COURSE: AI Assisted Coding

NAME:B.Thanmai

BATCH-33

HALLTICKET.NO : 2303A52126

Lab 4: Advanced Prompt Engineering: Zero-shot, one-shot, and few shot techniques

Objective: To explore and compare Zero-shot, One-shot, and Few-shot prompting techniques for classifying emails into predefined categories using a large language model (LLM).

1. Suppose that you work for a company that receives hundreds of customer emails daily. Management wants to automatically classify emails into categories like "Billing", "Technical Support", "Feedback", and "Others" before assigning them to appropriate departments. Instead of training a new model, your task is to use prompt engineering techniques with an existing LLM to handle the classification.

Tasks to be completed are as below

a. Prepare Sample Data:

• Create or collect 10 short email samples, each belonging to one of the 4 categories.

b. Zero-shot Prompting:

• Design a prompt that asks the LLM to classify a single email without providing any examples.

• Example prompt:

“Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Email: ‘I have not received my invoice for last month.’”

c. One-shot Prompting:

• Add one labeled example before asking the model to classify a new email.

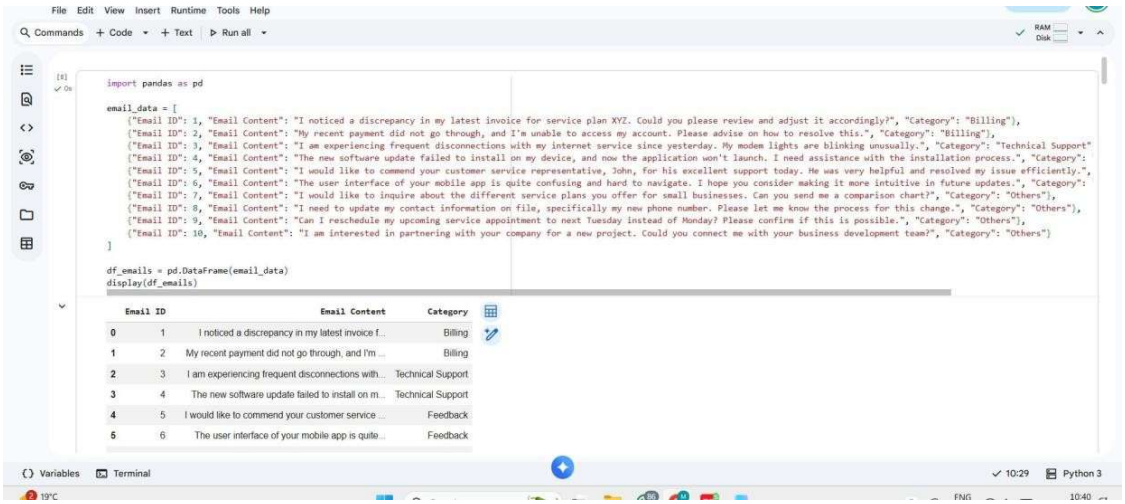
d. Few-shot Prompting:

• Use 3–5 labeled examples in your prompt before asking the model to classify a new email.

e. Evaluation:

• Run all three techniques on the same set of 5 test emails.

• Compare and document the accuracy and clarity of responses.



2. Travel Query Classification

Scenario:

A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info.

Tasks:

a. Prepare labeled travel queries.

b. Apply Zero-shot prompting.

c. Apply One-shot prompting.

d. Apply Few-shot prompting.

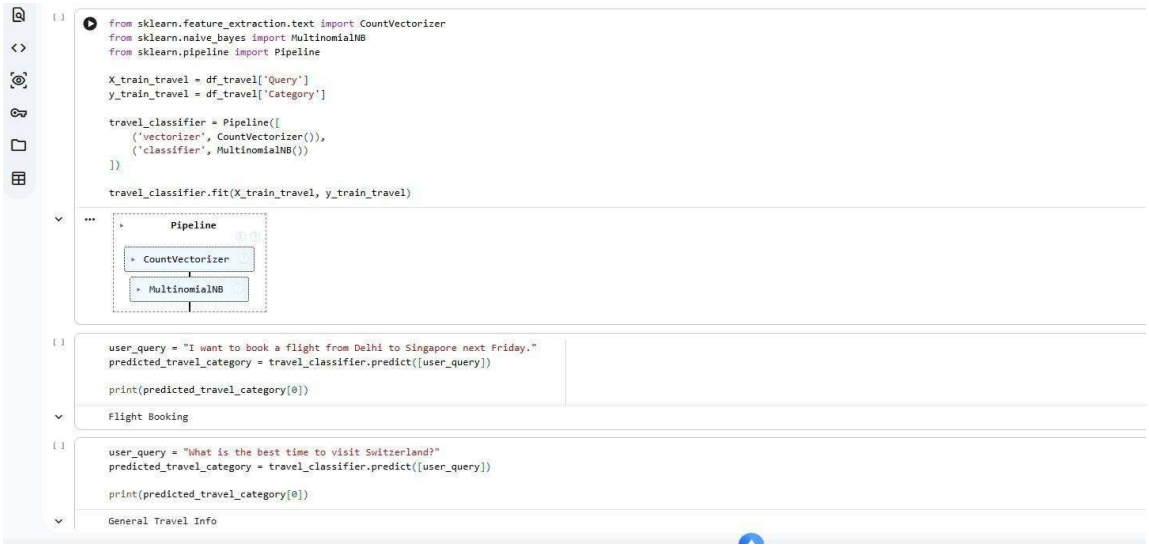
e. Compare response consistency.

```
new_travel_queries = [
    "I want to book a flight from Delhi to Singapore next Friday.",
    "Can you suggest budget hotels in Paris?",
    "I need to cancel my flight scheduled for tomorrow.",
    "What is the best time to visit Switzerland?",
    "Book a hotel near Times Square for 3 nights."
]

predicted_travel_categories = text_classifier.predict(new_travel_queries)

travel_results = []
for query, category in zip(new_travel_queries, predicted_travel_categories):
    travel_results.append({"Travel Query": query, "Predicted Category": category})

df_travel_results = pd.DataFrame(travel_results)
display(df_travel_results)
```

	Travel Query	Predicted Category
0	I want to book a flight from Delhi to Singapor...	Others
1	Can you suggest budget hotels in Paris?	Others
2	I need to cancel my flight scheduled for tomor...	Others
3	What is the best time to visit Switzerland?	Others
4	Book a hotel near Times Square for 3 nights.	Others

```
travel_data = [
    {"Query": "I want to book a flight from Delhi to Singapore next Friday.", "Category": "Flight Booking"},
    {"Query": "Can you find me a hotel in Paris for three nights next month?", "Category": "Hotel Booking"},
    {"Query": "I need to cancel my flight scheduled for tomorrow.", "Category": "Cancellation"},
    {"Query": "What are the visa requirements for traveling to Japan?", "Category": "General Travel Info"},
    {"Query": "Book a hotel near Times Square for 3 nights.", "Category": "Hotel Booking"},
    {"Query": "I would like to change my return flight date.", "Category": "Flight Booking"},
    {"Query": "How can I get a refund for my canceled trip?", "Category": "Cancellation"},
    {"Query": "What are the best attractions in Rome?", "Category": "General Travel Info"},
    {"Query": "Find cheapest flights to London for next summer.", "Category": "Flight Booking"},
    {"Query": "Is there a direct train from Florence to Venice?", "Category": "General Travel Info"}
]

df_travel = pd.DataFrame(travel_data)
display(df_travel)
```

	Query	Category
0	I want to book a flight from Delhi to Singapor...	Flight Booking
1	Can you find me a hotel in Paris for three nig...	Hotel Booking
2	I need to cancel my flight scheduled for tomor...	Cancellation
3	What are the visa requirements for traveling t...	General Travel Info
4	Book a hotel near Times Square for 3 nights.	Hotel Booking
5	I would like to change my return flight date.	Flight Booking
6	How can I get a refund for my canceled trip?	Cancellation
7	What are the best attractions in Rome?	General Travel Info
8	Find cheapest flights to London for next summer.	Flight Booking
9	Is there a direct train from Florence to Venice?	General Travel Info



3. Programming Question Type Identification

Scenario:

A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question.

Tasks:

a. Prepare coding-related user queries.

b. Perform Zero-shot classification.

c. Perform One-shot classification.

d. Perform Few-shot classification.

e. Analyze improvements in technical accuracy.

3

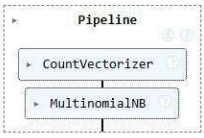
```
programming_queries = [
    {"Query": "Why am I getting a missing semicolon error in C?", "Category": "Syntax Error"},
    {"Query": "My program runs but gives wrong output.", "Category": "Logic Error"},
    {"Query": "How can I reduce time complexity of this loop?", "Category": "Optimization"},
    {"Query": "What is polymorphism in Java?", "Category": "Conceptual Question"},
    {"Query": "Code is too slow for large inputs.", "Category": "Optimization"}
]

df_programming = pd.DataFrame(programming_queries)
display(df_programming)
```

	Query	Category
0	Why am I getting a missing semicolon error in C?	Syntax Error
1	My program runs but gives wrong output.	Logic Error
2	How can I reduce time complexity of this loop?	Optimization
3	What is polymorphism in Java?	Conceptual Question
4	Code is too slow for large inputs.	Optimization

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
X_train_programming = df_programming['Query']
y_train_programming = df_programming['Category']

programming_classifier = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('classifier', MultinomialNB())
])

programming_classifier.fit(X_train_programming, y_train_programming)
```




```
new_programming_query = "I'm getting an 'index out of bounds' error. What does it mean?"
predicted_programming_category = programming_classifier.predict([new_programming_query])

print(f"Query: '{new_programming_query}'\nPredicted Category: {predicted_programming_category[0]}")
```

```
Query: 'I'm getting an 'index out of bounds' error. What does it mean?'
Predicted Category: Syntax Error
```

```
another_programming_query = "How to implement quicksort in Python?"
predicted_programming_category_2 = programming_classifier.predict([another_programming_query])

print(f"Query: '{another_programming_query}'\nPredicted Category: {predicted_programming_category_2[0]}")
```

```
Query: 'How to implement quicksort in Python?'
Predicted Category: Optimization
```

```
#b
```

```
new_query_to_classify = "My program runs but gives incorrect results."
predicted_category_new_query = programming_classifier.predict([new_query_to_classify])

print(predicted_category_new_query[0])
```

```
Logic Error
```

```
c
```

```
new_programming_query_user = "Why is my code throwing an unexpected token error?"
predicted_programming_category_user = programming_classifier.predict([new_programming_query_user])

print(predicted_programming_category_user[0])
```

```
Syntax Error
```

```
new_programming_query_user_2 = "Code is very slow when input size increases."
predicted_programming_category_user_2 = programming_classifier.predict([new_programming_query_user_2])

print(predicted_programming_category_user_2[0])
```

```
Optimization
```

4. Social Media Post Categorization

Scenario:

A social media analytics tool must classify posts into Promotion,

Complaint, Appreciation, or Inquiry.

Tasks:

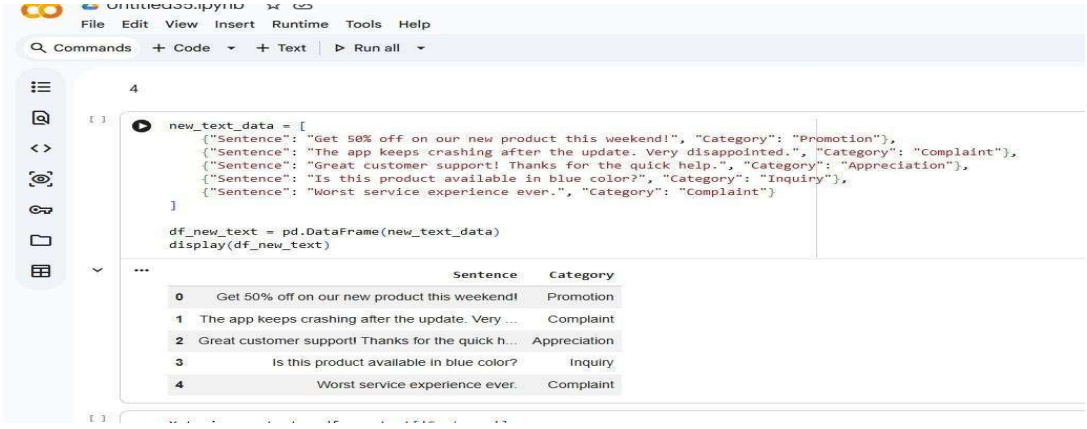
1. Prepare sample social media posts.

2. Use Zero-shot prompting.

3. Use One-shot prompting.

4. Use Few-shot prompting.

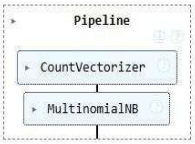
5. Analyze informal language handling



File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
X_train_new_text = df_new_text['Sentence']
y_train_new_text = df_new_text['Category']

new_text_classifier = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('classifier', MultinomialNB())
])

new_text_classifier.fit(X_train_new_text, y_train_new_text)
```




```
test_sentence_1 = "I love your products!"
predicted_category_1 = new_text_classifier.predict([test_sentence_1])
print(f"Sentence: '{test_sentence_1}'\nPredicted Category: {predicted_category_1[0]}")

test_sentence_2 = "This is the best deal ever!"
predicted_category_2 = new_text_classifier.predict([test_sentence_2])
print(f"Sentence: '{test_sentence_2}'\nPredicted Category: {predicted_category_2[0]}")
```

```
Sentence: 'I love your products!'
Predicted Category: Complaint
Sentence: 'This is the best deal ever!'
Predicted Category: Complaint
```

Commands + Code + Text Run all

```
social_media_post = "Great customer support! Thanks for the quick help."
predicted_category_post = new_text_classifier.predict([social_media_post])

print(predicted_category_post[0])
```

```
Appreciation
```

```
new_social_media_post = "Get 30% discount on all items today only!"
predicted_category_new_post = new_text_classifier.predict([new_social_media_post])

print(predicted_category_new_post[0])
```

```
Promotion
```

```
social_media_post_to_classify = "The delivery was delayed and support was unresponsive."
predicted_category_for_post = new_text_classifier.predict([social_media_post_to_classify])

print(predicted_category_for_post[0])
```

```
Complaint
```

Task

```
print("Grammar check results:\n")

# Iterate through each row of the DataFrame and display results
for index, row in df_grammar_results.iterrows():
    email_id = row['Email ID']
    email_content = row['Email Content']
    matches = row['Grammar Matches']

    print(f"Email ID: {email_id}")
    print(f"Email Content: {email_content}")

    if matches:
        print("  Detected Grammar Errors:")
        for match in matches:
            print(f"    - Message: {match.message}")
            print(f"      Suggestions: {', '.join(match.replacements)}")
            print(f"      Context: '{match.context}' (Error at offset {match.offset} with length {match.errorlength})")
    else:
        print("  No grammar errors detected.")
    print("\n" + "-" * 80 + "\n") # Separator for readability
```

```
Grammar check results:

Email ID: 1
Email Content: I noticed a discrepancy in my latest invoice for service plan XYZ. Could you please review and adjust it accordingly?
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 2
Email Content: My recent payment did not go through, and I'm unable to access my account. Please advise on how to resolve this.
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 3
```

Gmail YouTube Maps

Untitled35.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
Email ID: 5
Email Content: I would like to commend your customer service representative, John, for his excellent support today. He was very helpful and resolved my issue efficiently.
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 6
Email Content: The user interface of your mobile app is quite confusing and hard to navigate. I hope you consider making it more intuitive in future updates.
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 7
Email Content: I would like to inquire about the different service plans you offer for small businesses. Can you send me a comparison chart?
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 8
Email Content: I need to update my contact information on file, specifically my new phone number. Please let me know the process for this change.
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 9
Email Content: Can I reschedule my upcoming service appointment to next Tuesday instead of Monday? Please confirm if this is possible.
  No grammar errors detected.

--------------------------------------------------------------------------------

Email ID: 10
Email Content: I am interested in partnering with your company for a new project. Could you connect me with your business development team?
  No grammar errors detected.
```

```
[19]
✓ 0s
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

# Prepare the data
X_train = df_emails['Email Content']
y_train = df_emails['Category']

# Create a pipeline with CountVectorizer and Multinomial Naive Bayes
text_classifier = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('classifier', MultinomialNB())
])

# Train the classifier
text_classifier.fit(X_train, y_train)

# Email to classify
email_to_classify = 'I noticed a discrepancy in my latest invoice for service plan XYZ. Could you please review and a

# Predict the category
predicted_category = text_classifier.predict([email_to_classify])

print(predicted_category[0])

Billing
```

```
print(predicted_category[0])

Billing

[20]
✓ 0s
email_to_classify_new = "The app crashes every time I try to open it after the update."
predicted_category_new = text_classifier.predict([email_to_classify_new])
print(predicted_category_new[0])

Technical Support

[21]
✓ 0s
email_to_classify_user = "Website is very slow and pages take too long to load."
predicted_category_user = text_classifier.predict([email_to_classify_user])
print(predicted_category_user[0])

Feedback
```

```
[22]
✓ 0s
new_emails = [
    "Invoice not received",
    "App crashes",
    "Loved the design",
    "Website slow",
    "Office timings"
]

predictions = text_classifier.predict(new_emails)

results_data = []
for email, category in zip(new_emails, predictions):
    results_data.append({"Email": email, "Predicted Category": category})

df_results = pd.DataFrame(results_data)
display(df_results)
```

	Email	Predicted Category
0	Invoice not received	Billing
1	App crashes	Feedback
2	Loved the design	Others
3	Website slow	Others
4	Office timings	Others