

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

**COURSE: AI Assisted Coding**

**NAME: G. SHRAVANI**

**BATCH-34**

**HALLTICKET.NO : 2303A52361**

**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.

The screenshot shows a Jupyter Notebook interface in Google Colab. The code cell contains Python code to import pandas and read a CSV file into a DataFrame. The resulting DataFrame is displayed below the code cell, showing 10 rows of data with columns: Email ID, Email Content, and Category. The data is as follows:

Email ID	Email Content	Category
0	I noticed a discrepancy in my latest invoice ...	Billing
1	My recent payment did not go through, and I'm ...	Billing
2	I am experiencing frequent disconnections with ...	Technical Support
3	The new software update failed to install on m...	Technical Support
4	I would like to commend your customer service ...	Feedback
5	The user interface of your mobile app is quite...	Feedback
6	I would like to inquire about the different se...	Others
7	I need to update my contact information on fl...	Others
8	I am reschedule my upcoming service appointment...	Others
9	I am interested in partnering with your compan...	Others
10	I am interested in partnering with your compan...	Others

## 2. Travel Query Classification

### Scenario:

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

### Tasks:

- Prepare labeled travel queries.
- Apply Zero-shot prompting.
- Apply One-shot prompting.
- Apply Few-shot prompting.

## e. Compare response consistency.

A screenshot of a Jupyter Notebook titled "Untitled68.ipynb". The code cell contains Python code to process travel queries and predict their categories. The output shows a table of travel queries and their predicted categories:

Travel Query	Predicted Category
I want to book a flight from Delhi to Singapore next Friday.	Flight Booking
Can you suggest budget hotels in Paris?	Hotel Booking
I need to cancel my flight scheduled for tomorrow.	Cancellation
What is the best time to visit Switzerland?	General Travel Info
Book a hotel near Times Square for 3 nights.	Hotel Booking

The notebook interface includes tabs for "Variables" and "Terminal" at the bottom, and a status bar showing "6:58 PM" and "Python 3".

A screenshot of a Jupyter Notebook titled "Untitled68.ipynb". The code cell contains Python code to process travel queries and predict their categories. The output shows a table of travel queries and their predicted categories, including some new ones:

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

The notebook interface includes tabs for "Variables" and "Terminal" at the bottom, and a status bar showing "6:58 PM" and "Python 3".

A screenshot of a Jupyter Notebook titled "Untitled68.ipynb". The code cell contains Python code to create a machine learning pipeline for travel query classification. The pipeline consists of a CountVectorizer followed by a MultinomialNB classifier. The notebook interface includes tabs for "Commands", "Code", "Text", and "Run all", and a status bar showing "RAM Disk".

```
[4] 3s
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

X_train_travel = df_travel['Query']
y_train_travel = df_travel['Category']

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

travel_classifier.fit(X_train_travel, y_train_travel)
```

The screenshot shows a Google Colab notebook titled "Untitled68.ipynb". It contains two code cells. The first cell has a status of [5] 0s and contains the following Python code:

```
user_query = ["I want to book a flight from Delhi to Singapore next Friday."]
predicted_travel_category = travel_classifier.predict(user_query)

print(predicted_travel_category[0])
```

The output of this cell is "Flight Booking". The second cell has a status of [6] 0s and contains the same code with a different user query:

```
user_query = ["What is the best time to visit Switzerland?"]
predicted_travel_category = travel_classifier.predict(user_query)

print(predicted_travel_category[0])
```

The output of this cell is "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:

- Prepare coding-related user queries.
- Perform Zero-shot classification.
- Perform One-shot classification.
- Perform Few-shot classification.
- Analyze improvements in technical accuracy

The screenshot shows a Google Colab notebook titled "Untitled68.ipynb". It displays a DataFrame named "df\_programming" with the following data:

	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

A screenshot of a Google Colab notebook titled "Untitled68.ipynb". The notebook interface includes a toolbar with File, Edit, View, Insert, Runtime, Tools, Help, Commands, Code, Text, Run all, Share, and RAM/Disk status. The code cell [11] contains Python code to train a classifier on a dataset of programming queries:

```
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)
```

The code cell [12] shows the pipeline structure:

```
Pipeline
  -> CountVectorizer
  -> MultinomialNB
```

The code cell [13] contains a query and its predicted category:

```
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]}")
```

Output:

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

The code cell [14] contains another query and its predicted category:

```
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]}")
```

Output:

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

A screenshot of a Google Colab notebook titled "Untitled68.ipynb". The notebook interface includes a toolbar with File, Edit, View, Insert, Runtime, Tools, Help, Commands, Code, Text, Run all, Share, and RAM/Disk status. The code cell [11] contains Python code to predict categories for new queries:

```
# 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])
```

The output is labeled "Logic Error".

The code cell [12] contains another query and its predicted category:

```
# 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])
```

The output is labeled "Syntax Error".

The code cell [13] contains a third query and its predicted category:

```
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])
```

The output is labeled "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

Untitled68.ipynb

```
[16] 0s
new_text_data = [
    {"Sentence": "Get 50% off on our new product this weekend!", "Category": "Promotion"},
    {"Sentence": "The app keeps crashing after the update. Very disappointed.", "Category": "Complaint"},
    {"Sentence": "Great customer support! Thanks for the quick help.", "Category": "Appreciation"},
    {"Sentence": "Is this product available in blue color?", "Category": "Inquiry"},
    {"Sentence": "Worst service experience ever.", "Category": "Complaint"}]

df_new_text = pd.DataFrame(new_text_data)
display(df_new_text)
```

	Sentence	Category
0	Get 50% off on our new product this weekend!	Promotion
1	The app keeps crashing after the update. Very ...	Complaint
2	Great customer support! Thanks for the quick h...	Appreciation
3	Is this product available in blue color?	Inquiry
4	Worst service experience ever.	Complaint

Next steps: [Generate code with df\\_new\\_text](#) [New interactive sheet](#)

Untitled68.ipynb

```
[17] 0s
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)
```

Pipeline

- > CountVectorizer
- > MultinomialNB

```
[18] 0s
test_sentence_1 = "I love your products!"
predicted_category_1 = new_text_classifier.predict([test_sentence_1])
print(f"Sentences: {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"Sentences: {test_sentence_2}\nPredicted Category: {predicted_category_2[0]}")
```

Sentences: I love your products!
Predicted Category: Complaint  
Sentences: This is the best deal ever!
Predicted Category: Complaint

Untitled68.ipynb

```
[19] 0s
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

```
[20] 0s
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

```
[21] 0s
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

Untitled68.ipynb

```
[M] 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}'"
                  f" (Error at offset {match.offset}))")
    else:
        print("  No grammar errors detected.")

    # Separator for readability
    print("\n" + "-" * 80 + "\n")

--- 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.

Untitled68.ipynb

```
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▶ Run all ▶
```

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  
Email Content: I am experiencing frequent disconnections with the internet service since yesterday. My modem lights are blinking unusually.  
No grammar errors detected.

-----

Email ID: 4  
Email Content: The new software update failed to install on my device, and now the application won't launch. I need assistance with installation process.  
No grammar errors detected.

-----

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 that change.  
No grammar errors detected.

-----

Email ID: 9  
Email Content: I am reschedule my upcoming service appointment to next Tuesday instead of Monday? Please confirm if this is possible.  
Detected Grammar Errors:  
- Message: The base form of a verb does not usually follow 'am'. Double-check that the verb is in the correct form.  
Suggestions: reschedule, rescheduled, am rescheduling  
Context: 'I am reschedule my upcoming service appointment to next...' (Error at offset 2)

-----

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.

-----

Untitled68.ipynb

```
[27]: 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 adjust it accordingly?"
)

# Predict the category
predicted_category = text_classifier.predict([email_to_classify])
print(predicted_category[0])

... Billing
```

Untitled68.ipynb

```
[27]: )
# Predict the category
predicted_category = text_classifier.predict([email_to_classify])
print(predicted_category[0])
... Billing

[28]: # Classify a new email (technical issue)
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

[29]: # Classify another email (performance feedback)
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

[30]: new_emails = [
    "Invoice not received",
    "App crashes",
```

Untitled68.ipynb

```
[31]: 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
```