Prompt Engineering

How to Trick AI into Solving Your Problems

Shaw Talebi

Prompt Engineering??..

Whedon, J. (2012). *The Avengers*. Walt Disney Studios Motion Pictures.
Favreau, J. (2008). *Iron Man*. Paramount Pictures.

Agenda

I. What is Prompt Engineering?

II. Two Levels of Prompt Engineering

III. How to Build AI Apps with it

IV. 7 Tricks for Prompt Engineering

V. Example Code: Automatic Grader with LangChain

What is Prompt Engineering?

Any use of an LLM out-of-the-box

1) Prompt Engineering is "*the means by which LLMs are programmed with prompts.*" [1]

2) Prompt Engineering is "*an empirical art of composing and formatting the prompt to maximize a model's performance on a desired task.*" [2]

3) "*language models… want to complete documents, and so you can trick them into performing tasks just by arranging fake documents.*" [3]

[1] arXiv:2302.11382 [cs.SE] (White et al. 2023), [2] arXiv:2106.09685 (Hu et al. 2021) [cs.CL] , [3] State of GPT by Andrej Karpathy at Microsoft Build 2023

Two Levels of Prompt Engineering

1) The Easy Way: ChatGPT (or something similar)

2) The Less Easy Way: Programmatically

Building AI Apps with Prompt Engineering

The Less Easy Way unlocks a new paradigm of software development

Use Case: Automatic Grader for high school history class

Question: "*Who was the 35th president of the United States of America?*"

Potential Correct Answers:

- *John F. Kennedy*
- *JFK*
- *Jack Kennedy (a common nickname)*
- *John Fitzgerald Kennedy (probably trying to get extra credit)*
- *John F. Kenedy (misspelled last name)*

Building AI Apps with Prompt Engineering

The Less Easy Way unlocks a new paradigm of software development

Traditional Paradigm

- On the developer to figure out logic to handle all variations
- Might requir
- Could use e algorithms

New Paradigm

- Use LLM to handle logic via prompt engineering

```
You are a high school history teacher grading homework assignments. \
Based on the homework question indicated by "Q:" and the correct answer \
indicated by "A:", your task is to determine whether the student's answer is \
correct.
Grading is binary; therefore, student answers can be correct or wrong.
Simple misspellings are okay.

Q: {question}
A: {correct_answer}

Student Answer: {student_answer}
```

7 Tricks for Prompt Engineering

Trick 1: Be Descriptive (More is better)

Trick 2: Give Examples

Trick 3: Use Structured Text

Trick 4: Chain of Thought

Trick 5: Chatbot Personas

Trick 6: Flipped Approach

Trick 7: Reflect, Review, and Refine

Demo

Example Code: Automatic Grader with LangChain

Imports

```
from langchain.chat_models import ChatOpenAI
from langchain.prompts import PromptTemplate
from langchain.chains import LLMChain
from langchain.schema import BaseOutputParser
```

```
from sk import my_sk #importing secret key from another python file
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Automatic Grader with LangChain

Our 1st Chain

```
# define LLM object
chat_model = ChatOpenAI(openai_api_key=my_sk, temperature=0)
```

```
# define prompt template
prompt_template_text = """You are a high school history teacher grading \
homework assignments. Based on the homework question indicated by “**Q:**” \
and the correct answer indicated by “**A:**”, your task is to determine \
whether the student's answer is correct. Grading is binary; therefore, \
student answers can be correct or wrong. Simple misspellings are okay.

**Q:** {question}
**A:** {correct_answer}

**Student's Answer:** {student_answer}
"""

prompt = PromptTemplate(
            input_variables=["question", "correct_answer", "student_answer"], \
            template = prompt_template_text)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Automatic Grader with LangChain

Our 1st Chain

```
# define chain
chain = LLMChain(llm=chat_model, prompt=prompt)
```

```
# define inputs
question = "Who was the 35th president of the United States of America?"
correct_answer = "John F. Kennedy"
student_answer =  "FDR"

# run chain
chain.run({'question':question, 'correct_answer':correct_answer, \
    'student_answer':student_answer})

# output: Student's Answer is wrong.
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Automatic Grader with LangChain

Output Parser

```
# define output parser
class GradeOutputParser(BaseOutputParser):
    """Determine whether grade was correct or wrong"""

    def parse(self, text: str):
        """Parse the output of an LLM call."""
        return "wrong" not in text.lower()
```

```
# update chain
chain = LLMChain(
    llm=chat_model,
    prompt=prompt,
    output_parser=GradeOutputParser()
)
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Example Code: Automatic Grader with LangChain

In action

```
# run chain in for loop
student_answer_list = ["John F. Kennedy", "JFK", "FDR", "John F. Kenedy", \
                  "John Kennedy", "Jack Kennedy", "Jacquelin Kennedy", \
                  "Robert F. Kenedy"]

for student_answer in student_answer_list:
    print(student_answer + " - " +
      str(chain.run({'question':question, 'correct_answer':correct_answer, \
                     'student_answer':student_answer})))
    print('\n')

# Output:
# John F. Kennedy - True
# JFK - True
# FDR - False
# John F. Kenedy - True
# John Kennedy - True
# Jack Kennedy - True
# Jacqueline Kennedy - False
# Robert F. Kenedy - False
```


https://github.com/ShawhinT/YouTube-Blog/tree/main/LLMs

Limitations

- Optimal prompt strategies are model-dependent
- Not all pertinent information can fit in context window
- General-purpose model may be cost inefficient and even overkill
- A (smaller) specialized model can out-perform a (larger) general-purpose model

Solution: Model fine-tuning