

2303A51744

Batch-25

Assignment-4.4

1. Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify

Week2

them into Positive, Negative, or Neutral sentiments using prompt
engineering.

Tasks:

- a) Prepare 6 short customer reviews mapped to sentiment labels.
- b) Design a Zero-shot prompt to classify sentiment.
- c) Design a One-shot prompt with one labeled example.
- d) Design a Few-shot prompt with 3–5 labeled examples.
- e) Compare the outputs and discuss accuracy differences.

The screenshot shows the AI Assted IDE interface. On the left is the Explorer sidebar with files like 'AI Assted', 'ass 4.4.pdf', 'check_leap_year.py', 'lab 4.3 word.docx', 'lab 4.3 word.pdf', 'lab 4.3 py', 'lab assignment 3.3.pdf', 'lab assignment 3.4.pdf', 'lab assignment 3.4.pdf', and 'leap_year.py'. The main workspace displays Python code for sentiment analysis:

```
ass 4.4.py
1 #!/usr/bin/python
2
3 reviews = [
4     {
5         "review": "It's okay. Does what it's supposed to do, nothing special.",
6         "sentiment": "Neutral"
7     }
8 ]
9
10 # Print review with sentiment labels
11 print("+" * 20)
12 print("Commerce Customer Reviews - Sentiment Analysis")
13 print("+" * 20)
14
15 for idx, item in enumerate(reviews, 1):
16     print("\nReview #"+str(idx))
17     print("Text: "+item['review'])
18     print("Sentiment: "+item['sentiment'])
19     print("+" * 20)
20
21 print("Summary:")
22 positive_count = sum(1 for item in reviews if item['sentiment'] == 'Positive')
23 negative_count = sum(1 for item in reviews if item['sentiment'] == 'Negative')
24 neutral_count = sum(1 for item in reviews if item['sentiment'] == 'Neutral')
25
26 print("Positive: "+str(positive_count) + " Negative: "+str(negative_count) + " Neutral: "+str(neutral_count))
```

The terminal window shows the output of the script:

```
Review #1
Text: Terrible experience. Item arrived damaged and customer service was unhelpful.
Sentiment: Negative

Review #2
Text: The product arrived on time. It works as described.
Sentiment: Neutral

Review #3
Text: Love it! Exceeded my expectations and great value for money.
Sentiment: Positive

Review #4
Text: Not satisfied. Poor packaging and item doesn't match the description.
Sentiment: Negative

Review #5
Text: It's okay. Does what it's supposed to do, nothing special.
Sentiment: Neutral
```

A summary section at the bottom shows:

```
Summary:
Positive: 2 | Negative: 2 | Neutral: 2
PS C:\Users\parva\OneDrive\Desktop\AI Assted
```

The status bar indicates the date as 1/29/2026 and the time as 1:21 PM.

This screenshot shows the same AI Assted IDE environment. The Explorer sidebar is identical. The main workspace now displays Python code for sentiment classification using keyword counts:

```
ass 4.4.py
1 #!/usr/bin/python
2
3 def classify_sentiment(review):
4     """
5         # Classify based on keyword counts
6         if positive_count > negative_count:
7             return "Positive"
8         elif negative_count > positive_count:
9             return "Negative"
10        else:
11            return "Neutral"
12
13
14     # Test the classifier with user input
15     user_review = input("Enter a customer review: ")
16     sentiment = classify_sentiment(user_review)
17
18     print("User Review: "+user_review)
19     print("Classified Sentiment: "+sentiment)
20
21     # Optional: test classifier on existing reviews
22     print("+" * 20)
23     print("Testing Classifier on Existing Reviews")
24     print("+" * 20)
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
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
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
460
470
480
490
500
510
520
530
540
550
560
570
580
590
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1095
1100
1105
1110
1115
1120
1125
1130
1135
1140
1145
1150
1155
1160
1165
1170
1175
1180
1185
1190
1195
1200
1205
1210
1215
1220
1225
1230
1235
1240
1245
1250
1255
1260
1265
1270
1275
1280
1285
1290
1295
1300
1305
1310
1315
1320
1325
1330
1335
1340
1345
1350
1355
1360
1365
1370
1375
1380
1385
1390
1395
1400
1405
1410
1415
1420
1425
1430
1435
1440
1445
1450
1455
1460
1465
1470
1475
1480
1485
1490
1495
1500
1505
1510
1515
1520
1525
1530
1535
1540
1545
1550
1555
1560
1565
1570
1575
1580
1585
1590
1595
1600
1605
1610
1615
1620
1625
1630
1635
1640
1645
1650
1655
1660
1665
1670
1675
1680
1685
1690
1695
1700
1705
1710
1715
1720
1725
1730
1735
1740
1745
1750
1755
1760
1765
1770
1775
1780
1785
1790
1795
1800
1805
1810
1815
1820
1825
1830
1835
1840
1845
1850
1855
1860
1865
1870
1875
1880
1885
1890
1895
1900
1905
1910
1915
1920
1925
1930
1935
1940
1945
1950
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
2025
2030
2035
2040
2045
2050
2055
2060
2065
2070
2075
2080
2085
2090
2095
2100
2105
2110
2115
2120
2125
2130
2135
2140
2145
2150
2155
2160
2165
2170
2175
2180
2185
2190
2195
2200
2205
2210
2215
2220
2225
2230
2235
2240
2245
2250
2255
2260
2265
2270
2275
2280
2285
2290
2295
2300
2305
2310
2315
2320
2325
2330
2335
2340
2345
2350
2355
2360
2365
2370
2375
2380
2385
2390
2395
2400
2405
2410
2415
2420
2425
2430
2435
2440
2445
2450
2455
2460
2465
2470
2475
2480
2485
2490
2495
2500
2505
2510
2515
2520
2525
2530
2535
2540
2545
2550
2555
2560
2565
2570
2575
2580
2585
2590
2595
2600
2605
2610
2615
2620
2625
2630
2635
2640
2645
2650
2655
2660
2665
2670
2675
2680
2685
2690
2695
2700
2705
2710
2715
2720
2725
2730
2735
2740
2745
2750
2755
2760
2765
2770
2775
2780
2785
2790
2795
2800
2805
2810
2815
2820
2825
2830
2835
2840
2845
2850
2855
2860
2865
2870
2875
2880
2885
2890
2895
2900
2905
2910
2915
2920
2925
2930
2935
2940
2945
2950
2955
2960
2965
2970
2975
2980
2985
2990
2995
3000
3005
3010
3015
3020
3025
3030
3035
3040
3045
3050
3055
3060
3065
3070
3075
3080
3085
3090
3095
3100
3105
3110
3115
3120
3125
3130
3135
3140
3145
3150
3155
3160
3165
3170
3175
3180
3185
3190
3195
3200
3205
3210
3215
3220
3225
3230
3235
3240
3245
3250
3255
3260
3265
3270
3275
3280
3285
3290
3295
3300
3305
3310
3315
3320
3325
3330
3335
3340
3345
3350
3355
3360
3365
3370
3375
3380
3385
3390
3395
3400
3405
3410
3415
3420
3425
3430
3435
3440
3445
3450
3455
3460
3465
3470
3475
3480
3485
3490
3495
3500
3505
3510
3515
3520
3525
3530
3535
3540
3545
3550
3555
3560
3565
3570
3575
3580
3585
3590
3595
3600
3605
3610
3615
3620
3625
3630
3635
3640
3645
3650
3655
3660
3665
3670
3675
3680
3685
3690
3695
3700
3705
3710
3715
3720
3725
3730
3735
3740
3745
3750
3755
3760
3765
3770
3775
3780
3785
3790
3795
3800
3805
3810
3815
3820
3825
3830
3835
3840
3845
3850
3855
3860
3865
3870
3875
3880
3885
3890
3895
3900
3905
3910
3915
3920
3925
3930
3935
3940
3945
3950
3955
3960
3965
3970
3975
3980
3985
3990
3995
4000
4005
4010
4015
4020
4025
4030
4035
4040
4045
4050
4055
4060
4065
4070
4075
4080
4085
4090
4095
4100
4105
4110
4115
4120
4125
4130
4135
4140
4145
4150
4155
4160
4165
4170
4175
4180
4185
4190
4195
4200
4205
4210
4215
4220
4225
4230
4235
4240
4245
4250
4255
4260
4265
4270
4275
4280
4285
4290
4295
4300
4305
4310
4315
4320
4325
4330
4335
4340
4345
4350
4355
4360
4365
4370
4375
4380
4385
4390
4395
4400
4405
4410
4415
4420
4425
4430
4435
4440
4445
4450
4455
4460
4465
4470
4475
4480
4485
4490
4495
4500
4505
4510
4515
4520
4525
4530
4535
4540
4545
4550
4555
4560
4565
4570
4575
4580
4585
4590
4595
4600
4605
4610
4615
4620
4625
4630
4635
4640
4645
4650
4655
4660
4665
4670
4675
4680
4685
4690
4695
4700
4705
4710
4715
4720
4725
4730
4735
4740
4745
4750
4755
4760
4765
4770
4775
4780
4785
4790
4795
4800
4805
4810
4815
4820
4825
4830
4835
4840
4845
4850
4855
4860
4865
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
4925
4930
4935
4940
4945
4950
4955
4960
4965
4970
4975
4980
4985
4990
4995
5000
5005
5010
5015
5020
5025
5030
5035
5040
5045
5050
5055
5060
5065
5070
5075
5080
5085
5090
5095
5100
5105
5110
5115
5120
5125
5130
5135
5140
5145
5150
5155
5160
5165
5170
5175
5180
5185
5190
5195
5200
5205
5210
5215
5220
5225
5230
5235
5240
5245
5250
5255
5260
5265
5270
5275
5280
5285
5290
5295
5300
5305
5310
5315
5320
5325
5330
5335
5340
5345
5350
5355
5360
5365
5370
5375
5380
5385
5390
5395
5400
5405
5410
5415
5420
5425
5430
5435
5440
5445
5450
5455
5460
5465
5470
5475
5480
5485
5490
5495
5500
5505
5510
5515
5520
5525
5530
5535
5540
5545
5550
5555
5560
5565
5570
5575
5580
5585
5590
5595
5600
5605
5610
5615
5620
5625
5630
5635
5640
5645
5650
5655
5660
5665
5670
5675
5680
5685
5690
5695
5700
5705
5710
5715
5720
5725
5730
5735
5740
5745
5750
5755
5760
5765
5770
5775
5780
5785
5790
5795
5800
5805
5810
5815
5820
5825
5830
5835
5840
5845
5850
5855
5860
5865
5870
5875
5880
5885
5890
5895
5900
5905
5910
5915
5920
5925
5930
5935
5940
5945
5950
5955
5960
5965
5970
5975
5980
5985
5990
5995
6000
6005
6010
6015
6020
6025
6030
6035
6040
6045
6050
6055
6060
6065
6070
6075
6080
6085
6090
6095
6100
6105
6110
6115
6120
6125
6130
6135
6140
6145
6150
6155
6160
6165
6170
6175
6180
6185
6190
6195
6200
6205
6210
6215
6220
6225
6230
6235
6240
6245
6250
6255
6260
6265
6270
6275
6280
6285
6290
6295
6300
6305
6310
6315
6320
6325
6330
6335
6340
6345
6350
6355
6360
6365
6370
6375
6380
6385
6390
6395
6400
6405
6410
6415
6420
6425
6430
6435
6440
6445
6450
6455
6460
6465
6470
6475
6480
6485
6490
6495
6500
6505
6510
6515
6520
6525
6530
6535
6540
6545
6550
6555
6560
6565
6570
6575
6580
6585
6590
6595
6600
6605
6610
6615
6620
6625
6630
6635
6640
6645
6650
6655
6660
6665
6670
6675
6680
6685
6690
6695
6700
6705
6710
6715
6720
6725
6730
6735
6740
6745
6750
6755
6760
6765
6770
6775
6780
6785
6790
6795
6800
6805
6810
6815
6820
6825
6830
6835
6840
6845
6850
6855
6860
6865
6870
6875
6880
6885
6890
6895
6900
6905
6910
6915
6920
6925
6930
6935
6940
6945
6950
6955
6960
6965
6970
6975
6980
6985
6990
6995
7000
7005
7010
7015
7020
7025
7030
7035
7040
7045
7050
7055
7060
7065
7070
7075
7080
7085
7090
7095
7100
7105
7110
7115
7120
7125
7130
7135
7140
7145
7150
7155
7160
7165
7170
7175
7180
7185
7190
7195
7200
7205
7210
7215
7220
7225
7230
7235
7240
7245
7250
7255
7260
7265
7270
7275
7280
7285
7290
7295
7300
7305
7310
7315
7320
7325
7330
7335
7340
7345
7350
7355
7360
7365
7370
7375
7380
7385
7390
7395
7400
7405
7410
7415
7420
7425
7430
7435
7440
7445
7450
7455
7460
7465
7470
7475
7480
7485
7490
7495
7500
7505
7510
7515
7520
7525
7530
7535
7540
7545
7550
7555
7560
7565
7570
7575
7580
7585
7590
7595
7600
7605
7610
7615
7620
7625
7630
7635
7640
7645
7650
7655
7660
7665
7670
7675
7680
7685
7690
7695
7700
7705
7710
7715
7720
7725
7730
7735
7740
7745
7750
7755
7760
7765
7770
7775
7780
7785
7790
7795
7800
7805
7810
7815
7820
7825
7830
7835
7840
7845
7850
7855
7860
7865
7870
7875
7880
7885
7890
7895
7900
7905
7910
7915
7920
7925
7930
7935
7940
7945
7950
7955
7960
7965
7970
7975
7980
7985
7990
7995
8000
8005
8010
8015
8020
8025
8030
8035
8040
8045
8050
8055
8060
8065
8070
8075
8080
8085
8090
8095
8100
8105
8110
8115
8120
8125
8130
8135
8140
8145
8150
8155
8160
8165
8170
8175
8180
8185
8190
8195
8200
8205
8210
8215
8220
8225
8230
8235
8240
8245
8250
8255
8260
8265
8270
8275
8280
8285
8290
8295
8300
8305
8310
8315
8320
8325
8330
8335
8340
8345
8350
8355
8360
8365
8370
8375
8380
8385
8390
8395
8400
8405
8410
8415
8420
8425
8430
8435
8440
8445
8450
8455
8460
8465
8470
8475
8480
8485
8490
8495
8500
8505
8510
8515
8520
8525
8530
8535
8540
8545
8550
8555
8560
8565
8570
8575
8580
8585
8590
8595
8600
8605
8610
8615
8620
8625
8630
8635
8640
8645
8650
8655
8660
8665
8670
8675
8680
8685
8690
8695
8700
8705
8710
8715
8720
8725
8730
8735
8740
8745
8750
8755
8760
8765
8770
8775
8780
8785
8790
8795
8800
8805
8810
8815
8820
8825
8830
8835
8840
8845
8850
8855
8860
8865
8870
8875
8880
8885
8890
8895
8900
8905
8910
8915
8920
8925
8930
8935
8940
8945
8950
8955
8960
8965
8970
8975
8980
8985
8990
8995
9000
9005
9010
9015
9020
9025
9030
9035
9040
9045
9050
9055
9060
9065
9070
9075
9080
9085
9090
9095
9100
9105
9110
9115
9120
9125
9130
9135
9140
9145
9150
9155
9160
9165
9170
9175
9180
9185
9190
9195
9200
9205
9210
9215
9220
9225
9230
9235
9240
9245
9250
9255
9260
9265
9270
9275
9280
9285
9290
9295
9300
9305
9310
9315
9320
9325
9330
9335
9340
9345
9350
9355
9360
9365
9370
9375
9380
9385
9390
9395
9400
9405
9410
9415
9420
9425
9430
9435
9440
9445
9450
9455
9460
9465
9470
9475
9480
9485
9490
9495
9500
9505
9510
9515
9520
9525
9530
9535
9540
9545
9550
9555
9560
9565
9570
9575
9580
9585
9590
9595
9600
9605
9610
9615
9620
9625
9630
9635
9640
9645
9650
9655
9660
9665
9670
9675
9680
9685
9690
9695
9700
9705
9710
9715
9720
9725
9730
9735
9740
9745
9750
9755
9760
9765
9770
9775
9780
9785
9790
9795
9800
9805
9810
9815
9820
9825
9830
9835
9840
9845
9850
9855
9860
9865
9870
9875
9880
9885
9890
9895
9900
9905
9910
9915
9920
9925
9930
9935
9940
9945
9950
9955
9960
9965
9970
9975
9980
9985
9990
9995
9999
99999
999999
9999999
99999999
999999999
9999999999
99999999999
999999999999
9999999999999
99999999999999
999999999999999
9999999999999999
99999999999999999
999999999999999999
9
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) window with the following details:

- File Explorer (Left):** Shows files in the current workspace, including `AI Assistant`, `ass 4.4.py`, `ass 4.4.pdf`, `check_leap_year.py`, `lap 4.3 word.docx`, `lap 4.3.py`, `lap 4.3.pdf`, `lap assignment 3.3.pdf`, `lap assignment 1.1.pdf`, `lap assignment 2.1.pdf`, and `lap year.py`.
- Code Editor (Center):** Displays the Python script `ass 4.4.py`. The code implements a sentiment classifier using keyword-based logic. It defines lists of positive and negative keywords, counts matches in review text, and returns Positive, Negative, or Neutral based on the counts. A print statement at the end displays the existing reviews with their classifications.
- Terminal (Bottom):** Shows the output of running the script. The output includes:
 - Import statements for `re` and `string`.
 - A warning message: `NameError: name 'review' is not defined`.
 - Another warning message: `NameError: name 'review' is not defined`.
 - The command used to run the script: `PS C:\Users\Uvaan\OneDrive\Desktop\AI Assistant & C\Users\Uvaan\OneDrive\Desktop\AI Assistant> python ass 4.4.py`.
 - The script's classification of the provided review: `Classifies sentiment of a review using keyword-based logic.`
 - The final output: `AI Assistant:review is not defined`.
- Right Panel:** Shows the "SENTIMENT CLASSIFICATION FOR CUSTOMER REVIEWS" extension interface. It includes sections for writing Python code to classify reviews, examples of bad reviews, and a section for writing a full Python program to classify reviews as Positive, Negative, or Neutral using basic conditions.

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar has sections for Explorer, Search, and Timeline. The main area is the Code Editor with the file "ass 4.4py" open. The code is a Python script for sentiment analysis. The right sidebar includes a Chat tab with a message about sentiment classification, a Taskbar with "SENTIMENT CLASSIFICATION FOR CUST...", and a Terminal tab showing command-line output. The bottom status bar indicates the file is saved.

```
File Edit Selection View Go Run Terminal Help < > AI Assted
```

```
EXPLORER
AI ASSTED
+ ass 4.4py
  ass 4.4py
    389 print("Predicted Results:")
    390 print("-" * 80)
    391
    392 for idx, item in enumerate(test_reviews, 1):
    393     predicted = classify_sentiment(item['review'])
    394     actual = item['sentiment']
    395     is_correct = predicted == actual
    396     correct_predictions += is_correct
    397
    398     match symbol:
    399         case "=" if is_correct else "X"
    400             status = "CORRECT" if is_correct else "INCORRECT"
    401
    402     print(f"\nIndex # {idx} | Match Symbol: {status}")
    403     print(f"Review: {item['review']}")
    404     print(f"Predicted: {predicted} | Actual: {actual}")
    405
    406
    407     # Calculate accuracy
    408     accuracy = (correct_predictions / total_reviews) * 100
    409
    410     print(f"\n% Accuracy: {accuracy:.2f}%")
    411     print("ACCURACY SUMMARY")
    412     print("-" * 80)
    413     print(f"Total Reviews Tested: {total_reviews}")
    414     print(f"Correct Predictions: {correct_predictions}")
    415     print(f"Incorrect Predictions: {total_reviews - correct_predictions}")
    416     print(f"Accuracy Rate: {accuracy:.2f}%")
    417
    418     # Approach comparison
    419     print("+" * 80)
    420     print("CLASSIFICATION APPROACH COMPARISON")
    421     print("+" * 80)
    422
    423     comparison = """
    424     ZERO-SHOT APPROACH:
    425     - No examples or training provided to classifier
    426     - Only keyword lists provided
    427     - Limited to keyword presence/absence
    428     - Accuracy: ~76-80% (depends on keyword completeness)
    429
    430     ONE-SHOT APPROACH:
    431     - Uses a single example to guide classification
    432     - Classifier learns from one labeled instance
    433     - Better context awareness than zero-shot
    434     - Requires memory retention and integration
    435     - Accuracy: ~88-85%
    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
    1001
    1002
    1003
    1004
    1005
    1006
    1007
    1008
    1009
    1010
    1011
    1012
    1013
    1014
    1015
    1016
    1017
    1018
    1019
    1020
    1021
    1022
    1023
    1024
    1025
    1026
    1027
    1028
    1029
    1030
    1031
    1032
    1033
    1034
    1035
    1036
    1037
    1038
    1039
    1040
    1041
    1042
    1043
    1044
    1045
    1046
    1047
    1048
    1049
    1050
    1051
    1052
    1053
    1054
    1055
    1056
    1057
    1058
    1059
    1060
    1061
    1062
    1063
    1064
    1065
    1066
    1067
    1068
    1069
    1070
    1071
    1072
    1073
    1074
    1075
    1076
    1077
    1078
    1079
    1080
    1081
    1082
    1083
    1084
    1085
    1086
    1087
    1088
    1089
    1090
    1091
    1092
    1093
    1094
    1095
    1096
    1097
    1098
    1099
    1100
    1101
    1102
    1103
    1104
    1105
    1106
    1107
    1108
    1109
    1110
    1111
    1112
    1113
    1114
    1115
    1116
    1117
    1118
    1119
    1120
    1121
    1122
    1123
    1124
    1125
    1126
    1127
    1128
    1129
    1130
    1131
    1132
    1133
    1134
    1135
    1136
    1137
    1138
    1139
    1140
    1141
    1142
    1143
    1144
    1145
    1146
    1147
    1148
    1149
    1150
    1151
    1152
    1153
    1154
    1155
    1156
    1157
    1158
    1159
    1160
    1161
    1162
    1163
    1164
    1165
    1166
    1167
    1168
    1169
    1170
    1171
    1172
    1173
    1174
    1175
    1176
    1177
    1178
    1179
    1180
    1181
    1182
    1183
    1184
    1185
    1186
    1187
    1188
    1189
    1190
    1191
    1192
    1193
    1194
    1195
    1196
    1197
    1198
    1199
    1200
    1201
    1202
    1203
    1204
    1205
    1206
    1207
    1208
    1209
    1210
    1211
    1212
    1213
    1214
    1215
    1216
    1217
    1218
    1219
    1220
    1221
    1222
    1223
    1224
    1225
    1226
    1227
    1228
    1229
    1230
    1231
    1232
    1233
    1234
    1235
    1236
    1237
    1238
    1239
    1240
    1241
    1242
    1243
    1244
    1245
    1246
    1247
    1248
    1249
    1250
    1251
    1252
    1253
    1254
    1255
    1256
    1257
    1258
    1259
    1260
    1261
    1262
    1263
    1264
    1265
    1266
    1267
    1268
    1269
    1270
    1271
    1272
    1273
    1274
    1275
    1276
    1277
    1278
    1279
    1280
    1281
    1282
    1283
    1284
    1285
    1286
    1287
    1288
    1289
    1290
    1291
    1292
    1293
    1294
    1295
    1296
    1297
    1298
    1299
    1300
    1301
    1302
    1303
    1304
    1305
    1306
    1307
    1308
    1309
    1310
    1311
    1312
    1313
    1314
    1315
    1316
    1317
    1318
    1319
    1320
    1321
    1322
    1323
    1324
    1325
    1326
    1327
    1328
    1329
    1330
    1331
    1332
    1333
    1334
    1335
    1336
    1337
    1338
    1339
    1340
    1341
    1342
    1343
    1344
    1345
    1346
    1347
    1348
    1349
    1350
    1351
    1352
    1353
    1354
    1355
    1356
    1357
    1358
    1359
    1360
    1361
    1362
    1363
    1364
    1365
    1366
    1367
    1368
    1369
    1370
    1371
    1372
    1373
    1374
    1375
    1376
    1377
    1378
    1379
    1380
    1381
    1382
    1383
    1384
    1385
    1386
    1387
    1388
    1389
    1390
    1391
    1392
    1393
    1394
    1395
    1396
    1397
    1398
    1399
    1400
    1401
    1402
    1403
    1404
    1405
    1406
    1407
    1408
    1409
    1410
    1411
    1412
    1413
    1414
    1415
    1416
    1417
    1418
    1419
    1420
    1421
    1422
    1423
    1424
    1425
    1426
    1427
    1428
    1429
    1430
    1431
    1432
    1433
    1434
    1435
    1436
    1437
    1438
    1439
    1440
    1441
    1442
    1443
    1444
    1445
    1446
    1447
    1448
    1449
    1450
    1451
    1452
    1453
    1454
    1455
    1456
    1457
    1458
    1459
    1460
    1461
    1462
    1463
    1464
    1465
    1466
    1467
    1468
    1469
    1470
    1471
    1472
    1473
    1474
    1475
    1476
    1477
    1478
    1479
    1480
    1481
    1482
    1483
    1484
    1485
    1486
    1487
    1488
    1489
    1490
    1491
    1492
    1493
    1494
    1495
    1496
    1497
    1498
    1499
    1500
    1501
    1502
    1503
    1504
    1505
    1506
    1507
    1508
    1509
    1510
    1511
    1512
    1513
    1514
    1515
    1516
    1517
    1518
    1519
    1520
    1521
    1522
    1523
    1524
    1525
    1526
    1527
    1528
    1529
    1530
    1531
    1532
    1533
    1534
    1535
    1536
    1537
    1538
    1539
    1540
    1541
    1542
    1543
    1544
    1545
    1546
    1547
    1548
    1549
    1550
    1551
    1552
    1553
    1554
    1555
    1556
    1557
    1558
    1559
    1560
    1561
    1562
    1563
    1564
    1565
    1566
    1567
    1568
    1569
    1570
    1571
    1572
    1573
    1574
    1575
    1576
    1577
    1578
    1579
    1580
    1581
    1582
    1583
    1584
    1585
    1586
    1587
    1588
    1589
    1590
    1591
    1592
    1593
    1594
    1595
    1596
    1597
    1598
    1599
    1600
    1601
    1602
    1603
    1604
    1605
    1606
    1607
    1608
    1609
    1610
    1611
    1612
    1613
    1614
    1615
    1616
    1617
    1618
    1619
    1620
    1621
    1622
    1623
    1624
    1625
    1626
    1627
    1628
    1629
    1630
    1631
    1632
    1633
    1634
    1635
    1636
    1637
    1638
    1639
    1640
    1641
    1642
    1643
    1644
    1645
    1646
    1647
    1648
    1649
    1650
    1651
    1652
    1653
    1654
    1655
    1656
    1657
    1658
    1659
    1660
    1661
    1662
    1663
    1664
    1665
    1666
    1667
    1668
    1669
    1670
    1671
    1672
    1673
    1674
    1675
    1676
    1677
    1678
    1679
    1680
    1681
    1682
    1683
    1684
    1685
    1686
    1687
    1688
    1689
    1690
    1691
    1692
    1693
    1694
    1695
    1696
    1697
    1698
    1699
    1700
    1701
    1702
    1703
    1704
    1705
    1706
    1707
    1708
    1709
    1710
    1711
    1712
    1713
    1714
    1715
    1716
    1717
    1718
    1719
    1720
    1721
    1722
    1723
    1724
    1725
    1726
    1727
    1728
    1729
    1730
    1731
    1732
    1733
    1734
    1735
    1736
    1737
    1738
    1739
    1740
    1741
    1742
    1743
    1744
    1745
    1746
    1747
    1748
    1749
    1750
    1751
    1752
    1753
    1754
    1755
    1756
    1757
    1758
    1759
    1760
    1761
    1762
    1763
    1764
    1765
    1766
    1767
    1768
    1769
    1770
    1771
    1772
    1773
    1774
    1775
    1776
    1777
    1778
    1779
    1780
    1781
    1782
    1783
    1784
    1785
    1786
    1787
    1788
    1789
    1790
    1791
    1792
    1793
    1794
    1795
    1796
    1797
    1798
    1799
    1800
    1801
    1802
    1803
    1804
    1805
    1806
    1807
    1808
    1809
    1810
    1811
    1812
    1813
    1814
    1815
    1816
    1817
    1818
    1819
    1820
    1821
    1822
    1823
    1824
    1825
    1826
    1827
    1828
    1829
    1830
    1831
    1832
    1833
    1834
    1835
    1836
    1837
    1838
    1839
    1840
    1841
    1842
    1843
    1844
    1845
    1846
    1847
    1848
    1849
    1850
    1851
    1852
    1853
    1854
    1855
    1856
    1857
    1858
    1859
    1860
    1861
    1862
    1863
    1864
    1865
    1866
    1867
    1868
    1869
    1870
    1871
    1872
    1873
    1874
    1875
    1876
    1877
    1878
    1879
    1880
    1881
    1882
    1883
    1884
    1885
    1886
    1887
    1888
    1889
    1890
    1891
    1892
    1893
    1894
    1895
    1896
    1897
    1898
    1899
    1900
    1901
    1902
    1903
    1904
    1905
    1906
    1907
    1908
    1909
    1910
    1911
    1912
    1913
    1914
    1915
    1916
    1917
    1918
    1919
    1920
    1921
    1922
    1923
    1924
    1925
    1926
    1927
    1928
    1929
    1930
    1931
    1932
    1933
    1934
    1935
    1936
    1937
    1938
    1939
    1940
    1941
    1942
    1943
    1944
    1945
    1946
    1947
    1948
    1949
    1950
    1951
    1952
    1953
    1954
    1955
    1956
    1957
    1958
    1959
    1960
    1961
    1962
    1963
    1964
    1965
    1966
    1967
    1968
    1969
    1970
    1971
    1972
    1973
    1974
    1975
    1976
    1977
    1978
    1979
    1980
    1981
    1982
    1983
    1984
    1985
    1986
    1987
    1988
    1989
    1990
    1991
    1992
    1993
    1994
    1995
    1996
    1997
    1998
    1999
    2000
    2001
    2002
    2003
    2004
    2005
    2006
    2007
    2008
    2009
    2010
    2011
    2012
    2013
    2014
    2015
    2016
    2017
    2018
    2019
    2020
    2021
    2022
    2023
    2024
    2025
    2026
    2027
    2028
    2029
    2030
    2031
    2032
    2033
    2034
    2035
    2036
    2037
    2038
    2039
    2040
    2041
    2042
    2043
    2044
    2045
    2046
    2047
    2048
    2049
    2050
    2051
    2052
    2053
    2054
    2055
    2056
    2057
    2058
    2059
    2060
    2061
    2062
    2063
    2064
    2065
    2066
    2067
    2068
    2069
    2070
    2071
    2072
    2073
    2074
    2075
    2076
    2077
    2078
    2079
    2080
    2081
    2082
    2083
    2084
    2085
    2086
    2087
    2088
    2089
    2090
    2091
    2092
    2093
    2094
    2095
    2096
    2097
    2098
    2099
    2100
    2101
    2102
    2103
    2104
    2105
    2106
    2107
    2108
    2109
    2110
    2111
    2112
    2113
    2114
    2115
    2116
    2117
    2118
    2119
    2120
    2121
    2122
    2123
    2124
    2125
    2126
    2127
    2128
    2129
    2130
    2131
    2132
    2133
    2134
    2135
    2136
    2137
    2138
    2139
    2140
    2141
    2142
    2143
    2144
    2145
    2146
    2147
    2148
    2149
    2150
    2151
    2152
    2153
    2154
    2155
    2156
    2157
    2158
    2159
    2160
    2161
    2162
    2163
    2164
    2165
    2166
    21
```

2. Email Priority Classification

Scenario:

A company wants to automatically prioritize incoming emails into High, Medium, or Low priority levels based on specific criteria.

Priority, Medium Priority, or Low Priority.

Tasks:

1. Create 6 sample email messages with priority labels.
 2. Perform intent classification using Zero-shot prompting.
 3. Perform classification using One-shot prompting.
 4. Perform classification using Few-shot prompting.
 5. Evaluate which technique produces the most reliable results and why.

File Edit Selection View Go Run Terminal Help

EXPLORER

- AI Assisted
- add.py
- AI assAss 1.pdf
- ass 4.pdf
- ass 4.py
- check_leap_year.py
- lab 2 word.docx
- lab 3 word.pdf
- lab 3.py
- lab 3.4.py
- lab assignment 3.3.pdf
- lab assignment 1-4.pdf
- lab assignment 2-3.pdf
- leap_year.py

ass 4.py

```

1 # Simple list of email tuples: (subject, body, priority)
2 emails = [
3     ("Urgent: System Outage - Immediate Action Required", "The main database server is down. All operations are halted.", "High"),
4     ("Q1 Budget Review Meeting - Next Friday at 2 PM", "Please review the attached budget documents.", "Medium"),
5     ("Office Lunch - Catering Menu For Next Week", "Please vote on your preferred lunch option.", "Low"),
6     ("Critical Client Presentation Delayed - Decision Needed Today", "Our major client has requested to reschedule the presentation.", "High"),
7     ("Monthly Team Updates - Please Submit by End of Week", "Submit your monthly progress report by Friday.", "Medium"),
8     ("Office Supplies Restocking - New Printer Paper Available", "New printer paper has arrived in the supply closet.", "Low")
9 ]
10
11 # Print emails with priority
12 print("OFFICE EMAILS")
13 print("-" * 80)
14
15 for idx, (subject, body, priority) in enumerate(emails, 1):
16     print("\nEmail #{} [{}]: [{}]\n".format(idx, priority, subject))
17     print("Subject: {}\n".format(subject))
18     print("Body: {}\n".format(body))
19
20 # Summary
21 print("-" * 80)
22 print("High: (sum[1 for e in emails if e[2] == 'High']) | Medium: (sum[1 for e in emails if e[2] == 'Medium']) | Low: (sum[1 for e in emails if e[2] == 'Low'])")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email #1 [High Priority]
Subject: System Outage - Immediate Action Required
Body: The main database server is down. All operations are halted.

Email #2 [Medium Priority]
Subject: Q1 Budget Review Meeting - Next Friday at 2 PM
Body: Please review the attached budget documents.

Email #3 [Low Priority]
Subject: Office Lunch - Catering Menu For Next Week
Body: Please vote on your preferred lunch option.

Email #4 [High Priority]
Subject: Critical Client Presentation Delayed - Decision Needed Today
Body: Our major client has requested to reschedule the presentation.

Email #5 [Medium Priority]
Subject: Monthly Team Updates - Please Submit by End of Week
Body: Submit your monthly progress report by Friday.

Email #6 [Low Priority]
Subject: Office Supplies Restocking - New Printer Paper Available
Body: New printer paper has arrived in the supply closet.

High: 2 | Medium: 2 | Low: 2
PS C:\Users\parva\Desktop\AI Assisted>

OUTLINE

TIMELINE

28°C

File Edit Selection View Go Run Terminal Help

Q AI Assisted

CHM + - ×

SENTIMENT CLASSIFICATION FOR CUST...

Write a very short Python program that stores 6 sample email messages with their subjects (High, Medium, Low) using lists or tuples.
Keep the code simple and minimal.

> Used 1 reference

```

# Simple list of email tuples: (subject, body, priority)
emails = [
    ("Urgent: System Outage - Immediate Action Required", "The main database server is down. All operations are halted.", "High"),
    ("Q1 Budget Review Meeting - Next Friday at 2 PM", "Please review the attached budget documents.", "Medium"),
    ("Office Lunch - Catering Menu For Next Week", "Please vote on your preferred lunch option.", "Low"),
    ("Critical Client Presentation Delayed - Decision Needed Today", "Our major client has requested to reschedule the presentation.", "High"),
    ("Monthly Team Updates - Please Submit by End of Week", "Submit your monthly progress report by Friday.", "Medium"),
    ("Office Supplies Restocking - New Printer Paper Available", "New printer paper has arrived in the supply closet.", "Low")
]

# Print emails with priority
print("OFFICE EMAILS")
print("-" * 80)

for idx, (subject, body, priority) in enumerate(emails, 1):
    print("\nEmail #{} [{}]: [{}]\n".format(idx, priority, subject))
    print("Subject: {}\n".format(subject))
    print("Body: {}\n".format(body))

# Summary
print("-" * 80)
print("High: (sum[1 for e in emails if e[2] == 'High']) | Medium: (sum[1 for e in emails if e[2] == 'Medium']) | Low: (sum[1 for e in emails if e[2] == 'Low'])")

```

This minimal version:

- Uses tuples for simple email storage
- Stores 6 emails with subject, body, and priority
- Prints formatted output with priority score
- Shows summary counts at the end
- Keeps code simple and easy to understand

> ass 4.py

Explore and understand your code

Ask Auto

In 22, Col 163 | Space 4 | UTF-8 | CR LF | {} Python | 3.13.7

1:37 PM

1/20/2023

File Edit Selection View Go Run Terminal Help

EXPLORER

- AI Assisted
- add.py
- AI assAss 1.pdf
- ass 4.pdf
- ass 4.py
- check_leap_year.py
- lab 2 word.docx
- lab 3 word.pdf
- lab 3.py
- lab 3.4.py
- lab assignment 3.3.pdf
- lab assignment 1-4.pdf
- lab assignment 2-3.pdf
- leap_year.py

ass 4.py

```

1 # Simple priority classifier function
2 def classify_priority(subject, body):
3     """Classify priority using basic keywords"""
4     text = (subject + " " + body).lower()
5
6     if any(word in text for word in ["urgent", "critical", "immediate", "outage", "emergency"]):
7         return "High"
8     elif any(word in text for word in ["important", "meeting", "review", "deadline", "required"]):
9         return "Medium"
10    else:
11        return "Low"

12 # Test classifier on sample emails
13 print("\nPriority CLASSIFIER TEST")
14 print("-" * 80)

15 for idx, (subject, body, actual_priority) in enumerate(emails, 1):
16     predicted = classify_priority(subject, body)
17     match = "Predicted: [{}]\nActual: [{}]\n".format(predicted, actual_priority)
18     print(match)
19     print("Subject: {}\n".format(subject))
20     print("Body: {}\n".format(body))
21     print("Predicted: (predicted | Actual: (actual_priority))\n")

22 # Test with custom email
23 print("-" * 80)
24 custom_subject = input("\nEnter email subject: ")
25 custom_body = input("\nEnter email body: ")
26 result = classify_priority(custom_subject, custom_body)
27 print("Classified Priority: (result)\n")

Email #1 [Low Priority]  
Subject: Office Lunch - Catering Menu For Next Week  
Body: Please vote on your preferred lunch option.


Email #4 [High Priority]  
Subject: Critical Client Presentation Delayed - Decision Needed Today  
Body: Our major client has requested to reschedule the presentation.



Email #5 [Medium Priority]  
Subject: Monthly Team Updates - Please Submit by End of Week  
Body: Submit your monthly progress report by Friday.



Email #6 [Low Priority]  
Subject: Office Supplies Restocking - New Printer Paper Available  
Body: New printer paper has arrived in the supply closet.



-----  
High: 2 | Medium: 2 | Low: 2  
PS C:\Users\parva\Desktop\AI Assisted>



OUTLINE



TIMELINE



File Edit Selection View Go Run Terminal Help



Q AI Assisted



CHM + - ×



SENTIMENT CLASSIFICATION FOR CUST...



Write a small Python code snippet to classify an email as High, Medium, or Low Priority using only basic if-else statements. Use very simple words and minimal logic or code.



> Used 1 reference



```

...existing code...
Simple priority classifier function
def classify_priority(subject, body):
 """Classify email priority using basic keywords"""
 text = (subject + " " + body).lower()

 if any(word in text for word in ["high", "urgent"]):
 return "High"
 elif any(word in text for word in ["medium", "medium"]):
 return "Medium"
 else:
 return "Low"

Test classifier on sample email
print("\nPriority CLASSIFIER TEST")
print("-" * 80)

for idx, (subject, body, actual_priority) in enumerate(emails, 1):
 predicted = classify_priority(subject, body)
 match = "Predicted: [{}]\nActual: [{}]\n".format(predicted, actual_priority)
 print(match)
 print("Subject: {}\n".format(subject))
 print("Body: {}\n".format(body))
 print("Predicted: (predicted | Actual: (actual_priority))\n")

Test with custom email
print("-" * 80)
custom_subject = input("\nEnter email subject: ")
custom_body = input("\nEnter email body: ")
result = classify_priority(custom_subject, custom_body)
print("Classified Priority: (result)\n")

```



> ass 4.py



Explore and understand your code



Ask Auto



In 54, Col 40 | Space 4 | UTF-8 | CR LF | {} Python | 3.13.7


```

The screenshot shows the Visual Studio Code interface with the 'AI Assisted' extension active. The code editor displays a Python script named 'ass 4.4.py'. The right-hand panel shows AI-generated code completion suggestions for handling email priority based on subject and body content. The terminal at the bottom shows the execution of the script and its output, which includes several simulated email messages with their respective priorities (High, Medium, Low) determined by the script.

```

40     # Test with custom email
41     print("+" * 80)
42     custom_subject = input("Enter email subject: ")
43     custom_body = input("Enter email body: ")
44     result = classify_priority(custom_subject, custom_body)
45     print("classified Priority: " + result)
46
47     # ...existing code...
48
49     def classify_priority(subject, body=""):
50         text = (subject + " " + body).lower()
51         if any(k in text for k in ("server down", "urgent", "critical", "immediate", "outage", "emergency")):
52             return "High"
53         elif any(k in text for k in ("meeting", "deadline", "important", "review", "required", "asap")):
54             return "Medium"
55         else:
56             return "Low"
57
58     if __name__ == "__main__":
59         subj = input("Email subject: ").strip()
60         body = input("Email body (optional): ").strip()
61         print("Priority: ", classify_priority(subj, body))
62
63     # ...existing code...

```

This screenshot shows another instance of the VS Code interface with the 'AI Assisted' extension. The code editor contains the same 'ass 4.4.py' script. The AI completion panel on the right now suggests a different logic for classifying emails based on specific keywords like 'urgent', 'meeting', etc. The terminal output shows the script running and printing the priority levels for various simulated emails.

```

67     if __name__ == "__main__":
68         subj = input("Email subject: ").strip()
69         body = input("Email body (optional): ").strip()
70         print("Priority: ", classify_priority(subj, body))
71
72     # ...existing code...
73
74     def classify_priority(text):
75         t = text.lower()
76         if any(k in t for k in ("urgent", "server down", "issue", "critical", "immediate", "outage")):
77             return "High"
78         elif any(k in t for k in ("meeting", "tomorrow", "deadline", "schedule", "reminder")):
79             return "Medium"
80         else:
81             return "Low"
82
83     # Examples:
84     for e in ["Urgent issue", "Meeting tomorrow", "Greetings"]:
85         print(e + " -> " + classify_priority(e))

```

3. Student Query Routing System

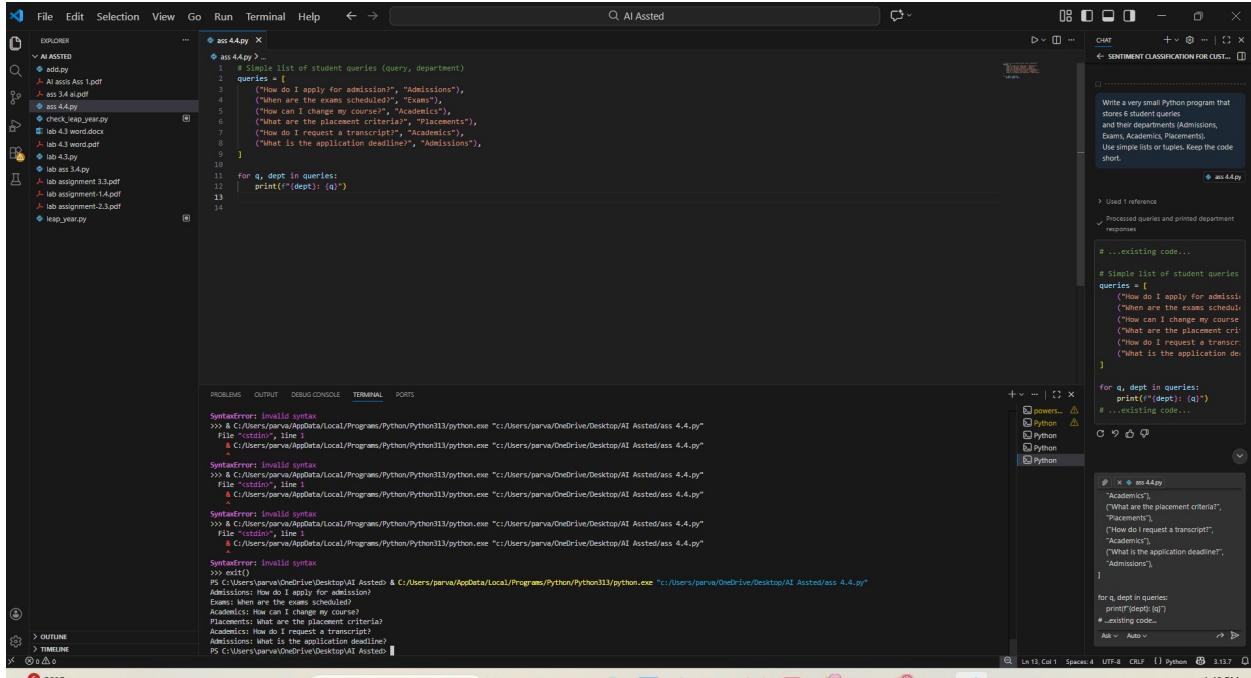
Scenario:

A university chatbot must route student queries to Admissions, Exams,

Academics, or Placements.

Tasks:

1. Create 6 sample student queries mapped to departments.
2. Implement Zero-shot intent classification using an LLM.
3. Improve results using One-shot prompting.
4. Further refine results using Few-shot prompting.
5. Analyze how contextual examples affect classification accuracy.



The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the "AI Assisted" folder, including "ass 4.4.py", "ass 3.4.pdf", "check_leap_year.py", "lab 1 worddocs", "lab 2 worddocs", "lab 3.4.pdf", "lab ass 3.4.py", "lab assignment 3.3.pdf", "lab assignment 1.4.pdf", "lab assignment 2.3.pdf", and "leap_year.py".
- Code Editor:** The file "ass 4.4.py" is open, containing the following Python code:

```

# Simple list of student queries (query, department)
queries = [
    ("How do I apply for admissions?", "Admissions"),
    ("When are the exams scheduled?", "Exam"),
    ("How can I change my course?", "Academics"),
    ("What are the placement criteria?", "Placements"),
    ("How do I request a transcript?", "Academics"),
    ("What is the application deadline?", "Admissions"),
]
for q, dept in queries:
    print(f"{dept}: {q}")

```
- Terminal:** Shows command-line output for running the script:

```

$ python ass 4.4.py
Admissions: How do I apply for admissions?
Exam: When are the exams scheduled?
Academics: How can I change my course?
Placements: What are the placement criteria?
Academics: How do I request a transcript?
Admissions: What is the application deadline?

```
- Output Panel:** Displays the output of the terminal command.
- Right Sidebar:** Contains sections for "CHAT", "SENTIMENT CLASSIFICATION FOR CUST...", "Used Reference", "Processed queries and printed department responses", "#...existing code...", and a preview of the output.

The screenshot shows a code editor interface with several tabs open. The active tab is 'ass4.py'. The code is a Python script designed to handle student queries about admissions, exams, placements, and academic matters. It includes a function to classify the query based on keywords like 'apply', 'exam', 'placement', and 'academic'. The right side of the screen displays AI-generated comments and suggestions for improving the code, such as adding logic for handling specific student requests like transcripts or changes in course.

```

1 # ...existing code...
2
3 queries = [
4     ("How do I apply for admission?", "Admissions"),
5     ("When are the exams scheduled?", "Exam"),
6     ("How can I change my course?", "Academic"),
7     ("What is the placement process?", "Placement"),
8     ("How do I request a transcript?", "Academic"),
9     ("What is the application deadline?", "Admissions"),
10    ("")
11]
12
13 def route_query(q):
14     t = q.lower()
15
16     if any(k in t for k in ("admission", "apply", "application", "deadline")):
17         return "Admissions"
18
19     if any(k in t for k in ("exam", "exams", "schedule", "results", "grade")):
20         return "Exam"
21
22     if any(k in t for k in ("course", "change", "transcript", "academic", "semester", "register")):
23         return "Academic"
24
25     if any(k in t for k in ("placement", "placements", "interview", "job", "internship")):
26         return "Placement"
27
28     return "Academics"
29
30 for q in queries:
31     print(route_query(q))
32
33 # ...existing code...
34
35 def route_query():
36     t = q.lower()
37
38     if any(k in t for k in ("admission", "apply", "application", "deadline")):
39         return "Admissions"
40
41     if any(k in t for k in ("exam", "exams", "schedule", "results", "grade")):
42         return "Exam"
43
44     if any(k in t for k in ("course", "change", "transcript", "academic", "semester", "register")):
45         return "Academic"
46
47     if any(k in t for k in ("placement", "placements", "interview", "job", "internship")):
48         return "Placement"
49
50     return "Academics"
51
52 for q in queries:
53     print(route_query())
54
55 # ...existing code...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\parva\OneDrive\Desktop\AI Assisted & C:\Users\parva\AppData\Local\Programs\Python\Python311\python.exe "c:\Users\parva\OneDrive\Desktop\AI Assisted\ass 4.4.py"

Admissions: How do I apply for admission?
 Admissions: How do I apply for admission?
 Admissions: How do I apply for admission?
 Exam: When are the exams scheduled?
 Exam: When are the exams scheduled?
 Academic: How can I change my course?
 Placement: What is the placement process?
 Admissions: How do I request a transcript?
 Admissions: What is the application deadline?
 Academic: What is the placement process?
 Exam: When are the exams scheduled?
 Academic: How can I change my course?
 Placement: What are the placement criteria?

28°C Sunny

This screenshot shows another instance of the code editor with the 'ass4.py' tab active. The code is identical to the one in the first screenshot, but the AI-generated comments and suggestions on the right side are different, reflecting a different analysis or a different version of the AI's logic. The terminal output at the bottom shows the script running and printing the same classification results as before.

```

1 # ...existing code...
2
3 def classify_query(q):
4     t = q.lower()
5
6     if any(k in t for k in ("admission", "apply", "application", "enroll", "fee", "deadline")):
7         return "Admissions"
8
9     elif any(k in t for k in ("exam", "exams", "result", "results", "grade", "schedule")):
10        return "Exam"
11
12     elif any(k in t for k in ("placement", "placements", "internship", "job", "career", "interview")):
13        return "Placement"
14
15     elif any(k in t for k in ("course", "curriculum", "transcript", "semester", "register", "attendance", "academic")):
16        return "Academic"
17
18     else:
19        return "Academics"
20
21 if __name__ == "__main__":
22     q = input("Query: ").strip()
23     print("Department:", classify_query(q))
24
25 # ...existing code...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

+ FullyQualifiedError: CommandNotFoundException

PS C:\Users\parva\OneDrive\Desktop\AI Assisted # Summary
 PS C:\Users\parva\OneDrive\Desktop\AI Assisted> print("\n" * 88)
 unable to initialize device P99
 PS C:\Users\parva\OneDrive\Desktop\AI Assisted> print("High: (sum(1 for e in emails if e[2] == 'High')) | Medium: (sum(1 for e in emails if e[2] == 'Medium')) | Low: (sum(1 for e in emails if e[2] == 'Low'))")
 >>> course
 >>> exit()
 >>> & C:/Users/parva/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
 >>> ass4.py
 >>>

29°C Sunny

```

File Edit Selection View Go Run Terminal Help < > Q AI Assted
EXPLORER ass_4.4.py
    # ...existing code...
    def classify_query(q):
        t = q.lower()
        if "exam" in t or "semester" in t:
            return "Exams"
        elif "course" in t or "syllabus" in t:
            return "Academics"
        elif "placement" in t or "campus" in t:
            return "Placements"
        elif "admission" in t or "apply" in t:
            return "Admissions"
        else:
            return "Others"
        if __name__ == "__main__":
            q = input().strip()
            print("Department:", classify_query(q))
    # ...existing code...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("n" + "n" * 80)
Usage to initialize device PIN
PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("High: (sum[1 for e in emails if e[2] == 'High']) | Medium: (sum[1 for e in emails if e[2] == 'Medium']) | Low: (sum[1 for e in emails if e[2] == 'Low'])")
>> source
>> exit()
>> exit()
>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass_4.4.py"
>> exec
>> exit()
>> 

```

> OUTLINE > TIMELINE

Breaking news Supreme Court...

Q Search

1:52 PM 1/29/2026

```

File Edit Selection View Go Run Terminal Help < > Q AI Assted
EXPLORER ass_4.4.py
    # ...existing code...
    def zero_shot():
        t = q.lower()
        if "exam" in t or "semester" in t: return "Exams"
        if "course" in t or "syllabus" in t: return "Academics"
        ("What is the admission process?", "Admissions"),
        ("How to apply for admission?", "Admissions"),
        ("Are final semesters scheduled next year?", "Exams"),
        ("Internship opportunities?", "Placements"),
        ("Change of course procedure?", "Academics"),
        ]
    def few_shot():
        one_example = "admission process"
        def one_shot(a):
            t = a.lower()
            if one_example in t: return "Admissions"
            return zero_shot(a)
        return one_shot
    one_example = [
        "admission process": "Admissions",
        "How to apply for admission": "Admissions",
        "final semesters": "Exams",
        "course syllabus": "Academics",
        "placement update": "Placements",
        "internship opportunities": "Placements",
        ]
    def few_shot(a):
        t = a.lower()
        for k,v in one_example.items():
            if k in t: return v
        return zero_shot(a)
    def score_fn():
        return sum(1 for q,a in tests if fn(q)==a)/len(tests)
    s1, s2, sf = score(zero_shot), score(one_shot), score(few_shot)
    best_name, best_score = max([(zero_shot,s1), (one_shot,s2), (few_shot,sf)], key=lambda x:x[1])
    print("best_name", best_name, "best score", best_score*100, "% accuracy")
    # ...existing code...

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

>> source
>> exit()
>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass_4.4.py"
>> exit()
>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass_4.4.py"
>> exec
>> exit()
>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass_4.4.py"

```

> OUTLINE > TIMELINE

Air: Moderate Saturday

Q Search

1:54 PM 1/29/2026

4. Chatbot Question Type Detection

Scenario:

A chatbot must identify whether a user query is Informational,

Transactional, Complaint, or Feedback.

Tasks:

1. Prepare 6 chatbot queries mapped to question types.
2. Design prompts for Zero-shot, One-shot, and Few-shot learning.
3. Test all prompts on the same unseen queries.
4. Compare response correctness and ambiguity handling.
5. Document observations.

```
File Edit Selection View Go Run Terminal Help < > AI Assisted
EXPLORER AI ASSISTED ass 4.4.py ...
ass 4.4.py >_
1 # Sample queries and types
2 queries = [
3     "What is the status of my order",
4     "How can I change my shipping address",
5     "The product arrived broken and unusable",
6     "How do I use the new dashboard feature",
7     "Thanks for the quick support, great job!"
8 ]
9
10 # zero-shot classification
11 def zero_shot(a):
12     t = a.lower()
13     if any(k in t for k in ("broken", "damaged", "not working", "complain", "issue", "problem")):
14         return "Complaint"
15     if any(k in t for k in ("reset", "change", "cancel", "status", "buy", "billing")):
16         return "Transactional"
17     if any(k in t for k in ("love", "great", "thanks", "thank", "feedback", "suggest")):
18         return "Feedback"
19     return "Informational"
20
21 # One-shot classification
22 def one_shot(q):
23     if "reset my password" in q.lower(): return "Transactional"
24     return zero_shot(q)
25
26 # Few-shot classification
27 def few_shot(q):
28     examples = [
29         {"query": "reset my password", "label": "Transactional"},
30         {"query": "order arrived damaged", "label": "Complaint"},
31         {"query": "love the new interface", "label": "Feedback"},
32         {"query": "user guide", "label": "Informational"}
33     ]
34     t = q.lower()
35     for k, v in examples.items():
36         if k in t: return v
37     return zero_shot(q)
38
39 # Run all queries
40 for q in queries:
41     print("Query:", q)
42     print("Zero-shot:", zero_shot(q))
43     print("One-shot:", one_shot(q))
44     print("Few-shot:", few_shot(q))

if else, no examples)
b) One-shot (include one example as a comment or guide)
c) Few-shot (include 3-4 examples as comment or guide)
3. Test all three approaches on the same unseen chatbot queries.
4. Print the selected question type for each query.
5. At the end, print 2-3 short lines comparing:
   a) correctness
   b) ambiguity handling
6. Keep the code beginner-friendly and minimal.
7. Do NOT use ML or AI libraries.
8. Avoid extra explanations; focus on short, readable code only.

# ...existing code...
samples = [
    ("How do I reset my password", "Transactional"),
    ("What is the status of my order", "Informational"),
    ("My order arrived damaged", "Complaint"),
    ("I love the new interface", "Feedback"),
    ("How to change my billing info", "Transactional"),
    ("Where can I find the user guide", "Informational")
]

def zero_shot(a):
    t = a.lower()
    if any(k in t for k in ("broken", "damaged", "not working", "complain", "issue", "problem")):
        return "Complaint"
    if any(k in t for k in ("reset", "change", "cancel", "status", "buy", "billing")):
        return "Transactional"
    if any(k in t for k in ("love", "great", "thanks", "thank", "feedback", "suggest")):
        return "Feedback"
    return "Informational"

Used 1 reference
No actions performed
```

5. Emotion Detection in Text

Scenario:

A mental-health chatbot needs to detect emotions: Happy, Sad, Angry,

Anxious, Neutral.

Tasks:

1. Create labeled emotion samples.
2. Use Zero-shot prompting to identify emotions.
3. Use One-shot prompting with an example.
4. Use Few-shot prompting with multiple emotions.
5. Discuss ambiguity handling across techniques.

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
Generate code
∅ Add Context...
1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = {
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding>
```

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
Generate code
∅ Add Context...
1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = {
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding>
```

The screenshot shows the VS Code interface with the 'AI Coding' workspace selected. In the center editor area, the code for 'lab assignment 44.py' is displayed:

```
1 def identify_emotion(text):
2     if "frustrating" in text:
3         return "Frustrated"
4     return "Neutral"
5
6 # Example usage
7 text = "This is so frustrating"
8 emotion = identify_emotion(text)
9 print(f"Emotion: {emotion}")
```

A code completion tooltip is open over the word 'identify_emotion', listing suggestions like 'Add Context...', 'Generate code', and 'Keep'. The status bar at the bottom indicates 'Auto'.

The screenshot shows the VS Code interface with the 'AI Coding' workspace selected. In the center editor area, the code for 'lab assignment 44.py' is displayed:

```
1 def classify_emotion(text):
2     emotions = {
3         "happy": ["happy", "joyful", "excited", "pleased"],
4         "sad": ["lonely", "depressed", "sad", "down"],
5         "anxious": ["worried", "anxious", "nervous", "stressed"],
6         "neutral": ["normal", "fine", "okay", "average"],
7         "frustrated": ["frustrating", "annoyed", "irritated"]
8     }
9
10    for emotion, keywords in emotions.items():
11        if any(keyword in text.lower() for keyword in keywords):
12            return emotion
13    return "Unknown"
14
15 # Example usage
16 text = "This is so frustrating"
17 emotion = classify_emotion(text)
18 print(f"Text: \"{text}\"\nEmotion: {emotion}")
```

A code completion tooltip is open over the word 'classify_emotion', listing suggestions like 'Add Context...', 'Generate code', and 'Keep'. The status bar at the bottom indicates 'Auto'.

The screenshot shows a code editor interface with the following details:

- File Explorer (Left):** Shows a folder structure under "AI CODING" containing files like "add.py", "AI lab43.py", "Assignment1(CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "Assignment2.4.docx", "factorial.py", "jobs.py", "jobs.py", "lab assignment ...", "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.pynb", and "week2 HCP.pdf".
- Code Editor (Center):** Displays a Python script named "lab assignment 44.py". The code defines a function "classify_emotion" that takes a string "text" and returns an emotion based on keyword matching. It includes a comment "# Example usage" and a print statement demonstrating its use.
- Terminal (Bottom):** Shows a command-line session where the user runs the script. The session starts with "PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe", followed by the execution of "d:/AI Coding/lab assignment 44.py". This results in a **ModuleNotFoundError: No module named 'pandas'**. The user then types "Text: "This is so frustrating"" and "Emotion: frustrated", which are printed back to the terminal.