assignment2

September 9, 2024

1 Getting started with Generative-AI

Submitted By: Jenish Twayana

Submitted Date: 2nd September, 2024

1.1 Exercise 2: Few-shot Prompting and Evaluation

1.1.1 Set up the client and function

[1]:
```
#importing necessary packages
from openai import OpenAI
from nltk.translate.bleu_score import sentence_bleu
import json
import os
```

[2]:
```
#initialize openai client
openai_api_key = os.getenv("OPENAI_API_KEY")
client = OpenAI(api_key=openai_api_key)
model = "gpt-3.5-turbo"
```

[3]:
```
#output parser
def parse_output(ai_message: str) -> str:
    """Parse the AI message."""
    return ai_message.choices[0].message.content
```

[4]:
```
#function for calling api
def get_completion(system_prompt, user_prompt, model="gpt-3.5-turbo"):
    messages = [
    {"role": "system", "content": system_prompt},
    {"role": "user", "content": user_prompt}
    ]
    response = client.chat.completions.create(
    model=model,
    messages=messages,
    temperature=0,
    )
    return response
```

1.1.2 Task 1 (Text Classification):

Create a few-shot prompt that classifies given statements into one of three categories: factual, opinion, or ambiguous. Note that 'factual' does not necessarily mean 'true.' Display the output for each test case in JSON format. Ensure that the test cases are not included in the prompt to avoid data leakage.

[5]:
```
#desired response for text classification in valid JSON format
text_classification_response_example = '''
<examples>
<example1>
Prompt: "The Earth is the third planet from the Sun."
Desired Response in valid JSON format: {
"text": "The Earth is the third planet from the Sun.",
"label": "Factual"
}
</example1>
<example2>
Prompt: "Swimming is fun."
Desired Response in valid JSON format: {
"text": "Swimming is fun.",
"label": "Opinion"
}
</example2>
<example3>
Prompt: "He might not come early."
Desired Response in valid JSON format: {
"text": "He might not come early.",
"label": "Ambiguous"
}
</example3>
</examples>
'''
```

[6]:
```
#system prompt for text classification
text_classification_task_system_prompt = f'''
Classify the following text into one of the following categories: factual,␣
 ↪opinion or ambiguous.
Here are the few examples which are delimited by <examples></examples>:
{text_classification_response_example}
'''
```

[7]:
```
# test dataset for text classification
text_classification_data = [
    ("The capital of France is Paris.", "Factual"),
    ("I believe chocolate is the best dessert.", "Opinion"),
    ("It might rain tomorrow.", "Ambiguous"),
    ("Mount Everest is the highest mountain in the world.", "Factual"),
```

```
    ("In my opinion, summer is the best season.", "Opinion"),
    ("She may arrive at the party tonight.", "Ambiguous"),
    ("Pizza is delicious.", "Opinion"),
    ("The Earth orbits around the Sun.", "Factual"),
    ("Sunsets are beautiful.", "Opinion"),
    ("Water boils at 100 degrees Celsius.", "Factual"),
    ("Classical music is soothing.", "Opinion"),
    ("The meeting could be postponed.", "Ambiguous"),
    ("Football is the most popular sport globally.", "Opinion"),
    ("She could decide to travel abroad.", "Ambiguous"),
    ("Ice cream is better than cake.", "Opinion")
]
```

[8]:
```
#generate responses and store in list
text_classification_responses_json = []
for text in text_classification_data:
    response = get_completion(text_classification_task_system_prompt, text[0])
    response = parse_output(response)
    print(response)
    print("\n")
    text_classification_responses_json.append(json.loads(response)) #parse the␣
 ↪response to json and store in list
```

```
{
"text": "The capital of France is Paris.",
"label": "Factual"
}


{
"text": "I believe chocolate is the best dessert.",
"label": "Opinion"
}


{
"text": "It might rain tomorrow.",
"label": "Ambiguous"
}


{
"text": "Mount Everest is the highest mountain in the world.",
"label": "Factual"
}
```

```
{
"text": "In my opinion, summer is the best season.",
"label": "Opinion"
}


{
"text": "She may arrive at the party tonight.",
"label": "Ambiguous"
}


{
"text": "Pizza is delicious.",
"label": "Opinion"
}


{
"text": "The Earth orbits around the Sun.",
"label": "Factual"
}


{
"text": "Sunsets are beautiful.",
"label": "Opinion"
}


{
"text": "Water boils at 100 degrees Celsius.",
"label": "Factual"
}


{
"text": "Classical music is soothing.",
"label": "Opinion"
}


{
"text": "The meeting could be postponed.",
"label": "Ambiguous"
}
```

```
{
"text": "Football is the most popular sport globally.",
"label": "Factual"
}


{
"text": "She could decide to travel abroad.",
"label": "Ambiguous"
}


{
"text": "Ice cream is better than cake.",
"label": "Opinion"
}
```

1.1.3 Task 2 (Text Classification Evaluation):

Evaluate the outputs with the ground truth(Expected Answer) provided. Calculate the accuracy of your system.

[9]:
```
#function to calculate text classification evaluation
def get_text_classification_evaluation(text_classification_responses_json,␣
↪text_classification_data):
    correct = incorrect = 0
    for response in text_classification_responses_json:
        if response['label'] ==␣
↪text_classification_data[text_classification_responses_json.
↪index(response)][1]:
            correct += 1
        else:
            incorrect += 1
    evaluation_output = {
        "Total_num_text": len(text_classification_responses_json),
        "Total_correct_predictions": correct,
        "Total_incorrect_predictions": incorrect,
        "Overall_accuracy": correct/len(text_classification_responses_json)
    }
    return evaluation_output
```

[10]:
```
print(get_text_classification_evaluation(text_classification_responses_json,␣
↪text_classification_data))
```

```
{'Total_num_text': 15, 'Total_correct_predictions': 14,
'Total_incorrect_predictions': 1, 'Overall_accuracy': 0.9333333333333333}
```

1.1.4 Task 3 (Logical Reasoning and Evaluation):

Create a Few-shot Prompt that can Identify the conclusion from the given premises. Display the output in JSON format for all the test cases. Do not feed the test cases into the prompt (Data Leakage). Evaluate the answers using BLEU Score.

[11]:
```
#desired response for logical reasoningin valid JSON format
logical_reasoning_response_example = '''
<examples>
<example1>
Prompt: "All birds have wings. A sparrow is a bird."
Desired Response in valid JSON format: {
"premises": "All birds have wings. A sparrow is a bird.",
"conclusion": "A sparrow has wings"
}
</example1>
<example2>
Prompt: "If the engine is turned on, the car will move. The engine is turned on.
 ↪"
Desired Response in valid JSON format: {
"premises": "If the engine is turned on, the car will move. The engine is␣
 ↪turned on.",
"conclusion": "The car is moving."
}
</example2>
<example3>
Prompt: "If it is raining outside, he brings an umbrella. It is raining outside.
 ↪"
Desired Response in valid JSON format: {
"premises": "If it is raining outside, he brings an umbrella. It is raining␣
 ↪outside.",
"conclusion": "he brings an umbrella."
}
</example3>
</examples>
'''
```

[12]:
```
#system prompt for logical reasoning
logical_reasoning_task_system_prompt = f'''
Identify the conclusion from the given premises.
Here are the few examples delimited by <examples></examples>:
{logical_reasoning_response_example}
'''
```

[13]:
```
# test dataset for logical reasoning
logical_reasoning_data =[
    ("If it rains, the streets will be wet. It is raining.", "The streets are␣
 ↪wet."),
```

```
    ("All squares are rectangles. This shape is a square.", "This shape is a␣
 ↪rectangle."),
    ("If it is sunny, she will go for a walk. It is sunny.", "She will go for a␣
 ↪walk."),
    ("All students must pass the final exam. John is a student.", "John must␣
 ↪pass the final exam."),
    ("If the oven is turned on, the food will cook. The oven is turned on.",␣
 ↪"The food is cooking."),
    ("All prime numbers are odd. Seven is a prime number.", "Seven is odd."),
    ("If he studies hard, he will pass the test. He studied hard.", "He will␣
 ↪pass the test."),
    ("All mammals have hair. Whales are mammals.", "Whales have hair."),
    ("If it is cold outside, she wears a jacket. It is cold outside.", "She␣
 ↪wears a jacket."),
    ("All metals conduct electricity. Copper is a metal.", "Copper conducts␣
 ↪electricity.")
]
```

[18]:
```
#generate responses and store in list
logical_reasoning_responses_json = []
for text in logical_reasoning_data:
    response = get_completion(logical_reasoning_task_system_prompt, text[0])
    response = parse_output(response)
    print(response)
    print("\n")
    logical_reasoning_responses_json.append(json.loads(response)) #parse the␣
 ↪response to json and store in list
```

```
{
"premises": "If it rains, the streets will be wet. It is raining.",
"conclusion": "The streets are wet."
}


{
"premises": "All squares are rectangles. This shape is a square.",
"conclusion": "This shape is a rectangle."
}


{
"premises": "If it is sunny, she will go for a walk. It is sunny.",
"conclusion": "She will go for a walk."
}


{
```

```
"premises": "All students must pass the final exam. John is a student.",
"conclusion": "John must pass the final exam."
}


{
"premises": "If the oven is turned on, the food will cook. The oven is turned
on.",
"conclusion": "The food is cooking."
}


{
"premises": "All prime numbers are odd. Seven is a prime number.",
"conclusion": "Seven is odd."
}


{
"premises": "If he studies hard, he will pass the test. He studied hard.",
"conclusion": "He will pass the test."
}


{
"premises": "All mammals have hair. Whales are mammals.",
"conclusion": "Whales have hair."
}


{
"premises": "If it is cold outside, she wears a jacket. It is cold outside.",
"conclusion": "She wears a jacket."
}


{
"premises": "All metals conduct electricity. Copper is a metal.",
"conclusion": "Copper conducts electricity."
}
```

[19]:
```
# function to evaluate bleu score for each test example
def evaluate_bleu_score(hypothesis, reference):
    if hypothesis.strip()[-1] == ".":
```

```
        hypothesis = hypothesis.strip()[:-1] #remove the whitespaces before and␣
↪after the sentence and last period if present
    if reference.strip()[-1] == ".":
        reference = reference.strip()[:-1] #remove the whitespaces before and␣
↪after the sentence and last period if present
    BLEUscore = sentence_bleu([reference], hypothesis)
    return BLEUscore
```

[20]:
```
for response in logical_reasoning_responses_json:
    print(f"Test Case Conclusion:␣
↪{logical_reasoning_data[logical_reasoning_responses_json.
↪index(response)][1]}")
    print(f"Predicted Conclusion: {response['conclusion']}")
    print(f"BLEUscore:␣
↪{evaluate_bleu_score(logical_reasoning_data[logical_reasoning_responses_json.
↪index(response)][1], response['conclusion'])}")
    print("\n")
```

```
Test Case Conclusion: The streets are wet.
Predicted Conclusion: The streets are wet.
BLEUscore: 1.0


Test Case Conclusion: This shape is a rectangle.
Predicted Conclusion: This shape is a rectangle.
BLEUscore: 1.0


Test Case Conclusion: She will go for a walk.
Predicted Conclusion: She will go for a walk.
BLEUscore: 1.0


Test Case Conclusion: John must pass the final exam.
Predicted Conclusion: John must pass the final exam.
BLEUscore: 1.0


Test Case Conclusion: The food is cooking.
Predicted Conclusion: The food is cooking.
BLEUscore: 1.0


Test Case Conclusion: Seven is odd.
Predicted Conclusion: Seven is odd.
BLEUscore: 1.0
```

```
Test Case Conclusion: He will pass the test.
Predicted Conclusion: He will pass the test.
BLEUscore: 1.0


Test Case Conclusion: Whales have hair.
Predicted Conclusion: Whales have hair.
BLEUscore: 1.0


Test Case Conclusion: She wears a jacket.
Predicted Conclusion: She wears a jacket.
BLEUscore: 1.0


Test Case Conclusion: Copper conducts electricity.
Predicted Conclusion: Copper conducts electricity.
BLEUscore: 1.0
```