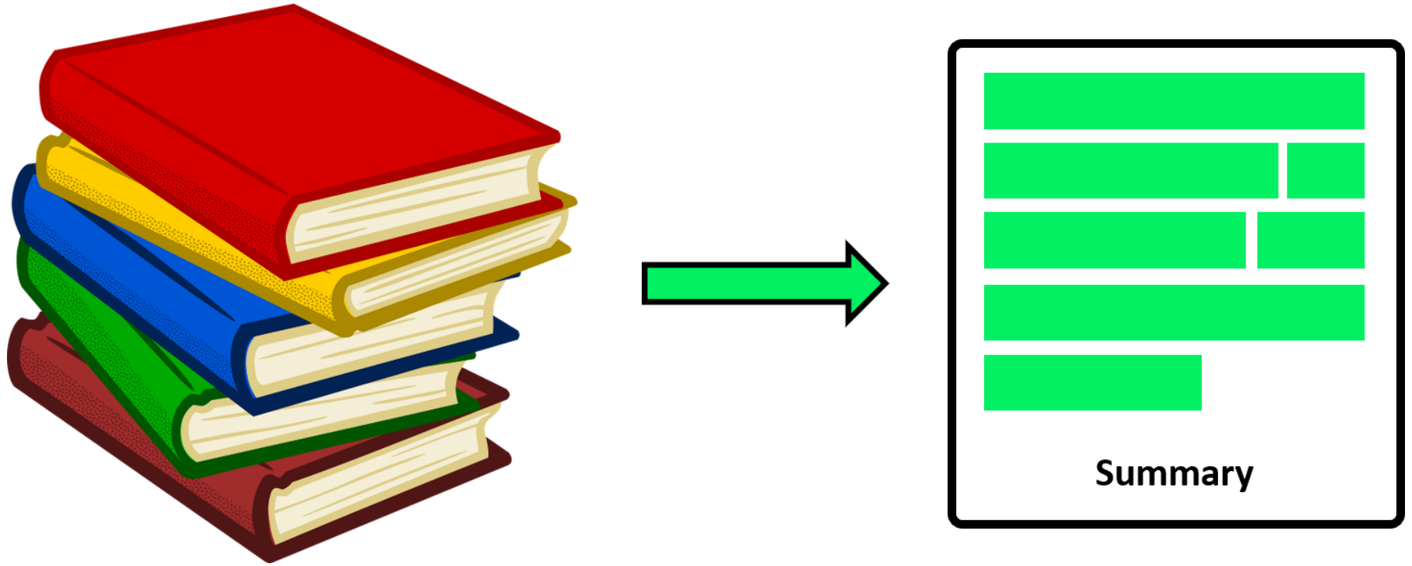
Text summarization and expansion

PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

Text summarization

- Condenses text into shorter format
- Streamlines business processes
  - Finance -> summarizes lengthy reports
  - Marketing -> transforms customer feedback into actionable insights
- LLMs can summarize texts with effective prompts



Ineffective prompt

- Only specifies text to summarize

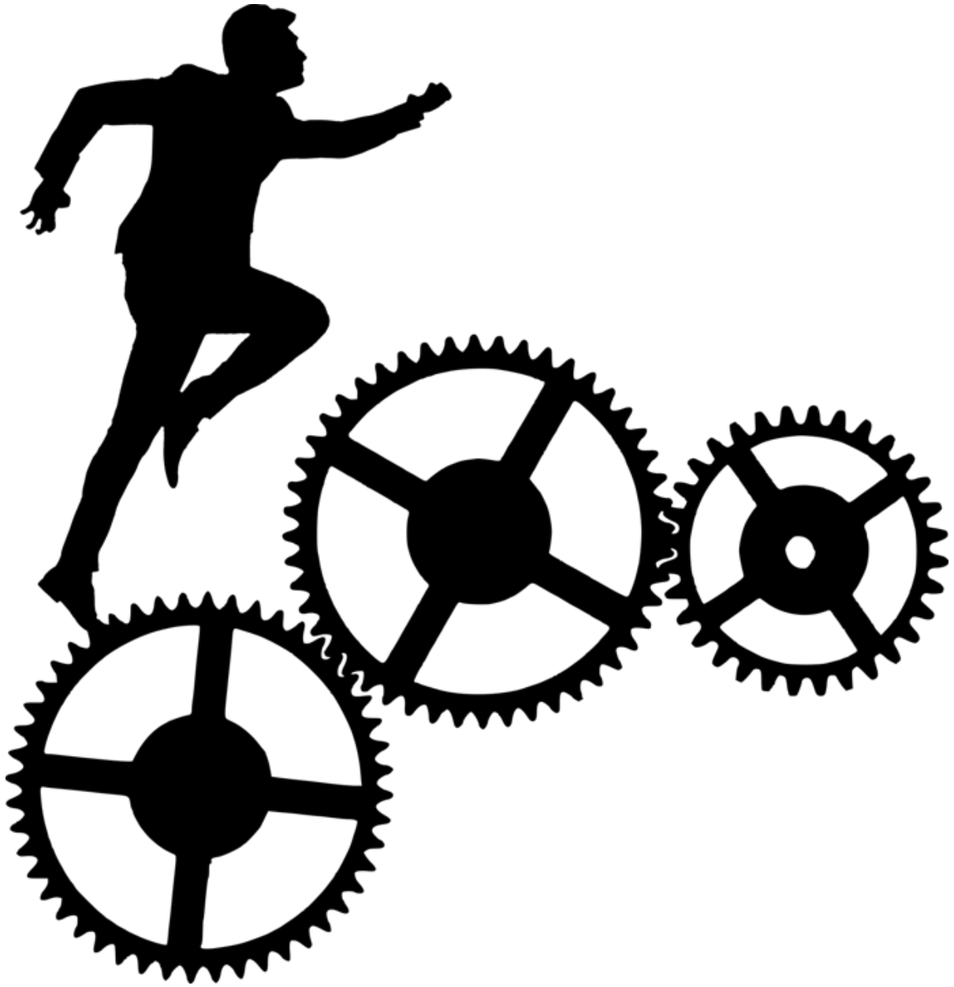
```
text = "I recently purchased your XYZ Smart Watch and wanted to provide some feedback
based on my experience with the product. I must say that I'm impressed with the sleek design
and build quality of the watch. It feels comfortable on the wrist and looks great with any
outfit. The touchscreen is responsive and easy to navigate through the various features."

prompt = f"""Summarize the text delimited by triple backticks:
            ```{text}```"""
print(get_response(prompt))
```

```
The author purchased the XYZ Smart Watch and is impressed with its sleek design, build
quality, and comfortable fit on the wrist. They find the touchscreen responsive and
user-friendly for navigating the watch's features.
```
```

Prompt improvement

- Output limits
- Output structure
- Summarization focus



Effective prompt: output limits

- Specify number of sentences, words, characters

```
text = "I recently purchased your XYZ Smart Watch and wanted to provide some feedback
based on my experience with the product. I must say that I'm impressed with the sleek design
and build quality of the watch. It feels comfortable on the wrist and looks great with any
outfit. The touchscreen is responsive and easy to navigate through the various features."

prompt = f"""Summarize the text delimited by triple backticks in one sentence:
            ```{text}```"""
print(get_response(prompt))
```

```
The customer is impressed with the sleek design, build quality, comfort, and responsiveness
of the XYZ Smart Watch's touch screen.
```
```

Effective prompt: output structure

- Specify output structure

```
text = "I recently purchased your XYZ Smart Watch and wanted to provide some feedback
based on my experience with the product. I must say that I'm impressed with the sleek design
and build quality of the watch. It feels comfortable on the wrist and looks great with any
outfit. The touchscreen is responsive and easy to navigate through the various features."

prompt = f"""Summarize the text delimited by triple backticks, in at most three bullet points.
            ```{text}```"""
print(get_response(prompt))
```

```
- The XYZ Smart Watch has a sleek and impressive design with  excellent build quality.
- It feels comfortable on the wrist and complements any outfit.
- The touch screen is responsive and user-friendly for easy navigation through the features.
```
```

Effective prompt: summarization focus

- Ask model to focus on specific parts of text

```
text = "I recently purchased your XYZ Smart Watch and wanted to provide some feedback
based on my experience with the product. I must say that I'm impressed with the sleek design
and build quality of the watch. It feels comfortable on the wrist and looks great with any
outfit. The touchscreen is responsive and easy to navigate through the various features."

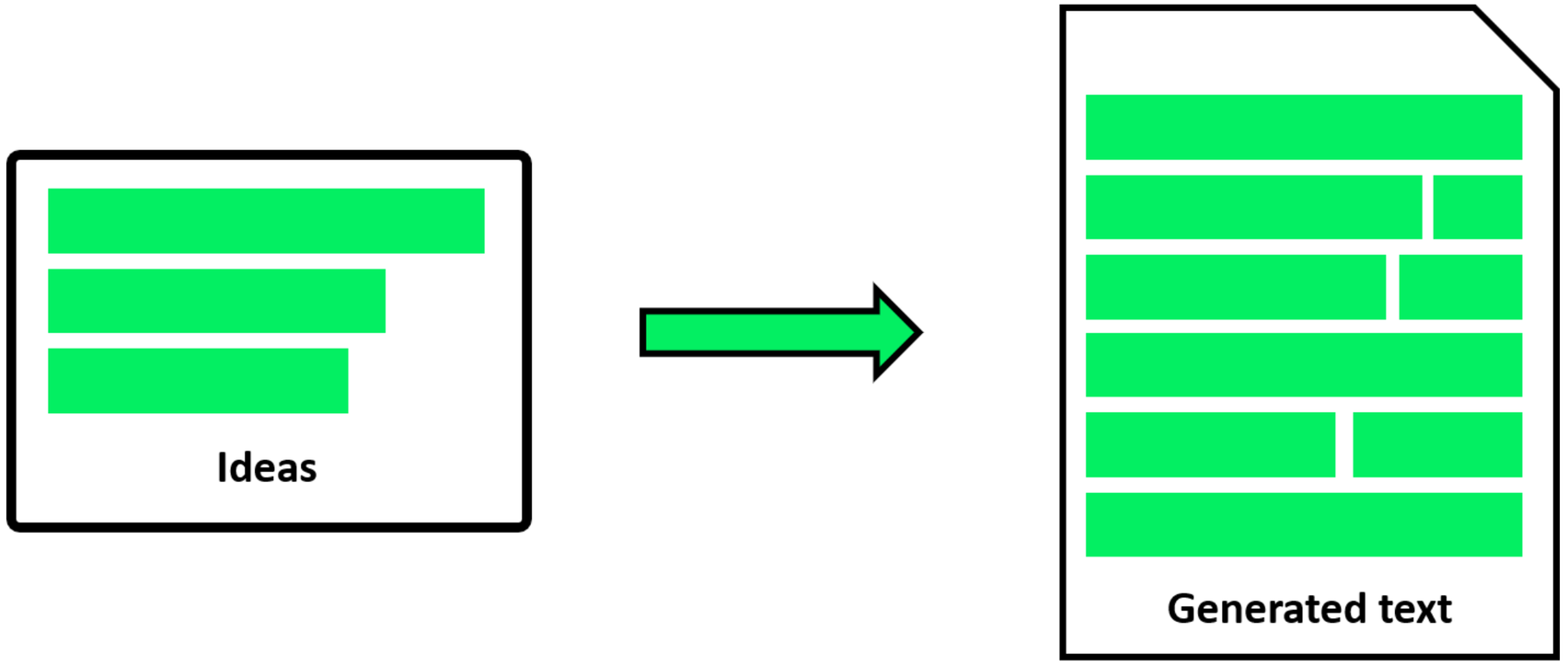
prompt = f"""Summarize the review delimited by triple backticks, in three sentences,
focusing on the key features and user experience:
            ```{text}```"""
print(get_response(prompt))
```
```

Effective prompt: summarization focus

```
The customer purchased the XYZ Smart Watch and was impressed with its sleek design
and build quality.
They found it comfortable to wear and versatile enough to match any outfit.
The touch screen was responsive and user-friendly, making it easy to navigate
through the watch's features.
```

Text expansion

- Generates text from ideas or bullet points
- Improves efficiency and productivity
- LLMs can expand text with well-crafted prompts



Text expansion prompts

- Ask model to expand delimited text
- Highlight aspects to focus on
- Provide output requirements (tone, length, structure, audience)

Expanding service description

```
service_description = """Service: Social XYZ
- Social Media Strategy Development
- Content Creation and Posting
- Audience Engagement and Community Building
- Increased Brand Visibility
- Enhanced Customer Engagement
- Data-Driven Marketing Decisions"""
```

```
prompt = f"""Expand the description for the Social XYZ service delimited by triple
backticks to provide an overview of its features and benefits, without bypassing
the limit of two sentences. Use a professional tone.
```{service_description}```"""
print(get_response(prompt))
```
```

Expanding service description

```
Social XYZ is a comprehensive social media service that offers strategic
development, content creation, and posting to help businesses effectively engage
with their target audience and build a strong online community.

With a focus on increasing brand visibility and enhancing customer engagement,
Social XYZ enables businesses to make data-driven marketing decisions for optimal
results.
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API

Text transformation

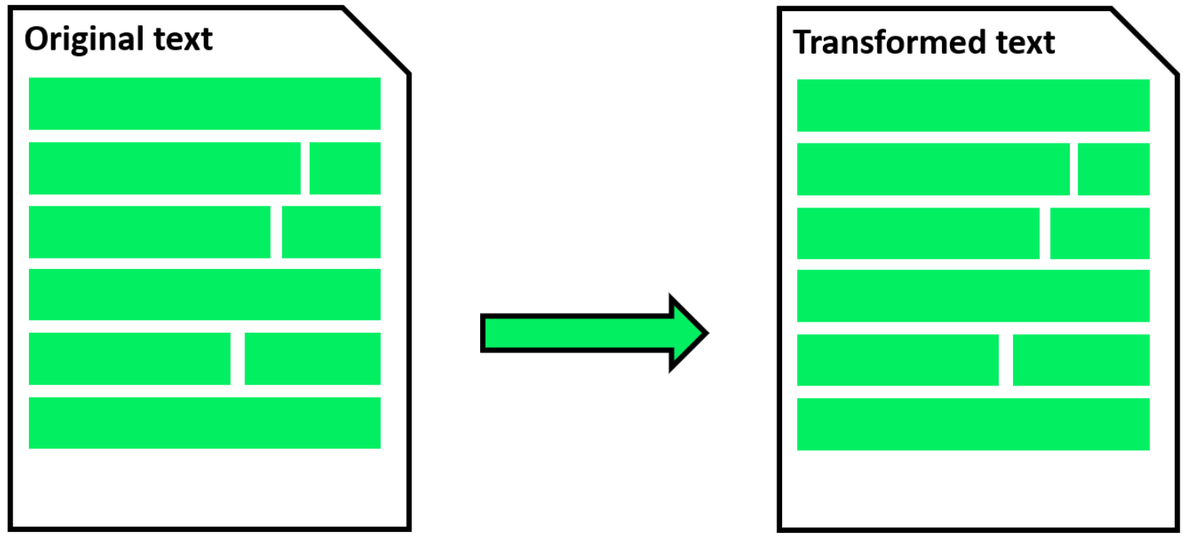
PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

datacamp

Text transformation

- Transforms given text to create a new text
- Has many applications
  - Language translation
  - Tone adjustments
  - Writing improvement



Language translation

- Specify input and output language in prompt

```
text = "XYZ Scooter is a cutting-edge electric scooter designed for urban
adventurers. Its sleek design, long-lasting battery, and smart connectivity offer
a seamless and eco-friendly way to navigate city streets."
prompt = f"""Translate the English text delimited by triple backticks to French:
            ```{text}```"""
print(get_response(prompt))
```

```
XYZ Scooter est une trottinette électrique de pointe conçue pour les aventuriers
urbains. Avec son design élégant, sa batterie longue durée et sa connectivité
intelligente, elle offre un moyen fluide et respectueux de l'environnement de se
déplacer dans les rues de la ville.
```
```

If text language is unknown ...

```
text = "Das Produkt ist wirklich schön, und der Preis ist fair."
```

Ask the model to specify text language

```
prompt = f"""Tell me which language is the text delimited by triple backticks:
            ```{text}```"""
print(get_response(prompt))
```

```
The text delimited by triple backticks (```) is in the German language.
```
```

Multilingual translation

- Translate to multiple languages simultaneously

```
text = "The product combines top quality with a fair price."
prompt =  f"""Translate the English text delimited by triple backticks to French,
Spanish, and German:
```{text}```"""
```

```
French: Le produit allie une qualité supérieure à un prix équitable.
Spanish: El producto combina una calidad superior con un precio justo.
German: Das Produkt vereint Spitzenqualität mit einem fairen Preis.
```

- **Note:** Model translations need to be verified if customers are involved
```

Tone adjustment

- Re-write text in a different tone

```
text = "Hey there! Check out our awesome summer deals! They're super cool, and you
won't want to miss them. Grab 'em now!"

prompt = f"""Write the text delimited by triple backticks using a formal and
persuasive tone:
```{text}```"""

print(get_response(prompt))
```
```

Tone adjustment

```
Dear [Recipient's Name],

We cordially invite you to explore our exceptional summer offers! These exclusive deals are truly extraordinary and should not be missed. Time is limited, so we encourage you to seize these incredible opportunities promptly.

[...]

Don't miss out on this chance! Take swift action to secure these remarkable summer deals before they expire. You won't find a more enticing selection of offers elsewhere.

[...]
```

Tone adjustment: specify audience

```
text = "Our cutting-edge widget employs state-of-the-art microprocessors and
advanced algorithms, delivering unparalleled efficiency and performance for a wide
range of applications."

prompt = f"""Write the text delimited by triple backticks to be suitable for
a non-technical audience:
```{text}```"""

print(get_response(prompt))
```

```
Our advanced widget uses the latest technology to provide exceptional efficiency
and performance for a variety of uses.
```
```

Grammar and writing improvements

- Correct spelling, grammar, and punctuation mistakes without modifying other aspects

```
text = "Dear sir, I wanted too discus a potentiel opportunity for colabration.
Pls let me know wen you're avaliable."
```

```
prompt = f"""Proofread the text delimited by triple backticks while keeping the
original text structure intact:
            ```{text}```"""
print(get_response(prompt))
```

```
Dear sir, I wanted to discuss a potential opportunity for collaboration. Please
let me know when you are available.
```
```

Grammar and writing improvements

- Enhance clarity by modifying text structure

```
text = "Dear sir, I wanted too discus a potentiel opportunity for colabration.
Pls let me know wen you're avaliable."
```

```
prompt = f"""Proofread and restructure the text delimited by triple backticks for
enhanced readability, flow, and coherence:
            ```{text}```"""
print(get_response(prompt))
```

```
Dear sir,

I wanted to discuss with you a potential opportunity for collaboration.
Please let me know your availability.

Thank you.
```
```

Multiple transformations

- Ask for multiple transformations at once -> multi-step prompts

```
text = "omg, I cant believe how awesome this product is! Its like the best thing ever! You guys gotta try it out!"
prompt = f"""Transform the text delimited by triple backticks with the following two steps:
             Step 1 - Proofread it without changing its structure
             Step 2 - Change the tone to be professional
           ```{text}```"""
response = get_response(prompt)
```

```
Step 1 - OMG, I can't believe how awesome this product is! It's like the best thing ever! You guys gotta try it out!
Step 2 - I am astounded by the exceptional qualities of this product. It represents a remarkable solution that surpasses expectations. I highly recommend trying it out, as its benefits are truly impressive.
```
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API

Text analysis

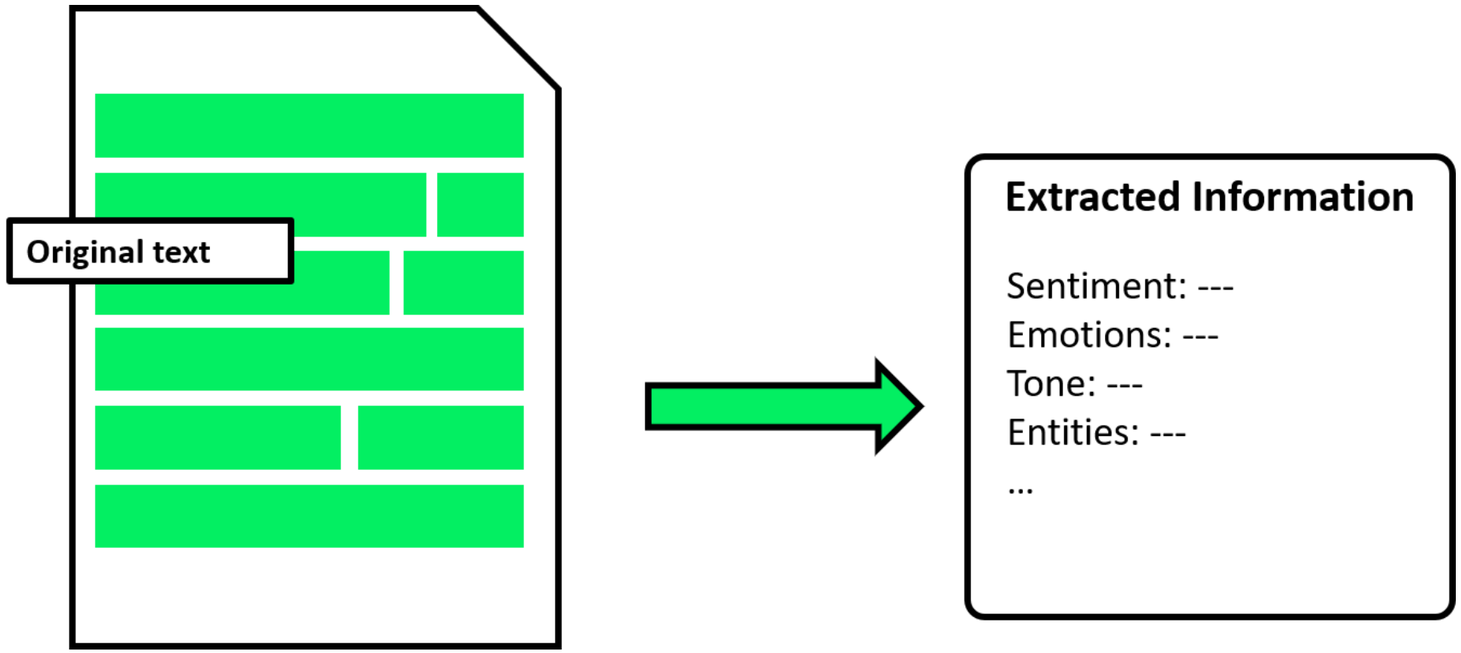
PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

datacamp

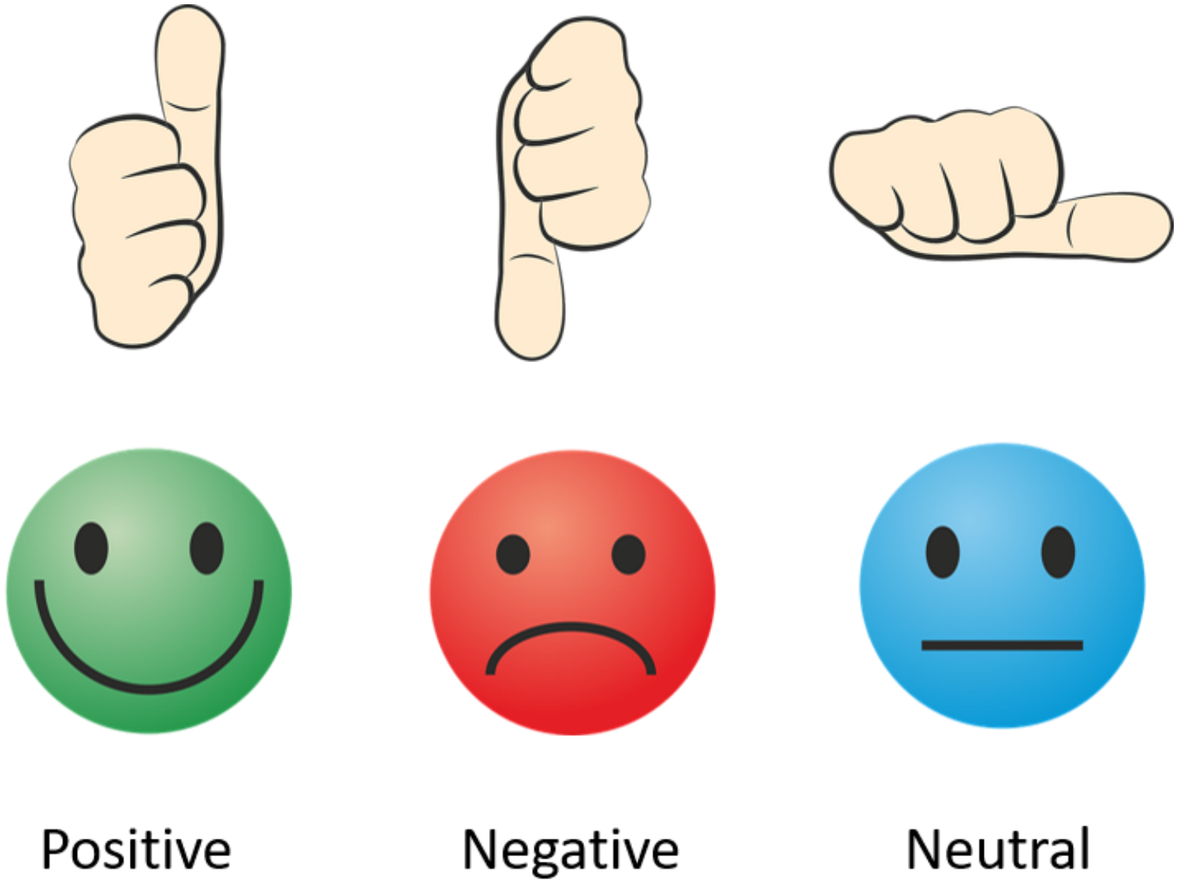
Text analysis

- Examining text to extract information
  - Text classification
  - Entity extraction
- Companies should seek legal advice when using customer data



Text classification

- Assign categories to text
- Example: sentiment analysis



Specified categories

- Specify classification categories when known
- Mention output requirements

```
text = "I bought your XYZ Smart Watch and wanted to share my positive experience.
Impressed with its sleek design, comfort, and touchscreen usability."

prompt = f"""Classify the sentiment of the text delimited by triple backticks as
positive, negative, or neutral. Give your answer as a single word:
            ```{text}```"""

print(get_response(prompt))
```

```
positive
```
```

Unspecified categories

- Model uses its knowledge when categories are not specified

```
text = "I bought your XYZ Smart Watch and wanted to share my positive experience.
Impressed with its sleek design, comfort, and touchscreen usability."

prompt = f"""Classify the sentiment of the text delimited by triple backticks.
            Give your answer as a single word.
          ```{text}```"""
print(get_response(response))
```

```
positive.
```

- For some open-ended problems this might not work well
```

Multiple classes

- Text can fit into multiple classes
- Define a maximum number of classes if they are not known

```
text = "I bought your XYZ Smart Watch and wanted to share my positive experience.
Impressed with its sleek design, comfort, and touchscreen usability."
prompt = f"""Identify emotions used in this text. Don't use more than 3 emotions.
            Format your answer as a list of words separated by commas:
            ```{text}```"""

print(get_response(prompt))
```

```
impressed, positive, comfortable
```
```

Entity extraction

- Extracting specific entities from text
- Examples: names, places, organizations, dates

Entity extraction: specify entities

- Specify entities to extract
- Mention output format

```
text = "The XYZ Mobile X200: a sleek 6.5-inch Super AMOLED smartphone with a 48MP
triple-camera, octa-core processor, 5000mAh battery, 5G connectivity, and Android
11 OS. Secure with fingerprint and facial recognition. 128GB storage, expandable up
to 512GB."
prompt = f"""Identify the following entities from the text delimited by triple backticks
            - Product Name



            ```{text}```"""
print(get_response(prompt))
```
```

Entity extraction: specify entities

- Specify entities to extract
- Mention output format

```
text = "The XYZ Mobile X200: a sleek 6.5-inch Super AMOLED smartphone with a 48MP
triple-camera, octa-core processor, 5000mAh battery, 5G connectivity, and Android
11 OS. Secure with fingerprint and facial recognition. 128GB storage, expandable up
to 512GB."
prompt = f"""Identify the following entities from the text delimited by triple backticks
             - Product Name
             - Display Size


            ```{text}```"""
print(get_response(prompt))
```
```

Entity extraction: specify entities

- Specify entities to extract
- Mention output format

```
text = "The XYZ Mobile X200: a sleek 6.5-inch Super AMOLED smartphone with a 48MP
triple-camera, octa-core processor, 5000mAh battery, 5G connectivity, and Android
11 OS. Secure with fingerprint and facial recognition. 128GB storage, expandable up
to 512GB."
prompt = f"""Identify the following entities from the text delimited by triple backticks
            - Product Name
            - Display Size
            - Camera Resolution

          ```{text}```"""
print(get_response(prompt))
```
```

Entity extraction: specify entities

- Specify entities to extract
- Mention output format

```
text = "The XYZ Mobile X200: a sleek 6.5-inch Super AMOLED smartphone with a 48MP
triple-camera, octa-core processor, 5000mAh battery, 5G connectivity, and Android
11 OS. Secure with fingerprint and facial recognition. 128GB storage, expandable up
to 512GB."
prompt = f"""Identify the following entities from the text delimited by triple backticks
            - Product Name
            - Display Size
            - Camera Resolution
            Format the answer as an unordered list.
          ```{text}```"""
print(get_response(prompt))
```
```

Entity extraction: specify entities

```
Product Name: XYZ Mobile X200
Display Size: 6.5-inch
Camera Resolution: 48MP triple-camera
```

Entity extraction with few-shot prompts

- For complex structures

```
ticket_1 = "Hello, I'm Emma Adams. I'd
like to ask about my reservation with
the code CAR123.
You can reach me at +123456 if needed."

ticket_2 = "This is Sarah Williams.
I would like to request some information
regarding my upcoming flight with
reservation code FLIGHT987. Thank you."
```

```
entities_1 = """
* Customer Details:
  - Name: Emma Adams
  - Phone: +123456
* Reservation Details:
  - Reservation Code: CAR123"""
entities_2 = """
* Customer Details:
  - Name: Sarah Williams
* Reservation Details:
  - Reservation Code: FLIGHT987"""
```

Entity extraction with few-shot prompts

```
ticket_3 = "Hello, I'm David Brown (CUST123). I need assistance with my reservation under
the code HOTEL456. There are some questions and issues related to my upcoming stay that
require your attention."

prompt = f"""Text: {ticket_1} -> Entities: {entities_1}
             Text: {ticket_2} -> Entities: {entities_2}
             Text: {ticket_3} -> Entities: """
print(get_response(prompt))
```

```
* Customer Details:
  - Name: David Brown
  - Customer ID: CUST123
* Reservation Details:
  - Reservation Code: HOTEL456
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API

Code generation and explanation

PROMPT ENGINEERING WITH THE OPENAI API

Fouad Trad
Machine Learning Engineer

datacamp

Code generation

- Creating source code to solve a given problem
- Essential in any domain using software solutions
- Understanding generated code is essential for effective use

Code generation prompts

- Problem description
- Programming language
- Format (script, function, class)

```
prompt = "Write a Python function that accepts a list of quarterly sales, and outputs the average sales per quarter."
print(get_response(prompt))
```

```
def calculate_average_sales(quarterly_sales):
    total_sales = sum(quarterly_sales)
    average_sales = total_sales / len(quarterly_sales)
    return average_sales
```

Input-output examples

- Give model input-output examples to generate a program that maps them

```
examples = """
Input: [150000, 180000, 200000, 170000] -> Output: 175000.0
Input: [10000, 25000, 30000, 15000] -> Output: 20000.0
Input: [50000, 75000, 60000, 45000] -> Output: 57500.0
"""
prompt = f"""You are provided with input-output examples delimited by triple
backticks for a python program that receives a list of quarterly sales data.
Write code for this program.
```{examples}```"""
print(get_response(prompt))
```
```

Input-output examples

```
Based on the provided input-output examples, it seems like you want to calculate the average of
quarterly sales data for each set of input values.

Here's a Python program that does that:

def calculate_average_quarterly_sales(sales_data):
    total_sales = sum(sales_data)
    num_quarters = len(sales_data)
    average_sales = total_sales / num_quarters
    return average_sales

print(calculate_average_quarterly_sales([150000, 180000, 200000, 170000]))  # Output: 175000.0
print(calculate_average_quarterly_sales([10000, 25000, 30000, 15000])) # Output: 20000.0
print(calculate_average_quarterly_sales([50000, 75000, 60000, 45000]))  # Output: 57500.0
```

Code modification

- Ask model to modify code according to requirements

```
script = """
quarterly_sales = [150, 180, 200, 170]
total_sales = sum(quarterly_sales)
print("Total sales: ", total_sales)
"""
prompt = f"""Modify the script delimited by triple backticks to a function that we
can call to compute the total sales given quarterly sales
```{script}```"""
print(get_response(prompt))
```
```

Code modification

```
def compute_total_sales(quarterly_sales):
    total_sales = sum(quarterly_sales)
    return total_sales

quarterly_sales = [150, 180, 200, 170]
total_sales = compute_total_sales(quarterly_sales)
print("Total sales: ", total_sales)
```

Multiple code modifications

```
script = """
quarterly_sales = [150, 180, 200, 170]
total_sales = sum(quarterly_sales)
print("Total sales: ", total_sales)
"""
prompt = f"""Modify the script delimited by triple backticks as follows:
            - Let user input parameters interactively


            ```{script}```"""
print(get_response(prompt))
```
```

Multiple code modifications

```
script = """
quarterly_sales = [150, 180, 200, 170]
total_sales = sum(quarterly_sales)
print("Total sales: ", total_sales)
"""
prompt = f"""Modify the script delimited by triple backticks as follows:
            - Let user input parameters interactively
            - Make sure to verify inputs are positive, otherwise, display a
            message for the user, and ask them to provide their input again.
          ```{script}```"""
print(get_response(prompt))
```
```

Multiple code modifications

```
quarterly_sales = []

for i in range(4):
    while True:
        sales = float(input("Enter quarterly sales for quarter {}: ".format(i+1)))
        if sales <= 0:
            print("Sales must be positive. Please try again.")
            continue
        quarterly_sales.append(sales)
        break

total_sales = sum(quarterly_sales)
print("Total sales: ", total_sales)
```

Code explanation

- Code can be difficult to interpret
- LLMs can be used to explain code

Code explanation requirements

```
code = """
def compute_average_sales_per_quarter(quarterly_sales):
    average_sales = sum(quarterly_sales) / len(quarterly_sales)
    return average_sales
"""
```

- Specify length of explanation

```
prompt = f"""Explain in one sentence what the code delimited by triple backticks does
            ```{code}```"""
print(get_response(prompt))
```

```
The code calculates the average sales per quarter by summing up the values in the
"quarterly_sales" list and dividing the sum by the number of elements in the list.
```
```

Detailed code explanation

```
code = """
def compute_average_sales_per_quarter(quarterly_sales):
    average_sales = sum(quarterly_sales) / len(quarterly_sales)
    return average_sales
"""
```

- Detailed explanation with chain-of-thought prompt

```
prompt = f"""Explain what the code delimited by triple backticks does.
Let's think step by step.
```{code}```"""
print(get_response(prompt))
```
```

Detailed code explanation

```
The code defines a function called `compute_average_sales_per_quarter` that takes in a list of `quarterly_sales` as a parameter.

Inside the function, it calculates the average sales by summing up all the values in the `quarterly_sales` list using the `sum()` function, and then dividing it by the length of the list using the `len()` function.

The result is stored in the variable `average_sales`.

Finally, the function returns the calculated average sales value.

This code snippet can be used to calculate the average sales per quarter given a list of quarterly sales data.
```

Let's practice!

PROMPT ENGINEERING WITH THE OPENAI API