Summarizing and editing text

WORKING WITH THE OPENAI API

James Chapman
Curriculum Manager, DataCamp

datacamp

Recap...

- Q&A

```
response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user", "content": "How many days are in October?"}]
)

print(response.choices[0].message.content)
```

```
October has 31 days.
```

Text editing

- **Example:** updating the name, pronouns, and job title

```
prompt = """
Update name to Maarten, pronouns to he/him, and job title to Senior Content Developer
in the following text:

Joanne is a Content Developer at DataCamp. Her favorite programming language is R,
which she uses for her statistical analyses.
"""
```

Text editing

```
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": prompt}]
)

print(response.choices[0].message.content)
```

```
Maarten is a Senior Content Developer at DataCamp. His favorite programming language is R, which he uses for his statistical analyses.
```

Text summarization

- **Example:** summary of customer chat transcripts

```
text = """
Customer: Hi, I'm trying to log into
my account, but it keeps saying
my password is incorrect. I'm sure
I'm entering the right one.

Support: I'm sorry to hear that!
Have you tried resetting your password?
...
"""
```

Text summarization

```
prompt = f"""Summarize the customer support chat
             in three concise key points: {text}"""

response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user", "content": prompt}]
)
print(response.choices[0].message.content)
```

```
1. Customer couldn't log in due to a password issue and missing reset link.
2. Support resent the reset email after confirming it was sent.
3. Customer resolved the issue by using Google sign-in.
```

Controlling response length

- `max_completion_tokens = 5`

```
response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user",
  "content": "Write a haiku about AI."}],
  max_completion_tokens=5
)
```

```
AI so powerful
Computers
```

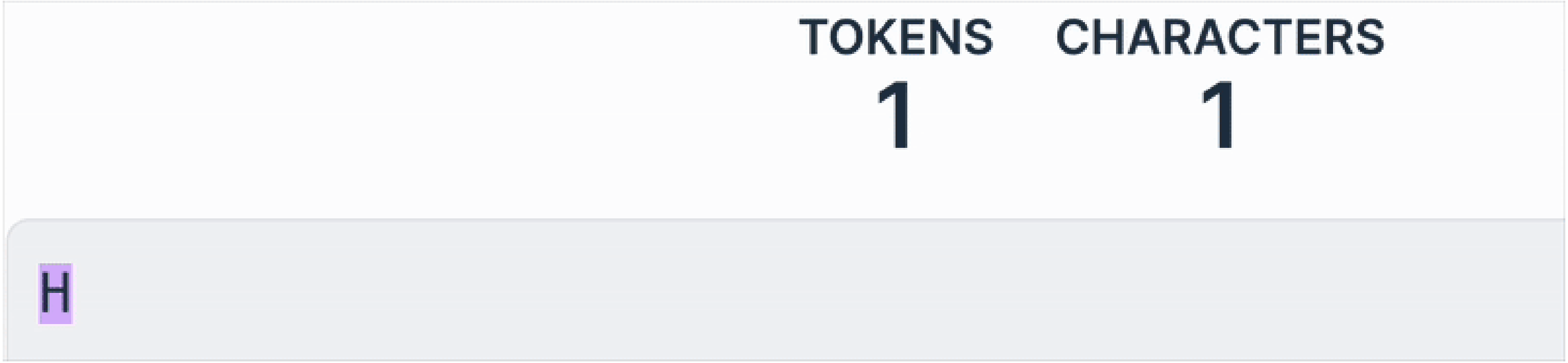
- `max_completion_tokens = 30`

```
response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user",
  "content": "Write a haiku about AI."}],
  max_completion_tokens=30
)
```

```
A machine mind thinks
Logic dictates its choices
Mankind ponders anew
```

Understanding tokens

- **Tokens:** units of text that help the AI understand and interpret text



[1] https://platform.openai.com/tokenizer

Calculating the cost

- Usage costs dependent on the **model** and the number of **tokens**
  - Models are priced by *cost/tokens*
  - Input and output tokens may have different costs
- Increasing `max_completion_tokens` increases cost

Calculating the cost

```
prompt = f"""Summarize the customer support chat
             in three concise key points: {text}"""

max_completion_tokens = 500

response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user", "content": prompt}],
  max_completion_tokens=max_completion_tokens
  )
```

Calculating the cost

```
# Define price per token
input_token_price = 0.15 / 1_000_000
output_token_price = 0.6 / 1_000_000
# Extract token usage
input_tokens = response.usage.prompt_tokens
output_tokens = max_completion_tokens
# Calculate cost
cost = (input_tokens * input_token_price + output_tokens * output_token_price)
print(f"Estimated cost: ${cost}")
```

```
Estimated cost: $0.00124
```

[1] https://openai.com/pricing

Let's practice!

WORKING WITH THE OPENAI API

Text generation

WORKING WITH THE OPENAI API

James Chapman
Curriculum Manager, DataCamp

datacamp

How is the output generated?

- Text *most likely* to complete the prompt

```
response = client.chat.completions.create(
  model="gpt-4o-mini",
  messages=[{"role": "user", "content": "Life is like a box of chocolates."}]
)

print(response.choices[0].message.content)
```

```
You never know what you're going to get. This famous quote from the movie
"Forrest Gump"...
```

- Response is **non-deterministic** (inherently random)

Controlling response randomness

- `temperature` : control on *determinism*
- Ranges from `0` (highly deterministic) to `2` (very random)

```
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": "Life is like a box of chocolates."}],
    temperature=2
)
print(response.choices[0].message.content)
```

```
"...you never know what you're gonna get." That quote reminds us of the
unpredictability of life and the diverse set of experiences we might encounter.
Whether sweet, nutty, bitter, or flashy, life holds a treasure trove of surprises.
```

Text generation: marketing

```
prompt = "Generate a powerful tagline for a new electric vehicle
that highlights innovation and sustainability."
```

```
response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": prompt}]
)

print(response.choices[0].message.content)
```

```
"Drive the Future - Electric, Effortless, Extraordinary."
```

Text generation: product description

```
prompt = """Write a compelling product description for the UltraFit Smartwatch.
Highlight its key features:  10-day battery life, 24/7 heart rate and sleep
tracking, built-in GPS, water resistance up to 50 meters, and lightweight design.

Use a persuasive and engaging tone to appeal to fitness enthusiasts
and busy professionals.
"""

response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": prompt}]
)
```

Text generation: product description

```
print(response.choices[0].message.content)
```

```
"The UltraFit Smartwatch is your all-in-one health and fitness companion.
Stay on top of your wellness with 24/7 heart rate and sleep tracking,
while the built-in GPS keeps you on course during runs.
With a 10-day battery life, you can track your progress without constant recharging.
Designed for both workouts and daily wear, its  lightweight build and 50-meter
water resistance make it the perfect smartwatch for an active lifestyle."
```

- **Iterate!** Add more details to the prompt and re-run the request
- Next video → providing examples to the model

Let's practice!

WORKING WITH THE OPENAI API

Shot prompting

WORKING WITH THE OPENAI API

James Chapman
Curriculum Manager, DataCamp

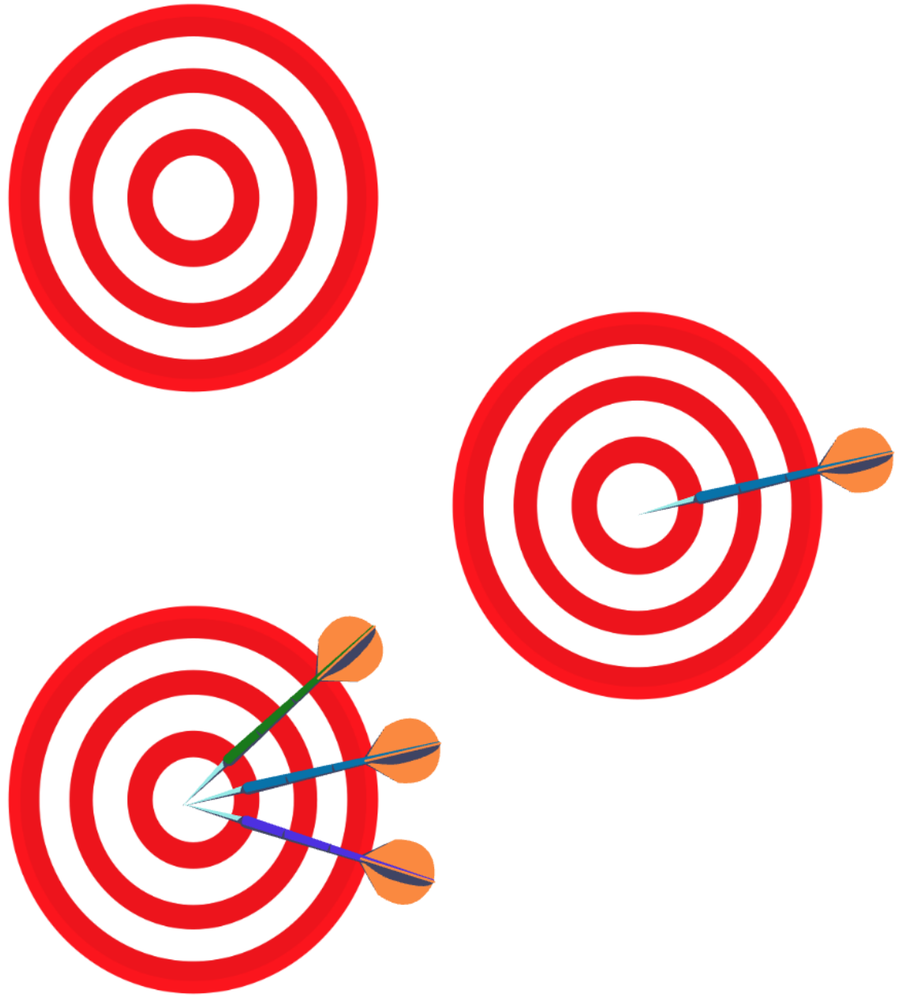
datacamp

Providing examples

- Shot prompting

What is shot prompting?

Shot prompting: including examples to guide AI responses



- **Zero-shot:** no examples, just instructions
- **One-shot:** one example guides the response
- **Few-shot:** multiple examples provide more context

Why does shot prompting matter?

Use cases of shot prompting:

- Text classification
- Sentiment analysis
- Data extraction
- ... and many more!

Zero-shot prompting

```
prompt = """Classify sentiment as 1-5 (bad-good) in the following statements:
1. Meal was decent, but I've had better.
2. My food was delayed, but drinks were good.
"""


...
```

```
1. Meal was decent, but I've had better. - 3 (Neutral)
2. My food was delayed, but drinks were good. - 3 (Neutral)
```

One-shot prompting

```
prompt = """Classify sentiment as 1-5 (bad-good) in the following statements:
1. The service was very slow -> 1
2. Meal was decent, but I've had better. ->
3. My food was delayed, but drinks were good. ->
"""
```

```
1. The service was very slow -> 1
2. Meal was decent, but I've had better. -> 3
3. My food was delayed, but drinks were good. -> 4
```

Few-shot prompting

```
prompt = """Classify sentiment as 1-5 (bad-good) in the following statements:
1. The service was very slow -> 1
2. The steak was awfully good! -> 5
3. It was ok, no massive complaints. -> 3
4. Meal was decent, but I've had better. ->
5. My food was delayed, but drinks were good. ->
"""
```

```
1. The service was very slow -> 1
2. The steak was awfully good! -> 5
3. It was ok, no massive complaints. -> 3
4. Meal was decent, but I've had better. -> 4
5. My food was delayed, but drinks were good. -> 3
```

General categorization

```
prompt = """Classify the following animals as Land, Sea, or Both:
1. Blue whale
2. Polar bear
3. Salmon
4. Dog
"""
```

```
1. **Blue whale** - Sea
2. **Polar bear** - Both (primarily land but is a marine mammal that lives in sea ice)
3. **Salmon** - Both (spends part of its life in freshwater and part in the ocean)
4. **Dog** - Land
```

```
prompt = """Classify the following animals as Land, Sea, or Both:
1. Zebra = Land
2. Crocodile = Both
3. Blue whale =
4. Polar bear =
5. Salmon =
6. Dog =
"""
```

```
1. Zebra = Land
2. Crocodile = Both
3. Blue whale = Sea
4. Polar bear = Both
5. Salmon = Sea
6. Dog = Land
```

Let's practice!

WORKING WITH THE OPENAI API