exercise2

September 3, 2024

1 Importing Libraries and set up the api key

[21]:
```
!pip install openai
from openai import OpenAI
from google.colab import userdata
import json
from nltk.translate import bleu_score
```

```
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages
(1.43.0)
Requirement already satisfied: anyio<5,>=3.5.0 in
/usr/local/lib/python3.10/dist-packages (from openai) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-
packages (from openai) (1.7.0)
Requirement already satisfied: httpx<1,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from openai) (0.27.2)
Requirement already satisfied: jiter<1,>=0.4.0 in
/usr/local/lib/python3.10/dist-packages (from openai) (0.5.0)
Requirement already satisfied: pydantic<3,>=1.9.0 in
/usr/local/lib/python3.10/dist-packages (from openai) (2.8.2)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-
packages (from openai) (1.3.1)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages
(from openai) (4.66.5)
Requirement already satisfied: typing-extensions<5,>=4.11 in
/usr/local/lib/python3.10/dist-packages (from openai) (4.12.2)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-
packages (from anyio<5,>=3.5.0->openai) (3.8)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio<5,>=3.5.0->openai) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from httpx<1,>=0.23.0->openai) (2024.7.4)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-
packages (from httpx<1,>=0.23.0->openai) (1.0.5)
Requirement already satisfied: h11<0.15,>=0.13 in
/usr/local/lib/python3.10/dist-packages (from
httpcore==1.*->httpx<1,>=0.23.0->openai) (0.14.0)
Requirement already satisfied: annotated-types>=0.4.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai)
(0.7.0)
Requirement already satisfied: pydantic-core==2.20.1 in
/usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0->openai)
(2.20.1)
```

[22]:
```
client = OpenAI(
    api_key=userdata.get('API_KEY'),
)
```

2 Define the function to get responses from openai

[23]:
```
def get_completion(system_prompt, prompt, model="gpt-3.5-turbo"):
    messages = [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": prompt}]
    response = client.chat.completions.create(
        model=model,
        messages=messages,
        temperature=0,
    )
    return response.choices[0].message.content
```

3 Test Datasets for all tasks

[24]:
```
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")
]
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.")
]
```

4 System Prompts for Tasks

[25]:
```
text_classification_sys_prompt='''
Classify 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.
Examples:
Input: The Earth is the third planet from the Sun.
Expected Output:
{
"text": "The Earth is the third planet from the Sun.",
"label": "Factual"
}

Input: Swmming is fun.
Expected Output:
{
"text": "Swmming is fun.",
"label": "Opinion"
}

Input: It might rain tomorrow.
Expected Output:
{
"text": "It might rain tomorrow.",
"label": "Ambiguous
```

```
}
'''
```

[26]:
```
logical_reasoning_sys_prompt='''
Identify the conclusion from the given premises. Display the
output in JSON format for all the test cases.
Examples:
Input:All birds have wings. A sparrow is a bird.
Expected Output:
{
"premises": "All birds have wings. A sparrow is a bird.",
"conclusion": "A sparrow has wings"
}

Input:If it snows, the ground will be covered in snow. It is
snowing.
Expected Output:
{
"premises": "If it snows, the ground will be covered in snow. It is
snowing.",
"conclusion": "The ground is covered in snow"
}

Input:All dogs are mammals. Max is a dog.
Expected Output:
{
"premises": "All dogs are mammals. Max is a dog.",
"conclusion": "Max is a mammal"
}
'''
```

5 Task 1(Text Classification):

[27]:
```
text_classification_responses = []
for text in text_classification_data:
  response = get_completion(text_classification_sys_prompt, text[0])
  print(response)
 ␣
  ↪print('-----------------------------------------------------------------------------')
  text_classification_responses.append(json.loads(response))
```

```
{
"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"
}
--------------------------------------------------------------------------------
```

6 Task 2(Text Classification Evaluation):

[28]:
```
def get_text_classification_evaluation():
  correct = incorrect = 0
  for i in range(len(text_classification_responses)):
    if text_classification_responses[i]['label'] ==␣
 ↪text_classification_data[i][1]:
      correct += 1
    else:
      incorrect += 1

  total_num_text = correct + incorrect
  accuracy = correct / total_num_text
  results = {
      "Total_num_text": total_num_text,
      "Total_correct_predictions": correct,
      "Total_incorrect_predictions": incorrect,
      "overall_accuracy": accuracy
  }
  return results
```

[29]:
```
print(get_text_classification_evaluation())
```

```
{'Total_num_text': 15, 'Total_correct_predictions': 14,
```

'Total_incorrect_predictions': 1, 'overall_accuracy': 0.9333333333333333}

7 Task 3:

7.1 Logical Reasoning

[30]:
```
logical_reasoning_conclusions=[]
for text in logical_reasoning_data:
  response = get_completion(logical_reasoning_sys_prompt, text[0])
  print(response)
 ␣
 ↪print('-----------------------------------------------------------------------')
  logical_reasoning_conclusions.append(json.loads(response)['conclusion'])
```

```
{
"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"
}
--------------------------------------------------------------------------
```

7.2 Reasoning Evaluation

[31]:
```python
def display_bleu(reference, hypothesis):
  # remove fullstops if there are any
  if reference[-1] =='.':
    reference = reference[:-1]
  if hypothesis[-1] =='.':
    hypothesis = hypothesis[:-1]

  print(f'Reference: {reference}')
  print(f'Hypothesis: {hypothesis}')
  bleu = bleu_score.sentence_bleu([reference],hypothesis)
  print(f'BLEU Score: {bleu}')
  print('----------------------------------------')
```

[32]:
```python
for i in range(len(logical_reasoning_data)):
  display_bleu(logical_reasoning_data[i][1],logical_reasoning_conclusions[i])
```

```
Reference: The streets are wet
Hypothesis: The streets are wet
BLEU Score: 1.0
----------------------------------------
Reference: This shape is a rectangle
Hypothesis: This shape is a rectangle
BLEU Score: 1.0
----------------------------------------
Reference: She will go for a walk
Hypothesis: She will go for a walk
BLEU Score: 1.0
```

```
----------------------------------------
Reference: John must pass the final exam
Hypothesis: John must pass the final exam
BLEU Score: 1.0
----------------------------------------
Reference: The food is cooking
Hypothesis: The food is cooking
BLEU Score: 1.0
----------------------------------------
Reference: Seven is odd
Hypothesis: Seven is odd
BLEU Score: 1.0
----------------------------------------
Reference: He will pass the test
Hypothesis: He will pass the test
BLEU Score: 1.0
----------------------------------------
Reference: Whales have hair
Hypothesis: Whales have hair
BLEU Score: 1.0
----------------------------------------
Reference: She wears a jacket
Hypothesis: She wears a jacket
BLEU Score: 1.0
----------------------------------------
Reference: Copper conducts electricity
Hypothesis: Copper conducts electricity
BLEU Score: 1.0
----------------------------------------
```