```python
# CIS 511 NLP - Assignment 4.2 - Dialog Act Classification

"""
Created on Sat Apr 11 15:20:36 2020

@author: Siyu Yang
@unique name: siyuya
@UMID:76998080
"""

import sys
import math

#=== Functions ===#

#   1)  find the middle content between begin word and end word
def middle_content(begin, end, content):
    mid_content  =''
    if content.find(begin): # find the line start with begin words
        beginword = content[content.find(begin):content.rfind(end)]
        mid_content = beginword[len(begin):]
        return mid_content


#   2)  Parse Training File
def parse_file(filename):

    # dialog_dict: key = DialogAct, value = list of all words
    dialog_dict = dict()
    # number_dict: key = DialogAct, value = times in train data
    number_dict = dict()

    with open(filename, 'r', encoding='utf-8') as f:
        content = f.read()
        lines = content.split("\n")
        for line in lines:

            if len(line) > 2:
                prev_words = []
                label_answers = []

                if line.startswith("Student:"):
                    words = line.split(" ")
                    for word in words:
                        label_answers.append(word)

                prev_words = label_answers

                if line.startswith("Advisor:"):
                    dialog_act = middle_content("[", "]", line)

                    if dialog_act == "social" or dialog_act == "pull" or dialog_act
                    == "push":
                        dialog_act = ""
                    else:
                        if dialog_act not in number_dict:
                            number_dict[dialog_act] = 1
                        else:
                            number_dict[dialog_act] += 1

                #student's words and advisor's words are both written to
                if len(prev_words) != 0 and len(dialog_act) != 0:
                    if dialog_act in dialog_dict:
                        for word in prev_words:
                            dialog_dict[dialog_act].append(word)
                    else:
                        dialog_dict[dialog_act] = prev_words

                    prev_words = []
                    dialog_act = ""

    return dialog_dict, number_dict
```

```
72
73
74  #   3)  Generate Probability Dictionary
75  def probability(number_dict):
76      # prob_dict: key = sense , value = probability of sense
77      prob_dict = dict()
78
79      total = 0
80      for sense in number_dict:
81          total += number_dict[sense]
82      for sense in number_dict:
83          prob_dict[sense] = number_dict[sense] / total
84
85      return prob_dict
86
87
88  #   4)  Generate Unique Dictionary
89  def generate_unique_dict(dialog_dict):
90      # unique_dict: key = DialogAct, value = list of unique words
91      unique_dict = dict()
92
93      for dialog_act in dialog_dict:
94          unique_dict[dialog_act] = list()
95          for word in dialog_dict[dialog_act]:
96              if not word in unique_dict[dialog_act]:
97                  unique_dict[dialog_act].append(word)
98
99      return unique_dict
100
101
102 #   5)  Process Test Data
103 def process_test_data(filename):
104     # test_dict: key = ID of the line , value = list of all words
105     test_dict = dict()
106
107     ID = 0
108     with open(filename, 'r', encoding='utf-8') as f:
109         content = f.read()
110         lines = content.split("\n")
111         for line in lines:
112             if line.startswith("Student:"):
113                 words = line.split(" ")
114                 for word in words:
115                     if word != "Student:":
116                         if ID in test_dict:
117                             test_dict[ID].append(word)
118                         else:
119                             test_dict[ID] = [word]
120             ID += 1
121     return test_dict
122
123
124 #   6)  Add One Smoothing and Get Output
125 def add_one_smoothing(test_dict, dialog_dict, number_dict, unique_dict, prob_dict,
    output):
126
127     # score_dict: key = ID , value = sense and score
128     score_dict = dict()
129     # final_dict: key = ID , value = final label
130     final_dict = dict()
131
132     for ID in test_dict:
133         for sense in dialog_dict:
134             total = 0
135             for word in test_dict[ID]:
136                 # number of word appearance in sense
137                 num_1 = dialog_dict[sense].count(word) + 1
138                 # number of sense appearance
139                 num_2 = number_dict[sense] + len(unique_dict[sense])
140
141                 total *= math.log((num_1 / num_2), 2)
142
```

```
143                 score = math.log(prob_dict[sense], 2) + total
144
145                 if ID not in score_dict:
146                     score_dict[ID] = list()
147                 score_dict[ID].append({sense:score})
148
149
150         # iterates a dictionary to find argmax and best sense for ID
151         final_label = ""
152         argMax = -999
153         for ID in score_dict:
154             for scores in score_dict[ID]:
155                 for sense in scores:
156                     score = scores[sense]
157                     if score > argMax:
158                         argMax = score
159                         final_label = sense
160             final_dict[ID] = final_label
161
162         for ID in final_dict:
163             for word in test_dict[ID]:
164                 output.write(word+" ")
165             output.write("\n"+"Label: "+str(final_dict[ID])+"\n\n")
166
167         return final_dict
168
169
170     #   7)  Calculate The Accuracy
171     def cal_acc(final_dict, testfile):
172
173         # test_out_dict: key = ID , value = test data
174         test_out_dict = dict()
175
176         with open(testfile, encoding='utf-8' ) as f:
177             line_num = 0
178             content = f.read()
179             lines = content.split("\n")
180             for line in lines:
181                 if len(line) > 2:
182                     if line.startswith("Advisor:"):
183                         dialog_act = middle_content("[", "]", line)
184                     if line.startswith("Student:"):
185                         if len(dialog_act) > 0:
186                             test_out_dict[line_num] = dialog_act
187                 line_num += 1
188
189
190         correct_num = 0
191         total_num = 0
192         for word in final_dict:
193             if word in final_dict and word in test_out_dict:
194                 if final_dict[word] == test_out_dict[word]:
195                     correct_num += 1
196                 total_num += 1
197         accuracy = correct_num/total_num
198         print("Accuracy:", accuracy)
199
200         return accuracy
201
202
203
204
205
206     if __name__ == '__main__':
207
208         # 1) load file
209         trainFile = sys.argv[1]
210         testFile = sys.argv[2]
211
212         # 2) create outputfile
213         outputName = testFile + ".out"
214         output = open(outputName, "w")
```

```
215
216         # 3)  Parse Training File
217         dialog_dict, number_dict = parse_file(trainFile)
218
219         # 4)  Generate Probability Dictionary
220         prob_dict = probability(number_dict)
221
222         # 5)  Generate Unique Dictionary
223         unique_dict = generate_unique_dict(dialog_dict)
224
225         # 6)  Process Test Data
226         test_dict = process_test_data(testFile)
227
228         # 7)  Add One Smoothing and Get Output
229         final_dict = add_one_smoothing(test_dict, dialog_dict, number_dict, unique_dict,
            prob_dict, output)
230
231         # 8)  Calculate The Accuracy
232         cal_acc(final_dict, testFile)
233
```