FewShotExamples

September 16, 2024

1 Setting up the environment

```
[1]: # !pip install openai

import os
from openai import OpenAI

from dotenv import load_dotenv, find_dotenv

# Read the local .env file, containing the Open AI secret key.
_ = load_dotenv(find_dotenv())

client = OpenAI(api_key = os.environ['OPENAI_API_KEY'])
```

2 Zero-Shot scenario

What is zero-shot? Zero-shot is the prompting style we are most used to, where we do not give examples to the LLM. This is useful for tasks where the task is well-specified in English.

So what is happening with this prompt? The problem is that the LLM may not have an idea of what a “character” is in a children’s story, since inanimate objects are very often given sentience in this genre. It may be confused and consider them a character, when they are not described as such.

```
[6]: prompt = 'Given the short story below, extract a graph of all the characters, '\
            'where characters are represented as nodes, and edges represent␣
 ↪relationships that are '\
            'expressed in the text. The graph should be output in JSON as a list of␣
 ↪(node1, node2, relationship)'\
            ' triplets.'

test_story = 'Short Story: "There was once a hare who was friends with a␣
 ↪tortoise. One day, he '\
            'challenged the tortoise to a race. Seeing how slow the tortoise was␣
 ↪going, the hare thought he\'d win'\
            ' this easily. So, he took a nap while the tortoise kept on going. When␣
 ↪the hare woke, he saw that the '\
        'tortoise was already at the finish line. Much to his chagrin, the␣
↪tortoise won the race while he was '\
        'busy sleeping."'

messages = [
    {
        'role': 'user',
        'content': prompt + '\n\n' + test_story
    }
]

response = client.chat.completions.create(
    model = "gpt-4",# "gpt-3.5-turbo",
    messages = messages,
    temperature = 0,
    max_tokens = 500
)

print(messages[0]['content'])
print()
print("Zero-shot response:")
print(response.choices[0].message.content)
```

```
Given the short story below, extract a graph of all the characters, where
characters are represented as nodes, and edges represent relationships that are
expressed in the text. The graph should be output in JSON as a list of (node1,
node2, relationship) triplets.

Short Story: "There was once a hare who was friends with a tortoise. One day, he
challenged the tortoise to a race. Seeing how slow the tortoise was going, the
hare thought he'd win this easily. So, he took a nap while the tortoise kept on
going. When the hare woke, he saw that the tortoise was already at the finish
line. Much to his chagrin, the tortoise won the race while he was busy
sleeping."

Zero-shot response:
[
  {
    "node1": "hare",
    "node2": "tortoise",
    "relationship": "friends"
  },
  {
    "node1": "hare",
    "node2": "tortoise",
    "relationship": "challenged to a race"
  },
  {
    "node1": "hare",
    "node2": "tortoise",
    "relationship": "underestimated"
  },
  {
    "node1": "hare",
    "node2": "tortoise",
    "relationship": "lost race to"
  }
]
```

3 Few-Shot examples

Few-shot is a prompting style where examples are given for the LLM to copy. This is also referred to as "in-context learning", since the model appears to improve. Improvement given experiences is called learning in machine learning contexts, but this learning does not change parameters so it is given a name to distinguish itself.

Parameters are, very simply, numbers that represent what the model has learned about its data. We will go more in-depth later in the course.

[8]:
```
# 'content': prompt + '\n\n' + shot1 + '\n\n' + shot2 + '\n\n' + test_story
# one cell for 0-shot, one for 2-shot

prompt = 'Given the short story below, extract a graph of all the characters, '\
        'where characters are represented as nodes, and edges represent␣
 ↪relationships that are '\
        'expressed in the text. The graph should be output in JSON as a list of␣
 ↪(node1, node2, relationship)'\
        ' triplets.'

shot1 = 'For example, if the short story is:\n" Red Riding Hood had lunch at␣
 ↪her '\
        'grandmother\'s place. Then she parted ways with her grandmother and␣
 ↪walked into the woods.'\
        ' She met and talked with the Wolf, who followed her deep into the␣
 ↪forest.",\nthen the '\
        'output should be \n[\n["Red Riding Hood", "grandmother", "parted ways␣
 ↪with"],\n["Red Riding '\
        'Hood", "Wolf", "met"],\n["Red Riding Hood", "Wolf", "talked␣
 ↪with"],\n["Wolf", "Red Riding Hood", '\
        '"followed into the forest"],\n]'
shot2 = 'If the short story is:\n"The raccoon brushed his teeth at 5pm, while␣
 ↪the '\
        'elephant got ready at 6pm. The raccoon met with the elephant at 7pm␣
 ↪for dinner. The elephant '\
        'got a steak, the raccoon got mashed potatoes. The elephant went home␣
  ↪at 9pm, the raccoon went home '\
        'at 10pm. Later, the elephant made a phone call to the raccoon.",\nthen␣
  ↪the output should be:\n[\n'\
        '["raccoon","elephant", "met at 7pm for dinner"]\n["elephant,␣
  ↪"raccoon", "made a phone call to"]\n]'

final_story = 'Now output '\
        'the graph for this short story: "There was once a hare who was friends␣
  ↪with a tortoise. One day, he '\
        'challenged the tortoise to a race. Seeing how slow the tortoise was␣
  ↪going, the hare thought he\'d win'\
        ' this easily. So, he took a nap while the tortoise kept on going. When␣
  ↪the hare woke, he saw that the '\
        'tortoise was already at the finish line. Much to his chagrin, the␣
  ↪tortoise won the race while he was '\
        'busy sleeping."'

messages = [
    {
        'role': 'user',
        'content': prompt + '\n\n' + shot1 + '\n\n' + shot2 + '\n\n' +␣
  ↪final_story
    }
]

response = client.chat.completions.create(
    model = "gpt-4",#3.5-turbo",
    messages = messages,
    temperature = 0,
    max_tokens = 500
)

print(messages[0]['content'])
print()
print("2-shot response:")
print(response.choices[0].message.content)
```

```
Given the short story below, extract a graph of all the characters, where
characters are represented as nodes, and edges represent relationships that are
expressed in the text. The graph should be output in JSON as a list of (node1,
node2, relationship) triplets.

For example, if the short story is:
" Red Riding Hood had lunch at her grandmother's place. Then she parted ways
with her grandmother and walked into the woods. She met and talked with the
Wolf, who followed her deep into the forest.",
then the output should be
[
["Red Riding Hood", "grandmother", "parted ways with"],
["Red Riding Hood", "Wolf", "met"],
["Red Riding Hood", "Wolf", "talked with"],
["Wolf", "Red Riding Hood", "followed into the forest"],
]

If the short story is:
"The raccoon brushed his teeth at 5pm, while the elephant got ready at 6pm. The
raccoon met with the elephant at 7pm for dinner. The elephant got a steak, the
raccoon got mashed potatoes. The elephant went home at 9pm, the raccoon went
home at 10pm. Later, the elephant made a phone call to the raccoon.",
then the output should be:
[
["raccoon","elephant", "met at 7pm for dinner"]
["elephant, "raccoon", "made a phone call to"]
]

Now output the graph for this short story: "There was once a hare who was
friends with a tortoise. One day, he challenged the tortoise to a race. Seeing
how slow the tortoise was going, the hare thought he'd win this easily. So, he
took a nap while the tortoise kept on going. When the hare woke, he saw that the
tortoise was already at the finish line. Much to his chagrin, the tortoise won
the race while he was busy sleeping."

2-shot response:
[
["hare", "tortoise", "was friends with"],
["hare", "tortoise", "challenged to a race"],
["hare", "tortoise", "saw at the finish line"],
["tortoise", "hare", "won the race while was sleeping"]
]
```

[ ]: