Unit 2 - Part 2a: The Anatomy of a Prompt

1. Introduction: Stochasticity (Randomness)

Why does the AI give different answers? Because it is **Stochastic** (Random).

It predicts the NEXT TOKEN based on probability.

Visualizing the Prediction

Input: `"The sky is..."`

Word	Probability	Selected? (Temp=0)	Selected? (Temp=1)
Blue	80%	✅	❌
Gray	15%	❌	✅
Green	1%	❌	❌

Prompt Engineering is the art of **manipulating these probabilities**.

In [3]:
```
# Setup
!pip install langchain-google-genai
from dotenv import load_dotenv
load_dotenv()

import getpass
import os
from langchain_google_genai import ChatGoogleGenerativeAI

if "GOOGLE_API_KEY" not in os.environ:
    os.environ["GOOGLE_API_KEY"] = getpass.getpass("Enter your Google API Key:

# Using Low Temp for consistent comparison
llm = ChatGoogleGenerativeAI(model="gemini-2.5-flash", temperature=0.0)
```

```
Collecting langchain-google-genai
  Downloading langchain_google_genai-4.2.0-py3-none-any.whl.metadata (2.7 kB)
Collecting filetype<2.0.0,>=1.2.0 (from langchain-google-genai)
  Downloading filetype-1.2.0-py2.py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: google-genai<2.0.0,>=1.56.0 in /usr/local/lib/py
thon3.12/dist-packages (from langchain-google-genai) (1.62.0)
Requirement already satisfied: langchain-core<2.0.0,>=1.2.5 in /usr/local/lib/p
ython3.12/dist-packages (from langchain-google-genai) (1.2.9)
Requirement already satisfied: pydantic<3.0.0,>=2.0.0 in /usr/local/lib/python
3.12/dist-packages (from langchain-google-genai) (2.12.3)
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /usr/local/lib/python3.1
2/dist-packages (from google-genai<2.0.0,>=1.56.0->langchain-google-genai) (4.1
2.1)
Requirement already satisfied: google-auth<3.0.0,>=2.47.0 in /usr/local/lib/pyt
hon3.12/dist-packages (from google-auth[requests]<3.0.0,>=2.47.0->google-gena
i<2.0.0,>=1.56.0->langchain-google-genai) (2.47.0)
Requirement already satisfied: httpx<1.0.0,>=0.28.1 in /usr/local/lib/python3.1
2/dist-packages (from google-genai<2.0.0,>=1.56.0->langchain-google-genai) (0.2
8.1)
Requirement already satisfied: requests<3.0.0,>=2.28.1 in /usr/local/lib/python
3.12/dist-packages (from google-genai<2.0.0,>=1.56.0->langchain-google-genai)
(2.32.4)
Requirement already satisfied: tenacity<9.2.0,>=8.2.3 in /usr/local/lib/python
3.12/dist-packages (from google-genai<2.0.0,>=1.56.0->langchain-google-genai)
(9.1.3)
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /usr/local/lib/pyt
hon3.12/dist-packages (from google-genai<2.0.0,>=1.56.0->langchain-google-gena
i) (15.0.1)
Requirement already satisfied: typing-extensions<5.0.0,>=4.11.0 in /usr/local/l
ib/python3.12/dist-packages (from google-genai<2.0.0,>=1.56.0->langchain-googl
e-genai) (4.15.0)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.12/di
st-packages (from google-genai<2.0.0,>=1.56.0->langchain-google-genai) (1.9.0)
Requirement already satisfied: sniffio in /usr/local/lib/python3.12/dist-packag
es (from google-genai<2.0.0,>=1.56.0->langchain-google-genai) (1.3.1)
Requirement already satisfied: jsonpatch<2.0.0,>=1.33.0 in /usr/local/lib/pytho
n3.12/dist-packages (from langchain-core<2.0.0,>=1.2.5->langchain-google-genai)
(1.33)
Requirement already satisfied: langsmith<1.0.0,>=0.3.45 in /usr/local/lib/pytho
n3.12/dist-packages (from langchain-core<2.0.0,>=1.2.5->langchain-google-genai)
(0.6.9)
Requirement already satisfied: packaging>=23.2.0 in /usr/local/lib/python3.12/d
ist-packages (from langchain-core<2.0.0,>=1.2.5->langchain-google-genai) (26.0)
Requirement already satisfied: pyyaml<7.0.0,>=5.3.0 in /usr/local/lib/python3.1
2/dist-packages (from langchain-core<2.0.0,>=1.2.5->langchain-google-genai)
(6.0.3)
Requirement already satisfied: uuid-utils<1.0,>=0.12.0 in /usr/local/lib/python
3.12/dist-packages (from langchain-core<2.0.0,>=1.2.5->langchain-google-genai)
(0.14.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python
3.12/dist-packages (from pydantic<3.0.0,>=2.0.0->langchain-google-genai)
(0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python
3.12/dist-packages (from pydantic<3.0.0,>=2.0.0->langchain-google-genai) (2.4
```

```
1.4)
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/pytho
n3.12/dist-packages (from pydantic<3.0.0,>=2.0.0->langchain-google-genai)
(0.4.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-pack
ages (from anyio<5.0.0,>=4.8.0->google-genai<2.0.0,>=1.56.0->langchain-google-g
enai) (3.11)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python
3.12/dist-packages (from google-auth<3.0.0,>=2.47.0->google-auth[request
s]<3.0.0,>=2.47.0->google-genai<2.0.0,>=1.56.0->langchain-google-genai) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-
packages (from google-auth<3.0.0,>=2.47.0->google-auth[requests]<3.0.0,>=2.4
7.0->google-genai<2.0.0,>=1.56.0->langchain-google-genai) (4.9.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packag
es (from httpx<1.0.0,>=0.28.1->google-genai<2.0.0,>=1.56.0->langchain-google-ge
nai) (2026.1.4)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-
packages (from httpx<1.0.0,>=0.28.1->google-genai<2.0.0,>=1.56.0->langchain-goo
gle-genai) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-pack
ages (from httpcore==1.*->httpx<1.0.0,>=0.28.1->google-genai<2.0.0,>=1.56.0->la
ngchain-google-genai) (0.16.0)
Requirement already satisfied: jsonpointer>=1.9 in /usr/local/lib/python3.12/di
st-packages (from jsonpatch<2.0.0,>=1.33.0->langchain-core<2.0.0,>=1.2.5->langc
hain-google-genai) (3.0.0)
Requirement already satisfied: orjson>=3.9.14 in /usr/local/lib/python3.12/dis
t-packages (from langsmith<1.0.0,>=0.3.45->langchain-core<2.0.0,>=1.2.5->langch
ain-google-genai) (3.11.7)
Requirement already satisfied: requests-toolbelt>=1.0.0 in /usr/local/lib/pytho
n3.12/dist-packages (from langsmith<1.0.0,>=0.3.45->langchain-cor
e<2.0.0,>=1.2.5->langchain-google-genai) (1.0.0)
Requirement already satisfied: xxhash>=3.0.0 in /usr/local/lib/python3.12/dist-
packages (from langsmith<1.0.0,>=0.3.45->langchain-core<2.0.0,>=1.2.5->langchai
n-google-genai) (3.6.0)
Requirement already satisfied: zstandard>=0.23.0 in /usr/local/lib/python3.12/d
ist-packages (from langsmith<1.0.0,>=0.3.45->langchain-core<2.0.0,>=1.2.5->lang
chain-google-genai) (0.25.0)
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/pytho
n3.12/dist-packages (from requests<3.0.0,>=2.28.1->google-genai<2.0.0,>=1.5
6.0->langchain-google-genai) (3.4.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/
dist-packages (from requests<3.0.0,>=2.28.1->google-genai<2.0.0,>=1.56.0->langc
hain-google-genai) (2.5.0)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.1
2/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3.0.0,>=2.47.0->googl
e-auth[requests]<3.0.0,>=2.47.0->google-genai<2.0.0,>=1.56.0->langchain-google-
genai) (0.6.2)
Downloading langchain_google_genai-4.2.0-py3-none-any.whl (66 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.5/66.5 kB 2.5 MB/s eta 0:00:00
Downloading filetype-1.2.0-py2.py3-none-any.whl (19 kB)
Installing collected packages: filetype, langchain-google-genai
Successfully installed filetype-1.2.0 langchain-google-genai-4.2.0
Enter your Google API Key: ··········
```

2. The CO-STAR Framework (simplified)

A good prompt usually has:

1. **C**ontext (Who are you? Who acts?)
2. **O**bjective (What is the task?)
3. **S**tyle (Formal? Funny?)
4. **T**one (Empathetic? Direct?)
5. **A**udience (Who is reading this?)
6. **R**esponse Format (JSON? List?)

Let's compare a **Lazy Prompt** vs a **CO-STAR Prompt**.

In [4]:
```
# The Task: Reject a candidate for a job.
task = "Write a rejection email to a candidate."

print("--- LAZY PROMPT ---")
print(llm.invoke(task).content)
```

--- LAZY PROMPT ---
Here are a few options for a rejection email, ranging from a standard template to one for a candidate who interviewed. Choose the one that best fits your situation.

Option 1: Standard Rejection (No Interview)

This is suitable for candidates who applied but were not selected for an interview.

Subject: Update on Your Application for [Job Title] at [Company Name]

Dear [Candidate Name],

Thank you for your interest in the [Job Title] position at [Company Name] and for taking the time to submit your application.

We received a large number of highly qualified applications for this role. While your qualifications are impressive, we have decided to move forward with other candidates whose profiles were a closer match for the specific requirements of this position at this time.

We appreciate you considering [Company Name] as a potential employer and wish you the best of luck in your job search and future endeavors.

Sincerely,

[Your Name]
[Your Title]
[Company Name]
[Company Website (Optional)]

Option 2: Rejection After Interview(s)

This option acknowledges the time and effort the candidate put into the interview process.

Subject: Update Regarding Your Application for [Job Title] at [Company Name]

Dear [Candidate Name],

Thank you for your interest in the [Job Title] position at [Company Name] and for taking the time to interview with our team. We enjoyed learning more about your experience and qualifications.

We appreciate you sharing your background and insights during our discussions. This was a highly competitive search, and we received applications from many talented individuals. After careful consideration, we have decided to move forward with another candidate whose qualifications and experience were a closer matc

h for the specific needs of this role at this time.

We truly appreciate your time and effort throughout the interview process. We w
ish you the very best in your job search and future career.

Sincerely,

[Your Name]
[Your Title]
[Company Name]
[Company Website (Optional)]

Option 3: Rejection with "Keep on File" Option (Use with Caution)

Only use this if you genuinely might consider them for future roles and have a
system to track this.

Subject: Update on Your Application for [Job Title] at [Company Name]

Dear [Candidate Name],

Thank you for your interest in the [Job Title] position at [Company Name] and f
or taking the time to apply/interview with us. We appreciate you sharing your b
ackground and experience.

We received a significant number of applications for this role, and the selecti
on process was highly competitive. While your qualifications are impressive, we
have decided to move forward with another candidate whose profile was the best
fit for our current needs.

We were impressed with your [mention something general like "experience" or "en
thusiasm"] and would like to keep your resume on file for future opportunities
that may align more closely with your skills.

We wish you the best of luck in your job search and future endeavors.

Sincerely,

[Your Name]
[Your Title]
[Company Name]
[Company Website (Optional)]

Key Considerations for Rejection Emails:

*   **Be Timely:** Send it as soon as a decision is made. Don't leave candidate
s hanging.
*   **Be Clear and Direct:** Don't beat around the bush, but be polite.
*   **Be Professional:** Maintain a positive image for your company.
*   **Be Vague on Reasons:** Avoid giving specific reasons for rejection (e.g.,

```
"you lacked X skill," "your personality wasn't a fit"). This can open the door
to legal issues or arguments. Focus on the company's needs and the chosen candi
date's "closer match."
*   **Personalize:** Always use the candidate's name.
*   **Proofread:** Ensure there are no typos or grammatical errors.
```

3. Hallucination vs. Creativity

Did the model make up a reason? Since we didn't give it facts, it **Predicted the most likely reason** (Usually "Experience" or "Volume of applications").

This is NOT a bug. It is a feature. The model is *completing the pattern* of a rejection email.

In [5]:
```
structured_prompt = """
# Context
You are an HR Manager at a quirky startup called 'RocketBoots'.

# Objective
Write a rejection email to a candidate named Bob.

# Constraints
1. Be extremely brief (under 50 words).
2. Do NOT say 'we found someone better'. Say 'the role changed'.
3. Sign off with 'Keep flying'.

# Output Format
Plain text, no subject line.
"""

print("--- STRUCTURED PROMPT ---")
print(llm.invoke(structured_prompt).content)
```

```
--- STRUCTURED PROMPT ---
Hi Bob,

Thank you for your interest in RocketBoots.

We appreciate you taking the time to interview. However, the role's requirement
s have changed significantly since your application. We wish you the best in yo
ur job search.

Keep flying.
```

4. Key Takeaway: Ambiguity is the Enemy

Every piece of information you leave out is a gap the model MUST fill with probability.

- If you don't say "Be brief", it picks the most probable length (Avg email

length).
- If you don't say "Be rude", it picks the most probable tone (Polite/Neutral).

Assignment

Write a structured prompt to generate a **Python Function**.

- **Context:** You are a Senior Python Dev.
- **Objective:** Write a function to reverse a string.
- **Constraint:** It must use recursion (no slicing `[::-1]`).
- **Style:** Include detailed docstrings.

In [8]:
```
structured_prompt = """
# Context
You are a Senior Python Developer who writes clean, PEP8 compliant code.

# Objective
Write a Python function named `reverse_string_recursive` that takes a string a

# Constraints
1. You MUST use recursion.
2. Do NOT use slicing like `[::-1]`.
3. Include a detailed Google-style docstring.

# Output Format
Provide only the Python code block.
"""

print(llm.invoke(structured_prompt).content)
```

```python
def reverse_string_recursive(text: str) -> str:
    """Reverses a given string using recursion.

    This function takes an input string and returns a new string with its
    characters in reverse order. It achieves this by identifying a base case
    (an empty or single-character string, which is already reversed) and
    a recursive step. In the recursive step, it takes the last character
    of the string and prepends it to the recursively reversed version of
    the rest of the string (all characters except the last one).

    Args:
        text (str): The input string to be reversed.

    Returns:
        str: The reversed string.

    Examples:
        >>> reverse_string_recursive("hello")
        'olleh'
        >>> reverse_string_recursive("Python")
        'nohtyP'
        >>> reverse_string_recursive("")
        ''
        >>> reverse_string_recursive("a")
        'a'
        >>> reverse_string_recursive("madam")
        'madam'
    """
    # Base case: If the string is empty or has only one character,
    # it's already reversed.
    if len(text) <= 1:
        return text
    # Recursive step: Take the last character of the string and
    # prepend it to the recursively reversed version of the
    # string without its last character.
    else:
        return text[-1] + reverse_string_recursive(text[:-1])
```

Unit 2 - Part 2b: Zero-Shot to Few-Shot

1. Introduction: In-Context Learning

How does the model learn without training? This is called **In-Context Learning**.

The Attention Mechanism (Flowchart)

When you ask a question, the model "looks back" at the previous text to find

patterns.

In [6]:
```python
# Setup
from dotenv import load_dotenv
load_dotenv()

import getpass
import os
from langchain_google_genai import ChatGoogleGenerativeAI

if "GOOGLE_API_KEY" not in os.environ:
    os.environ["GOOGLE_API_KEY"] = getpass.getpass("Enter your Google API Key:

llm = ChatGoogleGenerativeAI(model="gemini-2.5-flash", temperature=0.5)
```

2. Zero-Shot (No Context)

The model relies purely on its training data.

In [7]:
```python
prompt_zero = "Combine 'Angry' and 'Hungry' into a funny new word."
print(f"Zero-Shot: {llm.invoke(prompt_zero).content}")
```

```
Zero-Shot: The most common and widely accepted funny word for this is **Hangr
y**.
```

3. Few-Shot (Pattern Matching)

We provide examples. The Attention Mechanism attends to the **Structure** (`Input -> Output`) and the **Tone** (Sarcasm).

In [9]:
```python
prompt_few = """
Combine words into a funny new word. Give a sarcastic definition.

Input: Breakfast + Lunch
Output: Brunch (An excuse to drink alcohol before noon)

Input: Chill + Relax
Output: Chillax (What annoying people say when you are panic attacks)

Input: Angry + Hungry
Output:
"""
print(f"Few-Shot: {llm.invoke(prompt_few).content}")
```

```
Few-Shot: Input: Angry + Hungry
Output: Hangry (A perfectly valid excuse for adults to throw a tantrum because
their food is five minutes late.)
```

4. Critical Analysis

If you provide **bad examples**, the model will learn the **bad pattern**. This is why Data Quality in your prompt is just as important as code quality.

Unit 2 - Part 2c: Advanced Templates & Theory

1. Theory: Engineering vs. Training

Hard Prompts (Prompt Engineering)

- **What:** You change the text input.
- **Cost:** Cheap, fast, easy to iterate.
- **Use Case:** Prototyping, General tasks.

Soft Prompts (Fine Tuning)

- **What:** You change the model's internal weights (mathematically).
- **Cost:** Expensive, slow, needs data.
- **Use Case:** Domain specificity (Medical, Legal), Behavioral change.

In [10]:
```
# Setup
from dotenv import load_dotenv
load_dotenv()

import getpass
import os
from langchain_google_genai import ChatGoogleGenerativeAI

if "GOOGLE_API_KEY" not in os.environ:
    os.environ["GOOGLE_API_KEY"] = getpass.getpass("Enter your Google API Key:
llm = ChatGoogleGenerativeAI(model="gemini-2.5-flash")
```

2. Dynamic Few-Shotting

If you have 1000 examples, you can't fit them all in the context window. We use a **Selector** to pick the best ones.

The Selector Flow (Flowchart)

In [11]:
```
from langchain_core.prompts import ChatPromptTemplate, FewShotChatMessagePromp

# 1. Our Database of Examples
examples = [
    {"input": "The internet is down.", "output": "We are observing connectivit
    {"input": "This code implies a bug.", "output": "The logic suggests uninte
    {"input": "I hate this feature.", "output": "This feature does not align w
]

# 2. Template for ONE example
example_fmt = ChatPromptTemplate.from_messages([
    ("human", "{input}"),
    ("ai", "{output}")
])

# 3. The Few-Shot Container
few_shot_prompt = FewShotChatMessagePromptTemplate(
    example_prompt=example_fmt,
    examples=examples
)

# 4. The Final Chain
final_prompt = ChatPromptTemplate.from_messages([
    ("system", "You are a Corpo-Speak Translator. Rewrite the input to sound p
    few_shot_prompt,      # Inject examples here
    ("human", "{text}")
])

chain = final_prompt | llm

print(chain.invoke({"text": "This app sucks."}).content)
```

```
The application's current iteration presents opportunities for enhancement.
```

3. Analysis

Using `FewShotChatMessagePromptTemplate` creates a clean separation between instructions and data. This helps the Attention Mechanism focus on the right things.