

Name: Achi Vaishnavi

Hall ticket:2303A51554

Batch:29

Assessment: 4.5

Task 1: Suppose that you work for a company that receives hundreds of customer emails daily. Management wants to automatically classify mails 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.

Prompt :

Classify the above below samples into one of the following categories: Billing, Technical Support, Feedback, Others.

Classify the above email samples into one of the following categories: Billing, Technical Support, Feedback, Others.Email: 'I have not received my invoice for last month.

Classify the below email samples into one of the following categories: Billing, Technical Support, Feedback, Others.Email: 'I have not received my invoice for last month. Billing, "Subject: Invoice #12345 Dear Customer, your invoice for the month of June is attached. Please make the payment by the due date. Best regards, Billing Team", "Technical Support", "Subject: Issue with Software Installation Hello Support Team, I am facing issues while installing the software on my computer. It shows an error code 404. Please assist. Thanks, User".

Code :

```
AI-45.py > classify_multiple_emails
1 #Management wants to automatically classify emails into categories like "Billing", "Technical Support", "Feedback", and "Others" before assigning them to
2 #Create or collect 10 short email samples, each belonging to one of the 4 categories.
3 #Classify the above below samples into one of the following categories: Billing, Technical Support, Feedback, Others.
4 def classify_email(email_content):
5     """Classify the email content into one of the categories: Billing, Technical Support, Feedback, Others.
6     Parameters:
7     email_content (str): The content of the email to classify.
8     Returns:
9     str: The category of the email."""
10    email_content = email_content.lower()
11
12    if any(keyword in email_content for keyword in ["invoice", "payment", "billing", "due date", "overdue"]):
13        return "Billing"
14    elif any(keyword in email_content for keyword in ["issue", "error", "support", "installation", "bug", "connectivity"]):
15        return "Technical Support"
16    elif any(keyword in email_content for keyword in ["feedback", "suggestion", "purchase", "quality", "service"]):
17        return "Feedback"
18    else:
19        return "Others"
20
21 email_samples = [
22     ("Billing", "Subject: Invoice #12345\nDear Customer, your invoice for the month of June is attached. Please make the payment by the due date.\nBest re
23     ("Technical Support", "Subject: Issue with Software Installation\nHello Support Team, I am facing issues while installing the software on my computer.
24     ("Feedback", "Subject: Feedback on Recent Purchase\nHi Team, I recently purchased a product from your store and I am very satisfied with the quality ar
25     ("Others", "Subject: Meeting Reminder\nDear Team, this is a reminder for our meeting scheduled tomorrow at 10 AM in the conference room. Please be on t
26     ("Billing", "Subject: Payment Confirmation\nDear Customer, we have received your payment for the invoice #67890. Thank you for your prompt payment.\nBe
27     ("Technical Support", "Subject: Network Connectivity Issue\nHello, I am experiencing frequent disconnections from the internet. Can you please help me
28     ("Feedback", "Subject: Suggestion for New Features\nHi Team, I would like to suggest a few new features for your app that I believe would enhance user
29     ("Others", "Subject: Holiday Announcement\nDear All, please note that the office will be closed next Friday in observance of the holiday. Enjoy your d
30     ("Billing", "Subject: Overdue Payment Notice\nDear Customer, our records indicate that your payment for invoice #54321 is overdue. Please make the paym
31     ("Technical Support", "Subject: Software Bug Report\nHello Support, I have encountered a bug in the latest version of your software. It crashes when I
32 ]
33 print(classify_email(email_samples[1][1])) # Output: Technical Support
34 print(classify_email(email_samples[4][1])) # Output: Billing
35 print(classify_email(email_samples[7][1])) # Output: Others
36
37 #Classify the above email samples into one of the following categories: Billing, Technical Support, Feedback, Others.Email: 'I have not received my invoice
38 def classify_email_single(email_content):
39     """Classify a single email content into one of the categories: Billing, Technical Support, Feedback, Others.
40     Parameters:
```

```

AI-4.5.py > classify_multiple_emails
37 def classify_email_single(email_content):
38     Parameters:
39         email_content (str): The content of the email to classify.
40     Returns:
41         str: The category of the email.
42     """
43     email_content = email_content.lower()
44
45     if any(keyword in email_content for keyword in ["invoice", "payment", "billing", "due date", "overdue"]):
46         return "Billing"
47     elif any(keyword in email_content for keyword in ["issue", "error", "support", "installation", "bug", "connectivity"]):
48         return "Technical Support"
49     elif any(keyword in email_content for keyword in ["feedback", "suggestion", "purchase", "quality", "service"]):
50         return "Feedback"
51     else:
52         return "Others"
53
54 # Example Usage:
55 email_to_classify = "I have not received my invoice for last month."
56 print(classify_email_single(email_to_classify)) # Output: Billing
57
58 #Classify the below email samples into one of the following categories: Billing, Technical Support, Feedback, Others.Email: 'I have not received my invoice
59 def classify_multiple_emails(email_contents):
60     """Classify multiple email contents into one of the categories: Billing, Technical Support, Feedback, Others.
61     Parameters:
62         email_contents (list): A list of email contents to classify.
63     Returns:
64         list: A list of categories corresponding to each email.
65     """
66     categories = []
67     for email_content in email_contents:
68         email_content = email_content.lower()
69
70         if any(keyword in email_content for keyword in ["invoice", "payment", "billing", "due date", "overdue"]):
71             categories.append("Billing")
72         elif any(keyword in email_content for keyword in ["issue", "error", "support", "installation", "bug", "connectivity"]):
73             categories.append("Technical Support")
74         elif any(keyword in email_content for keyword in ["feedback", "suggestion", "purchase", "quality", "service"]):
75             categories.append("Feedback")
76         else:
77             categories.append("Others")
78     return categories
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Output:

```

Technical Support
Billing
Others
Billing
Billing
○ ['Billing', 'Technical Support']

```

Justification:

This task demonstrates how prompt engineering can automate email sorting without the need to build a custom machine learning model. Zero-shot prompting performs well for clearly defined emails, but may struggle when categories overlap. One-shot prompting improves the model's understanding by providing a reference example. Few-shot prompting achieves higher accuracy by enabling the model to learn category patterns from multiple examples. Overall, this approach reduces manual effort and significantly improves operational efficiency.

Task 2: Travel Query Classification

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.
- Compare response consistency.

Prompt :

A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info. Tasks: a. Prepare labeled travel queries.

Classify the travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New York next week.",

Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New York next week.", "Can you help me find a hotel in Paris?"

Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "Can you help me find a hotel in Paris?" "I would like to cancel my reservation for tomorrow.", "What are the travel restrictions for Italy?"

Code:

```
AI-4.5.py > classify_multiple_travel_queries
116 def classify_single_travel_query(query):
117     """
124     if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
125         return "Flight Booking"
126     elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
127         return "Hotel Booking"
128     elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
129         return "Cancellation"
130     else:
131         return "General Travel Info"
132 # Example Usage:
133 single_query = "Can you help me find a hotel in Paris?"
134 print(classify_single_travel_query(single_query)) # Output: Hotel Booking
135
136 #Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight
137 def classify_multiple_travel_queries(queries):
138     """Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
139     Parameters:
140     queries (list): A list of travel queries to classify.
141     Returns:
142     list: A list of categories corresponding to each travel query."""
143     categories = []
144     for query in queries:
145         query = query.lower()
146
147         if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
148             categories.append("Flight Booking")
149         elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
150             categories.append("Hotel Booking")
151         elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
152             categories.append("Cancellation")
153         else:
154             categories.append("General Travel Info")
155     return categories
156 # Example Usage:
157 queries_to_classify = [
158     "I need to book a flight to New York next week.",
159     "Can you help me find a hotel in Paris?"
160 ]
```

```

AI-4.5.py > classify_multiple_travel_queries
84
85 #A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info. Tasks: a. Prepare labeled travel queri
86 def classify_travel_query(query):
87     """Classify the travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
88     Parameters:
89     query (str): The travel query to classify.
90     Returns:
91     str: The category of the travel query."""
92     query = query.lower()
93
94     if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
95         return "Flight Booking"
96     elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
97         return "Hotel Booking"
98     elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
99         return "Cancellation"
100    else:
101        return "General Travel Info"
102
103 # Example Usage:
104 travel_queries = [
105     "I need to book a flight to New York next week.",
106     "Can you help me find a hotel in Paris?",
107     "I would like to cancel my reservation for tomorrow.",
108     "What are the travel restrictions for Italy?"
109 ]
110 for query in travel_queries:
111     print(f"Query: {query}\nCategory: {classify_travel_query(query)}\n")
112 # Output: # Query: I need to book a flight to New York next week.
113 # Category: Flight Booking
114
115 #Classify the travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New Y
116 def classify_single_travel_query(query):
117     """Classify a single travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
118     Parameters:
119     query (str): The travel query to classify.
120     Returns:
121     str: The category of the travel query."""
122     query = query.lower()

```

```

#Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "Can you help me find a hot
def classify_more_travel_queries(queries):
    """Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
    Parameters:
    queries (list): A list of travel queries to classify.
    Returns:
    list: A list of categories corresponding to each travel query."""
    categories = []
    for query in queries:
        query = query.lower()

        if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
            categories.append("Flight Booking")
        elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
            categories.append("Hotel Booking")
        elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
            categories.append("Cancellation")
        else:
            categories.append("General Travel Info")
    return categories
# Example Usage:
more_queries_to_classify = [
    "Can you help me find a hotel in Paris?",
    "I would like to cancel my reservation for tomorrow.",
    "What are the travel restrictions for Italy?"
]
print(classify_more_travel_queries(more_queries_to_classify)) # Output: ['Hotel Booking', 'Cancellation', 'General Travel Info']

```

Output:

```

Query: I need to book a flight to New York next week.
Category: Flight Booking

Query: Can you help me find a hotel in Paris?
Category: Hotel Booking

Query: I would like to cancel my reservation for tomorrow.
Category: Cancellation

Query: What are the travel restrictions for Italy?
Category: General Travel Info

Hotel Booking
['Flight Booking', 'Hotel Booking']
['Hotel Booking', 'Cancellation', 'General Travel Info']
PS D:\AI>

```

Justification:

Travel-related queries often use similar language to express different user intents. Because zero-shot prompting lacks contextual examples, it may misclassify such queries. One-shot prompting improves intent detection by providing a basic reference example. Few-shot prompting further enhances consistency and accuracy by exposing the model to multiple

intent patterns. This task highlights the importance of contextual examples in building effective user-facing assistants.

Task 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.

Prompt :

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.

Code:

```
191 #A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question. Tasks:a. Prepare coding-related user qu
192 def classify_coding_query(query):
193     """Classify the coding query into one of the categories: Syntax Error, Logic Error, Optimization, Conceptual Question.
194     Parameters:
195     query (str): The coding query to classify.
196     Returns:
197     str: The category of the coding query."""
198     query = query.lower()
199
200     if any(keyword in query for keyword in ["syntax error", "unexpected indent", "missing parenthesis", "invalid syntax"]):
201         return "Syntax Error"
202     elif any(keyword in query for keyword in ["logic error", "wrong output", "incorrect result", "bug"]):
203         return "Logic Error"
204     elif any(keyword in query for keyword in ["optimize", "performance", "efficiency", "speed up"]):
205         return "Optimization"
206     elif any(keyword in query for keyword in ["how to", "what is", "explain", "concept"]):
207         return "Conceptual Question"
208     else:
209         return "Others"
210
211 # Example Usage:
212 coding_queries = [
213     "I am getting a syntax error when I run my Python code.",
214     "My program is producing the wrong output, what could be the logic error?",
215     "How can I optimize my code for better performance?",
216     "Can you explain the concept of recursion in programming?"
217 ]
218 for query in coding_queries:
219     print(f"Query: {query}\nCategory: {classify_coding_query(query)}\n")
220 # Output:# Query: I am getting a syntax error when I run my Python code.
221 # Category: Syntax Error
222 # Query: My program is producing the wrong output, what could be the logic error?
223 # Category: Logic Error
224 # Query: How can I optimize my code for better performance?
225 # Category: Optimization
226 # Query: Can you explain the concept of recursion in programming?
227 # Category: Conceptual Question
```

Output:


```
Query: I am getting a syntax error when I run my Python code.  
Category: Syntax Error  
  
Query: My program is producing the wrong output, what could be the logic error?  
Category: Logic Error  
  
Query: How can I optimize my code for better performance?  
Category: Optimization  
  
Query: Can you explain the concept of recursion in programming?  
Category: Conceptual Question  
  
PS D:\AI>
```

Justification :

Programming-related queries require strong technical and contextual understanding. Zero-shot prompting may confuse syntax-related and logic-related issues due to the absence of examples. One-shot prompting provides initial guidance to the model. Few-shot prompting significantly improves technical accuracy by exposing the model to multiple coding scenarios. This demonstrates how examples enhance domain-specific query classification.

Task 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.

Prompt :

Social Media Post Categorization, A social media analytics tool must classify posts into Promotion, Complaint, Appreciation, or Inquiry. 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.

Code:

```
229 #Social Media Post Categorization
230 # Scenario:
231 # A social media analytics tool must classify posts into Promotion,
232 # Complaint, Appreciation, or Inquiry.
233 # Tasks:
234 # 1. Prepare sample social media posts.
235 # 2. Use Zero-shot prompting.
236 # 3. Use One-shot prompting.
237 # 4. Use Few-shot prompting.
238 # 5. Analyze informal language handling.
239 def classify_social_media_post(post):
240     """Classify the social media post into one of the categories: Promotion, Complaint, Appreciation, Inquiry.
241     Parameters:
242     post (str): The social media post to classify.
243     Returns:
244     str: The category of the social media post."""
245     post = post.lower()
246
247     if any(keyword in post for keyword in ["buy now", "sale", "discount", "offer", "promo"]):
248         return "Promotion"
249     elif any(keyword in post for keyword in ["not happy", "disappointed", "bad service", "complaint", "issue"]):
250         return "Complaint"
251     elif any(keyword in post for keyword in ["thank you", "great job", "love it", "appreciate", "awesome"]):
252         return "Appreciation"
253     elif any(keyword in post for keyword in ["how to", "where can i", "what is", "help me"]):
254         return "Inquiry"
255     else:
256         return "Others"
257
258 # Example Usage:
259 social_media_posts = [
260     "Huge sale on all products! Buy now and save big!",
261     "I'm really disappointed with the service I received today.",
262     "Thank you for the amazing support! You guys are awesome!",
263     "Can someone help me with my account settings?",
264     "Just wanted to share how much I love this new app!"
265 ]
266 for post in social_media_posts:
267     print(f"Post: {post}\nCategory: {classify_social_media_post(post)}\n")
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
```

Output:

```
• Post: Huge sale on all products! Buy now and save big!
  Category: Promotion

Post: I'm really disappointed with the service I received today.
Category: Complaint

Post: Thank you for the amazing support! You guys are awesome!
Category: Appreciation

Post: Can someone help me with my account settings?
Category: Inquiry

Post: Just wanted to share how much I love this new app!
Category: Others
```

Justification:

Social media posts often contain informal language, slang, and emojis, making tone and intent harder to interpret. Zero-shot prompting may struggle to classify such content accurately. One-shot prompting provides limited improvement by offering a basic reference.

Few-shot prompting handles informal and unstructured expressions more effectively. This task demonstrates the importance of examples in understanding unstructured text.