

AI Assisted Coding

Assignment 4.5

Name: C. Vaishnav Reddy

Hallticket:2303A52203

Task-01

Prompt:

Generate a simple Python program that demonstrates zero-shot, one-shot, and few-shot prompting for classifying customer emails into Billing, Technical Support, Feedback, and Others. The program should create sample emails, show example prompts for each technique, classify the same 5 test emails using mocked responses, and print the results with clear comments explaining each step.

CODE:

```
# Sample emails for classification
sample_emails = [
    "I have a question about my recent bill.",
    "The app keeps crashing on my phone.",
    "I love the new features you added!",
    "Can you help me reset my password?",
    "I want to cancel my subscription."
]

# Define prompts for zero-shot, one-shot, and few-shot classification
zero_shot_prompt = "Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.\nEmail: '{}'"
one_shot_prompt = "Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.\nExample: 'I have a question about my recent bill.' -> Billing\nEmail: '{}'"
few_shot_prompt = (
    "Classify the following emails into one of the categories: Billing, Technical Support, Feedback, Others.\n"
    "Examples:\n"
    "'I have a question about my recent bill.' -> Billing\n"
    "'The app keeps crashing on my phone.' -> Technical Support\n"
    "'I love the new features you added!' -> Feedback\n"
    "Email: '{}'"
)

# Mock LLM responses for classification
def mock_llm_response(prompt):
    if "billing" in prompt.lower():
        return "Billing"
    elif "crashing" in prompt.lower() or "reset" in prompt.lower():
        return "Technical Support"
    elif "love" in prompt.lower() or "features" in prompt.lower():
        return "Feedback"
    else:
        return "Others"

# Classify emails using different prompting techniques
results = {
    "Zero-Shot": [],
    "One-Shot": [],
    "Few-Shot": []
}
```

```

        else:
            return "Others"

# Classify emails using different prompting techniques
results = {
    "Zero-Shot": [],
    "One-Shot": [],
    "Few-Shot": []
}

for email in sample_emails:
    # Zero-shot classification
    response = mock_llm_response(zero_shot_prompt.format(email))
    results["Zero-Shot"].append(response)

    # One-shot classification
    response = mock_llm_response(one_shot_prompt.format(email))
    results["One-Shot"].append(response)

    # Few-shot classification
    response = mock_llm_response(few_shot_prompt.format(email))
    results["Few-Shot"].append(response)

# Print results and accuracy comparison
print("Classification Results:")
for technique, classifications in results.items():
    print(f"{technique}: {classifications}")

# Accuracy calculation (mocked for demonstration)
true_labels = ["Billing", "Technical Support", "Feedback", "Technical Support", "Billing"]
# Correcting the true labels for accuracy calculation
true_labels = ["Billing", "Technical Support", "Feedback", "Technical Support", "Billing"]
for technique, classifications in results.items():
    accuracy = sum(1 for i in range(len(classifications)) if classifications[i] == true_labels[i]) / len(true_labels) * 100
    print(f"Accuracy for {technique}: {accuracy:.2f}%")

```

Output:

```

PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding\email_classifier.py"
Email 5: Billing
Email 6: Billing
Email 7: Billing
Email 8: Technical Support
Email 9: Others
Email 10: Feedback
Email 11: Technical Support
Email 12: Others
Email 13: Billing
Email 14: Technical Support
Email 15: Feedback

Accuracy Comparison:

Zero-Shot Accuracy: 93.33%
One-Shot Accuracy: 93.33%
Few-Shot Accuracy: 93.33%
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>

```

Task 2 :

Prompt:

Generate a simple demonstration of zero-shot, one-shot, and few-shot prompting for classifying travel-related user queries into Flight Booking, Hotel Booking, Cancellation, and General Travel Info. Create sample travel queries, write prompts for each prompting technique, classify the same set of test queries, and display the results clearly. Use simple language and add comments to explain each step.

Code:

```
#####
# Travel Query Classification using Zero-Shot, One-Shot, and Few-Shot Prompting
#####

# Sample travel queries
travel_queries = [
    "I want to book a flight from New York to London next week.",
    "What's the best time to visit Paris?",
    "Can I cancel my hotel reservation?",
    "I need a 5-star hotel in Tokyo for 3 nights.",
    "How do I get a refund for my flight?",
    "What are the visa requirements for India?",
    "Book me a direct flight to Sydney.",
    "I want to cancel my booking immediately.",
    "Where can I find budget hotels in Barcelona?",
    "Tell me about local attractions in Rome."
]

# True labels for evaluation
true_labels_travel = [
    "Flight Booking",
    "General Travel Info",
    "Cancellation",
    "Hotel Booking",
    "Cancellation",
    "General Travel Info",
    "Flight Booking",
    "Cancellation",
    "Hotel Booking",
    "General Travel Info"
]

# Zero-Shot Prompt: No examples provided
zero_shot_travel = (
    "Classify the following travel query into one of these categories: "
    "Flight Booking, Hotel Booking, Cancellation, General Travel Info.\n"
    "Query: '{}'\n"
```

```

# One-Shot Prompt: One example provided
one_shot_travel = (
    "Classify the following travel query into one of these categories: "
    "Flight Booking, Hotel Booking, Cancellation, General Travel Info.\n"
    "Example: 'Book me a flight to Paris' → Flight Booking\n"
    "Query: '{}'\n"
    "Category:"
)

# Few-Shot Prompt: Multiple examples provided
few_shot_travel = (
    "Classify the following travel queries into one of these categories: "
    "Flight Booking, Hotel Booking, Cancellation, General Travel Info.\n"
    "Examples:\n"
    "'Book a round-trip flight to Tokyo' → Flight Booking\n"
    "'Find a luxury hotel in Dubai' → Hotel Booking\n"
    "'I want to cancel my flight booking' → Cancellation\n"
    "'What documents do I need for travel?' → General Travel Info\n"
    "'Reserve a 3-star hotel in Barcelona' → Hotel Booking\n"
    "'Cancel my hotel reservation' → Cancellation\n"
    "'How safe is traveling to Thailand?' → General Travel Info\n"
    "Query: '{}'\n"
    "Category:"
)

# Rule-based mock LLM for travel classification
def classify_travel_query(query_text):
    text = query_text.lower()

    # Flight Booking keywords
    if any(word in text for word in ["flight", "book", "airline", "departure", "arrival", "direct"]):
        if "cancel" not in text and "refund" not in text:
            return "Flight Booking"

    # Hotel Booking keywords
    if any(word in text for word in ["hotel", "accommodation", "lodge", "resort", "room", "booking"]):
        if "cancel" not in text:
            return "Hotel Booking"

    # Cancellation keywords
    if any(word in text for word in ["cancel", "refund", "cancelation", "cancellation"]):
        return "Cancellation"

    # General Travel Info keywords
    if any(word in text for word in ["documents", "travel info", "travel information"]):
        return "General Travel Info"

    # Direct Booking keywords
    if any(word in text for word in ["direct", "self-service", "online booking"]):
        return "Direct Booking"

    # Default category
    return "Unknown"
}

# Display results clearly
print("\n" + "*80)
print("TRAVEL QUERY CLASSIFICATION RESULTS")
print("*80)

for technique, predictions in travel_results.items():
    print(f"\n{technique} Prompting Results:")
    print("-" * 80)
    for i, (query, prediction, true_label) in enumerate(zip(travel_queries, predictions, true_labels_travel)):
        status = "✓" if prediction == true_label else "✗"
        print(f"{i+1}. {status} Query: {query[:50]}...")
        print(f"  Predicted: {prediction} | Actual: {true_label}\n")

    # Calculate and display accuracy
    print("\n" + "*80)
    print("ACCURACY COMPARISON")
    print("*80 + "\n")

    for technique, predictions in travel_results.items():
        correct = sum(1 for pred, true in zip(predictions, true_labels_travel) if pred == true)
        accuracy = (correct / len(true_labels_travel)) * 100
        print(f"{technique:15} Accuracy: {accuracy:6.2f}% ({correct}/{len(true_labels_travel)} correct)")

print("\n" + "*80)

```

Output:

```
=====
```

```
TRAVEL QUERY CLASSIFICATION RESULTS
```

```
=====
```

```
Zero-Shot Prompting Results:
```

1. ✓ Query: I want to book a flight from New York to London ne...
Predicted: Flight Booking | Actual: Flight Booking
2. ✓ Query: What's the best time to visit Paris?...
Predicted: General Travel Info | Actual: General Travel Info
3. ✓ Query: Can I cancel my hotel reservation?...
Predicted: Cancellation | Actual: Cancellation
4. ✓ Query: I need a 5-star hotel in Tokyo for 3 nights....
Predicted: Hotel Booking | Actual: Hotel Booking
5. ✓ Query: How do I get a refund for my flight?...
Predicted: Cancellation | Actual: Cancellation
6. ✓ Query: What are the visa requirements for India?...
Predicted: General Travel Info | Actual: General Travel Info
7. ✓ Query: Book me a direct flight to Sydney....
Predicted: Flight Booking | Actual: Flight Booking
8. ✓ Query: I want to cancel my booking immediately....
Predicted: Cancellation | Actual: Cancellation
9. ✓ Query: Where can I find budget hotels in Barcelona?...
Predicted: Hotel Booking | Actual: Hotel Booking
10. ✓ Query: Tell me about local attractions in Rome....
Predicted: General Travel Info | Actual: General Travel Info

```
One-Shot Prompting Results:
```

1. ✓ Query: I want to book a flight from New York to London ne...
Predicted: Flight Booking | Actual: Flight Booking
2. ✓ Query: What's the best time to visit Paris?...
Predicted: General Travel Info | Actual: General Travel Info

① ⌂ ④ ⌄ ⑥

2. ✓ Query: What's the best time to visit Paris?...
Predicted: General Travel Info | Actual: General Travel Info
3. ✓ Query: Can I cancel my hotel reservation?...
Predicted: Cancellation | Actual: Cancellation
4. ✓ Query: I need a 5-star hotel in Tokyo for 3 nights....
Predicted: Hotel Booking | Actual: Hotel Booking
5. ✓ Query: How do I get a refund for my flight?...
Predicted: Cancellation | Actual: Cancellation
6. ✓ Query: What are the visa requirements for India?...
Predicted: General Travel Info | Actual: General Travel Info
7. ✓ Query: Book me a direct flight to Sydney....
Predicted: Flight Booking | Actual: Flight Booking
8. ✓ Query: I want to cancel my booking immediately....
Predicted: Cancellation | Actual: Cancellation
9. ✓ Query: Where can I find budget hotels in Barcelona?...
Predicted: Hotel Booking | Actual: Hotel Booking
10. ✓ Query: Tell me about local attractions in Rome....
Predicted: General Travel Info | Actual: General Travel Info

Few-Shot Prompting Results:

1. ✓ Query: I want to book a flight from New York to London ne...
Predicted: Flight Booking | Actual: Flight Booking
2. ✓ Query: What's the best time to visit Paris?...
Predicted: General Travel Info | Actual: General Travel Info
3. ✓ Query: Can I cancel my hotel reservation?...
Predicted: Cancellation | Actual: Cancellation
4. ✓ Query: I need a 5-star hotel in Tokyo for 3 nights....
Predicted: Hotel Booking | Actual: Hotel Booking

! ⚡ ⚡ 4 ⚡ 4 ⚡ 6

```
Predicted: Hotel Booking | Actual: Hotel Booking
10. ✓ Query: Tell me about local attractions in Rome....
    Predicted: General Travel Info | Actual: General Travel Info

=====
ACCURACY COMPARISON
=====

Zero-Shot      Accuracy: 100.00% (10/10 correct)
One-Shot       Accuracy: 100.00% (10/10 correct)
Few-Shot        Accuracy: 100.00% (10/10 correct)

=====
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task 03:

Prompt

Create a basic test which shows how zero-shot and one-shot and few-shot methods work for detecting programming question types. The system needs to categorize each incoming query into one of the following four categories: Syntax Error and Logic Error and Optimization and Conceptual Question. The task requires creation of coding questions which will be tested using three different prompting methods to showcase their performance. The solution requires using simple language together with explanatory comments which describe each process step.

AI_CODE:

```

# =====
# Programming Question Classification using Zero-Shot, One-Shot, and Few-Shot Prompting
# =====

# Sample programming questions
programming_questions = [
    "Why does my loop variable not update inside the function?", 
    "How do I optimize this nested loop to O(n log n)?",
    "What's the difference between == and is in Python?", 
    "My code runs but produces wrong output. Where's the bug?", 
    "Can you explain what recursion means?", 
    "I get a 'NameError: name not defined' error. How do I fix it?", 
    "How can I make this algorithm faster?", 
    "Why doesn't my if statement execute when I expect it to?", 
    "What are the pros and cons of using lists vs tuples?", 
    "My program crashes with 'IndexError: list index out of range'."
]

# True labels for evaluation
true_labels_programming = [
    "Logic Error", 
    "Optimization", 
    "Conceptual Question", 
    "Logic Error", 
    "Conceptual Question", 
    "Syntax Error", 
    "Optimization", 
    "Logic Error", 
    "Conceptual Question", 
    "Syntax Error"
]

# Zero-Shot Prompt: No examples, just category definition
zero_shot_programming = (
    "Classify the following programming question into one of these categories: "
    "Syntax Error, Logic Error, Optimization, Conceptual Question.\n"
    "Question: '{}'\n"
    "Category:"
)

for technique, predictions in programming_results.items():
    print(f"\n{technique} Prompting Results:")
    print("-" * 90)
    for i, (question, prediction, true_label) in enumerate(zip(programming_questions, predictions, true_labels_programming)):
        status = "✓" if prediction == true_label else "✗"
        print(f"{i+1}. {status} Question: {question[:55]}...")
        print(f"  Predicted: {prediction:20} | Actual: {true_label}\n")

# Calculate and display accuracy comparison
print("\n" + "="*90)
print("ACCURACY COMPARISON")
print("="*90 + "\n")

for technique, predictions in programming_results.items():
    correct = sum(1 for pred, true in zip(predictions, true_labels_programming) if pred == true)
    accuracy = (correct / len(true_labels_programming)) * 100
    print(f"{technique:15} Accuracy: {accuracy:6.2f}% ({correct}/{len(true_labels_programming)} {correct})")

print("\n" + "="*90)

```

Output:

```
PROGRAMMING QUESTION CLASSIFICATION RESULTS
=====
Zero-Shot Prompting Results:
-----
1. X Question: Why does my loop variable not update inside the functio...
   Predicted: Conceptual Question | Actual: Logic Error

2. ✓ Question: How do I optimize this nested loop to O(n log n)?...
   Predicted: Optimization      | Actual: Optimization

3. ✓ Question: What's the difference between == and is in Python?...
   Predicted: Conceptual Question | Actual: Conceptual Question

4. ✓ Question: My code runs but produces wrong output. Where's the bug...
   Predicted: Logic Error       | Actual: Logic Error

5. ✓ Question: Can you explain what recursion means?...
   Predicted: Conceptual Question | Actual: Conceptual Question

6. ✓ Question: I get a 'NameError: name not defined' error. How do I f...
   Predicted: Syntax Error      | Actual: Syntax Error

7. ✓ Question: How can I make this algorithm faster?...
   Predicted: Optimization      | Actual: Optimization

8. X Question: Why doesn't my if statement execute when I expect it to...
   Predicted: Conceptual Question | Actual: Logic Error

9. ✓ Question: What are the pros and cons of using lists vs tuples?...
   Predicted: Conceptual Question | Actual: Conceptual Question

10. ✓ Question: My program crashes with 'IndexError: list index out of ...
    Predicted: Syntax Error      | Actual: Syntax Error

-----
4. Navigate to Command          wrong output. Where's the bug...
   Scroll to Previous Command (Ctrl+UpArrow)  al: Logic Error
   Scroll to Next Command (Ctrl+DownArrow)

5. . . . . bision means?...
   Predicted: Conceptual Question | Actual: Conceptual Question

6. ✓ Question: I get a 'NameError: name not defined' error. How do I f...
   Predicted: Syntax Error       | Actual: Syntax Error

7. ✓ Question: How can I make this algorithm faster?...
   Predicted: Optimization      | Actual: Optimization

8. X Question: Why doesn't my if statement execute when I expect it to...
   Predicted: Conceptual Question | Actual: Logic Error

9. ✓ Question: What are the pros and cons of using lists vs tuples?...
   Predicted: Conceptual Question | Actual: Conceptual Question

10. ✓ Question: My program crashes with 'IndexError: list index out of ...
   Predicted: Syntax Error      | Actual: Syntax Error

=====
ACCURACY COMPARISON
=====

Zero-Shot      Accuracy: 80.00% (8/10 correct)
One-Shot       Accuracy: 80.00% (8/10 correct)
Few-Shot       Accuracy: 80.00% (8/10 correct)

=====
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```

Task_04

Prompt:

Generate a simple demonstration of zero-shot, one-shot, and few-shot prompting for classifying social media posts into Promotion, Complaint, Appreciation, and Inquiry categories. Create sample social media posts, apply all three prompting techniques on the same test posts, and clearly display the results. The study aims to analyze informal short language while providing detailed explanations of each research process.

```
# =====
# Social Media Post Classification using Zero-Shot, One-Shot, and Few-Shot Prompting
# =====

# Sample social media posts
social_media_posts = [
    "Just got the new product and it's amazing! 10/10 would recommend!",
    "Why is your customer service so terrible? Been waiting 2 hours!",
    "Check out our summer sale - 50% off everything! Link in bio",
    "How do I track my order?",
    "This broke after one day. Total waste of money.",
    "Love this brand so much! They always deliver quality",
    "Limited time offer! Buy now and get free shipping",
    "Can someone help me with my account settings?",
    "Worst experience ever. Never buying again!",
    "Thanks for the quick response! Problem solved",
    "New collection just dropped! Shop now before it sells out",
    "Does this product come in other colors?"
]

# True labels for evaluation
true_labels_social = [
    "Appreciation",
    "Complaint",
    "Promotion",
    "Inquiry",
    "Complaint",
    "Appreciation",
    "Promotion",
    "Inquiry",
    "Complaint",
    "Appreciation",
    "Promotion",
    "Inquiry"
]

# Zero-Shot Prompt: No examples, just category definition
zero_shot_social = (
    "Classify the following social media post into one of these categories: "
    "Promotion, Complaint, Appreciation, Inquiry.\n"
    "Post: '{1}'\n"
)
```

```
# Apply classifier to all posts
for post in social_media_posts:
    social_results["Zero-Shot"].append(classify_social_post(post))
    social_results["One-Shot"].append(classify_social_post(post))
    social_results["Few-Shot"].append(classify_social_post(post))

# Display results in a formatted table
print("\n" + "="*100)
print("SOCIAL MEDIA POST CLASSIFICATION RESULTS")
print("="*100)

for technique, predictions in social_results.items():
    print(f"\n{technique} Prompting Results:")
    print("-" * 100)
    for i, (post, prediction, true_label) in enumerate(zip(social_media_posts, predictions, true_labels_social)):
        status = "✓" if prediction == true_label else "✗"
        print(f"\t{i+1}. {status} Post: {post[:60]}...")
        print(f"\t\tPredicted: {prediction:15} | Actual: {true_label}\n")

# Calculate and display accuracy comparison
print("\n" + "="*100)
print("ACCURACY COMPARISON")
print("="*100 + "\n")

for technique, predictions in social_results.items():
    correct = sum(1 for pred, true in zip(predictions, true_labels_social) if pred == true)
    accuracy = (correct / len(true_labels_social)) * 100
    print(f"\t{technique:15} Accuracy: {accuracy:6.2f}% ({correct}/{len(true_labels_social)} correct)")

print("\n" + "="*100)
print("\nRESEARCH INSIGHTS:")
print("-" * 100)
print("\nZero-Shot: Classifies based solely on category definitions without examples.")
print("\t\tGood for quick classification but may struggle with informal language nuances.")
print("\nOne-Shot: Uses a single example to guide classification.")
print("\t\tProvides minimal context but better than zero-shot in some cases.")
print("\nFew-Shot: Uses multiple diverse examples to establish clear patterns.")
print("\t\tBest performance as it captures various writing styles in social media")
```

Output:

```
=====
SOCIAL MEDIA POST CLASSIFICATION RESULTS
=====

Zero-Shot Prompting Results:

1. ✓ Post: Just got the new product and it's amazing! 10/10 would recom...
   Predicted: Appreciation | Actual: Appreciation

2. ✓ Post: Why is your customer service so terrible? Been waiting 2 hou...
   Predicted: Complaint     | Actual: Complaint

3. ✓ Post: Check out our summer sale - 50% off everything! Link in bio...
   Predicted: Promotion     | Actual: Promotion

4. ✓ Post: How do I track my order?...
   Predicted: Inquiry       | Actual: Inquiry

5. ✓ Post: This broke after one day. Total waste of money....
   Predicted: Complaint     | Actual: Complaint

6. ✓ Post: Love this brand so much! They always deliver quality...
   Predicted: Appreciation  | Actual: Appreciation

7. ✓ Post: Limited time offer! Buy now and get free shipping...
   Predicted: Promotion     | Actual: Promotion

8. ✓ Post: Can someone help me with my account settings?...
   Predicted: Inquiry       | Actual: Inquiry

9. X Post: Worst experience ever. Never buying again!...
   Predicted: Promotion     | Actual: Complaint

10. ✓ Post: Thanks for the quick response! Problem solved...
main@main ~ %
```

```
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding> python -u "c:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding\classification.py"
12. ✓ Post: Does this product come in other colors?...
   Predicted: Inquiry     | Actual: Inquiry

=====
ACCURACY COMPARISON
=====

Zero-Shot      Accuracy: 91.67% (11/12 correct)
One-Shot       Accuracy: 91.67% (11/12 correct)
Few-Shot       Accuracy: 91.67% (11/12 correct)

=====
RESEARCH INSIGHTS:
=====

Zero-Shot: Classifies based solely on category definitions without examples.
           Good for quick classification but may struggle with informal language nuances.

One-Shot:  Uses a single example to guide classification.
           Provides minimal context but better than zero-shot in some cases.

Few-Shot:  Uses multiple diverse examples to establish clear patterns.
           Best performance as it captures various writing styles in social media.

=====
PS C:\Users\Vivek\OneDrive\Desktop\3-2\AI-Assisted-Coding>
```