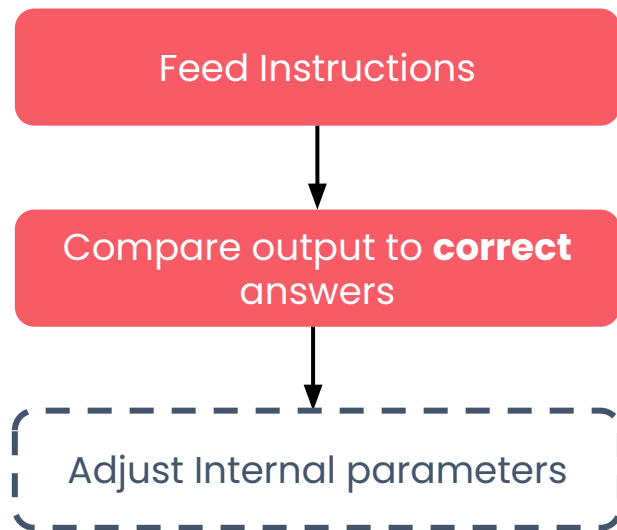
LLMs and Text
Generation

Understanding Fine-Tuning

The core idea of fine-tuning is to retrain an LLM with your own data to update its internal parameters.

Done through **supervised fine-tuning**

- SFT retrains the model with labeled examples
- Instruction tuning teaches task-following behavior



Fine-Tuning Example



**Before
Fine-Tuning**

Input: I have joint pain, skin rash, and sun sensitivity. What could this be?

Output: Those are pretty common symptoms. Joint pain might be from working out too much. Skin rashes happen all the time. Sun sensitivity could just mean you need better sunscreen!



**After
Fine-Tuning**

Input: I have joint pain, skin rash, and sun sensitivity. What could this be?

Output: This symptom combination may suggest autoimmune conditions like systemic lupus erythematosus (SLE). These symptoms warrant medical evaluation for proper assessment.

When Fine-Tuning Works Well

Fine-tuning can work well in

- Providing initial medical diagnosis
- Summarizing legal briefs
- Small models can be fine-tuned for narrow tasks in agent systems.

Performance improves in target domain but can decrease in others. Fine-tuning only optimizes for the target domain.

Fine tuning does not teach new facts well. It mostly **changes how a model talks, not what it knows.**

RAG vs Fine-Tuning

RAG

- RAG is best when the LLM needs new information.
- You can inject context into the prompt, and the model will use it.

RAG = best for knowledge injection

Fine Tuning

- Fine-tuning is best for task or domain specialization.
- It's ideal for one clear, focused task.

Fine tuning = best for domain adaptation

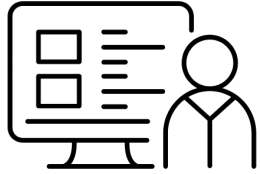
Why Choose?

Fine-tuning for RAG: Train models specifically to incorporate retrieved information into responses.

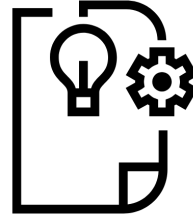
- RAG provides current, accurate information
- Fine-tuning optimizes how models use that information

When deciding whether to use fine-tuning or RAG, the best choice might be **both!**

Getting Started with Fine-Tuning



Take a dedicated
course



Use pre-tuned
models



DeepLearning.AI

Module 4 Conclusion

LLMs and Text
Generation

Conclusion

- **Transformer Architecture:** How LLMs process text to generate meaningful completions
- **Sampling Strategies:** Controlling randomness to tune text generation for your needs
- **Model Selection & Prompting:** Choosing LLMs, engineering prompts, preventing hallucinations
- **Performance Evaluation:** Techniques to measure LLM effectiveness
- **Advanced Capabilities:** Agentic components and fine-tuning to enhance RAG systems