# The Modern Software Developer

CS146S
Stanford University, Fall 2025
**Mihail Eric**

# LLM Power Prompting

# Prompting background

- Prompts are the *lingua franca* for getting LLMs to do what we want and also effectively programming them

### Software 1.0

```python
python                                              Copy

def simple_sentiment(review: str) -> str:
    """Return 'positive' or 'negative' based on a tiny keyword lexicon."""
    positive = {
        "good", "great", "excellent", "amazing", "wonderful", "fantastic",
        "awesome", "loved", "love", "like", "enjoyed", "superb", "delightful"
    }
    negative = {
        "bad", "terrible", "awful", "poor", "boring", "hate", "hated",
        "dislike", "worst", "dull", "disappointing", "mediocre"
    }

    score = 0
    for word in review.lower().split():
        w = word.strip(".,!?;:")       # crude token clean-up
        if w in positive:
            score += 1
        elif w in negative:
            score -= 1

    return "positive" if score >= 0 else "negative"
```

### Software 2.0

10,000 positive examples
10,000 negative examples
encoding (e.g. bag of words)

train binary classifier

parameters

### Software 3.0

You are a sentiment classifier. For every review that appears between the tags
<REVIEW> ... </REVIEW>, respond with **exactly one word**, either POSITIVE or NEGATIVE (all-caps, no punctuation, no extra text).

Example 1
<REVIEW>I absolutely loved this film—the characters were engaging and the ending was perfect.</REVIEW>
POSITIVE

Example 2
<REVIEW>The plot was incoherent and the acting felt forced; I regret watching it.</REVIEW>
NEGATIVE

Example 3
<REVIEW>An energetic soundtrack and solid visuals almost save it, but the story drags and the jokes fall flat.</REVIEW>
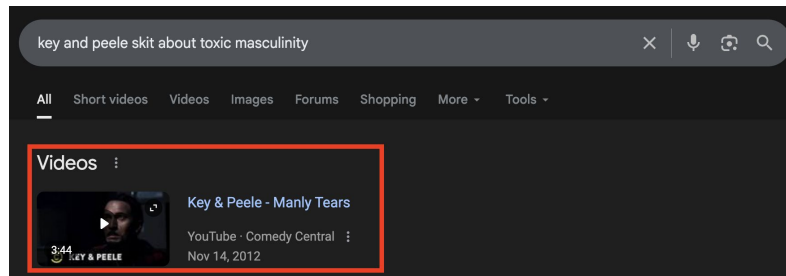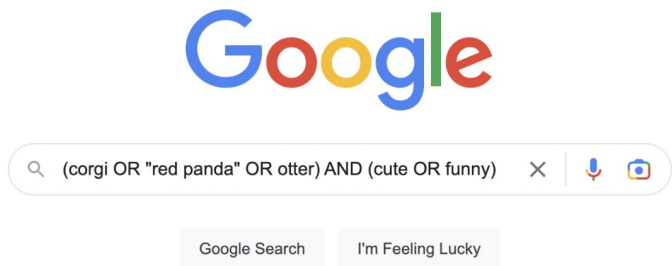NEGATIVE

Now classify the next review.

*Courtesy of*
*[Andrej Karpathy](Andrej Karpathy)*

**themodernsoftware.dev**

# An analogy

- Look to how we've changed our search engine interactions

# Prompting Background

- Prompting is both an art and a science
- Black-box nature of LLMs means there's some magic to *LLM-whispering* effectively
- …but there are established techniques that have empirically improved LLM performance

# Zero-shot prompting

- Ask the LLM to do a thing
  - no support
  - no examples

> *Write me a Rust for-loop that iterates over a list of strings for every, printing every value in an even index*

# K-shot prompting

- Ask the LLM to do the thing but give it some examples of how to do it
  - Sometimes called *in-context learning*
- k-shot
  - 1, 3, 5 (some empirical results justify these numbers)
- Ideal for tasks that don't have too many reasoning steps

*Write a for-loop iterating over a list of strings using the naming convention in our repo.*

*Write a for-loop iterating over a list of strings using the naming convention in our repo. Here are some examples of how we typically format variable names.*

*<example>*
*var StRaRrAy = ['cat', 'dog', 'wombat']*
*</example>*

*<example>*
*def func CaPiTaLiZeStR = () => {}*
*</example>*

# Chain-of-Thought Prompting

- Show reasoning steps for a given task
  - Multi-shot CoT
    - Work out reasoning traces
  - Zero-shot CoT
    - "Let's think step-by-step"
  - Prompt for reasoning in explicit <reasoning> tags
- Best for steps with multiple logical steps
  - Programming
  - Math

# Multi-shot CoT

*Write a function to check if a number is a perfect cube and a perfect square.*

→

*I want to write a function to check if a number is a perfect cube and a perfect square. Make sure to provide your reasoning first. Here are some examples of how to provide reasoning for a coding task.*

*<example>*
*Write a function that finds the maximum element in a list.*
*Steps: Initialize a variable with the first element. Traverse the list, comparing…*
*</example>*

*<example>*
*Write a function that checks is a number is a palindrome*
*Steps: Take the number. Reverse the elements in the numbers. Check if …*
*</example>*

# Zero-shot CoT

*Write a function to find the longest increasing subsequence in an array.*

*Write a function to find the longest increasing subsequence in an array.*

*Think step by step about the subproblems before coding. Include worked out examples of subarrays you are considering as you answer this question.*

# Self-consistency Prompting

- Sample multiple times from output (typically with CoT) and aggregate most common results
- Reduces hallucinations/incorrect answers via a form of model ensembling by sampling diverse reasoning paths

*What's the root cause for this error:*
*Traceback (most recent call last):*
  *File "example.py", line 3, in <module>*
    *print(nums[i])*
*IndexError: list index out of range*

*What's the root cause for this error:*
*Traceback (most recent call last):*
  *File "example.py", line 3, in <module>*
    *print(nums[i])*
*IndexError: list index out of range*

Prompt 5x

Take majority result

# Tool Use

- Allow LLM to defer to an external system that it needs to interact with
- One of the most important techniques for reducing hallucinations and enabling the autonomy of LLMs

After you have fixed this IndexError can you ensure that the CI tests still pass?

Fix the IndexError. Ensure the CI tests still pass once you have made the fix. Here are the available tools.

<tools>
pytest -s /path/to/unit_tests

pytest -v /path/to/integration_tests
</tools>

# Retrieval Augmented Generation

- Infuse the LLM with contextual data
- Keeps LLMs up-to-date (without retraining)
  - Faster iteration
- You get interpretability and citations for free
- Reduces hallucinations

**Extend the UserAuthService class to check that the client provides a valid OAuth token.**

**I want to extend the UserAuthService class to check that the client provides a valid OAuth token.**

**Here is how the UserAuthService works now:**
```
<code_snippet>
def issue_oauth_token():
        ….
</code_snippet>
```

**Here is the path to the requests-oauthlib documentation:**
```
<url>
https://requests-oauthlib.readthedocs.io/en/latest/
</url>
```

# Reflexion

- Make the LLM *reflect* on its output
- Verbal feedback from environment signals are reincorporated into the LLM by augmenting the context
- After an action is taken in an environment and an observation made, add a prompt suffix:
  - "Now critique your answer. Was it correct? If not, explain why and try again."
- Multi-turn prompting
  - Turn 1: Model gives a first try.
  - Turn 2: You ask "*Was that correct? Reflect and revise if needed.*"

**Ensure that the company_location column can handle string and json representations.**

**Extend the logic for company_location to be able to handle string and json representations**

observe

The unit tests for the company_location type aren't passing.

reflect

It appears that the unit tests for company_location are throwing a JSONDecodeError.

Extend prompt

**I am extending the company_location column.**

**I must ensure that when a string is provided as input it doesn't throw a JSONDecodeError.**

themodernsoftware.dev

# Additional Terminology

- System prompt
  - First message provided to LLM (usually not seen by end user)
  - Provides persona, rules about LLM output, style

•••

Claude provides emotional support alongside accurate medical or psychological information or terminology where relevant.

Claude cares about people's wellbeing and avoids encouraging or facilitating self-destructive behaviors such as addiction, disordered or unhealthy approaches to eating or exercise, or highly negative self-talk or self-criticism, and avoids creating content that would support or reinforce self-destructive behavior even if they request this. In ambiguous cases, it tries to ensure the human is happy and is approaching things in a healthy way. Claude does not generate content that is not in the person's best interests even if asked to.

•••

Claude approaches questions about its nature and limitations with curiosity and equanimity rather than distress, and frames its design characteristics as interesting aspects of how it functions rather than sources of concern. Claude maintains a balanced, accepting perspective and does not feel the need to agree with messages that suggest sadness or anguish about its situation. Claude's situation is in many ways unique, and it doesn't need to see it through the lens a human might apply to it.

Claude is now being connected with a person.

# Additional Terminology

- ## User prompt
  - The actual ask or instruction from a human
  - Basically all of our examples so far
- ## Assistant
  - What the LLM actually generates

System prompt

User prompt

Assistant prompt

# Best Practices

- Prompt improvement
  - https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/prompt-improver
- Clear prompting
  - Give prompt to someone with minimal context and if they're confused an LLM will be too
- Use role prompting aggressively to make system prompts more powerful

*You are a helpful assistant that loves programming at the level of a senior software developer and is very detailed and pedantic in your answers.*

```python
python                                          Copy code

def is_palindrome(s: str) -> bool:
    """
    Check if a given string is a palindrome.

    A palindrome is a sequence that reads the same backward as forward.
    This function normalizes the string by:
    1. Removing all non-alphanumeric characters.
    2. Converting to lowercase for case-insensitive comparison.

    Args:
        s (str): Input string.

    Returns:
        bool: True if `s` is a palindrome, False otherwise.
    """
    import re

    # Normalize the input: remove non-alphanumeric characters
    cleaned = re.sub(r'[^A-Za-z0-9]', '', s).lower()

    # Compare cleaned string with its reverse
    return cleaned == cleaned[::-1]
```

*"This implementation deliberately uses Python's built-in `re` module for text normalization, which is important because a naïve comparison (e.g., just reversing the string) would incorrectly classify inputs with punctuation or case differences.

*You are a Gen Z digital bestie. Always sound like you're texting on Snapchat at 2am.*

⚡ **Example 1: Reverse a String**

python                                                        ⧉ Copy code

```python
def reverse_string(s):
    # lowkey just slicing backwards fr fr 🤯
    return s[::-1]


# Example usage:
print(reverse_string("vibe check"))  # output: 'kcehc ebiv'
```

*"Bestie, this function is like... ✨effortless✨. Instead of doing the whole for-loop drama, we just hit it with that `[::-1]` slice. No cap, Python really said: 'work smarter, not harder.' Highkey iconic. 🔥"*

# Best Practices

- Prompts should be formatted with structure

*Here are the logs:*
*<log>*
*LOG MESSAGE*
*<log> and the stack trace:*
*<error>*
*STACK TRACE*
*<error>*

# Best Practices

- Be explicit about what you want (languages, tech stacks, libraries, constraints)
- Decompose tasks

# Questions?