

```
1 '''
2 ***** PLEASE READ *****
3
4 Script that reads in spam and ham messages and converts each training example
5 into a feature vector
6
7 Code intended for UC Berkeley course CS 189/289A: Machine Learning
8
9 Requirements:
10 -scipy ('pip install scipy')
11
12 To add your own features, create a function that takes in the raw text and
13 word frequency dictionary and outputs a int or float. Then add your feature
14 in the function 'def generate_feature_vector'
15
16 The output of your file will be a .mat file. The data will be accessible
17 using
18 the following keys:
19     -'training_data'
20     -'training_labels'
21     -'test_data'
22
23 Please direct any bugs to kevintee@berkeley.edu
24 '''
25 from collections import defaultdict
26 import glob
27 import re
28 import scipy.io
29 import numpy as np
30
31 from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
32
33 from tqdm import tqdm
34
35 NUM_TRAINING_EXAMPLES = 5172
36 NUM_TEST_EXAMPLES = 5857
37
38 BASE_DIR = 'data/input/'
39 SPAM_DIR = 'spam/'
40 HAM_DIR = 'ham/'
41 TEST_DIR = 'test/'
42
43 # ***** Features *****
44
45 # Features that look for certain words
46
47
48 def freq_pain_feature(text, freq):
49     return float(freq['pain'])
50
51
52 def freq_private_feature(text, freq):
53     return float(freq['private'])
54
55
56 def freq_bank_feature(text, freq):
57     return float(freq['bank'])
58
59
```

```
60 def freq_money_feature(text, freq):
61     return float(freq['money'])
62
63
64 def freq_drug_feature(text, freq):
65     return float(freq['drug'])
66
67
68 def freq_spam_feature(text, freq):
69     return float(freq['spam'])
70
71
72 def freq_prescription_feature(text, freq):
73     return float(freq['prescription'])
74
75
76 def freq_creative_feature(text, freq):
77     return float(freq['creative'])
78
79
80 def freq_height_feature(text, freq):
81     return float(freq['height'])
82
83
84 def freq_featured_feature(text, freq):
85     return float(freq['featured'])
86
87
88 def freq_differ_feature(text, freq):
89     return float(freq['differ'])
90
91
92 def freq_width_feature(text, freq):
93     return float(freq['width'])
94
95
96 def freq_other_feature(text, freq):
97     return float(freq['other'])
98
99
100 def freq_energy_feature(text, freq):
101     return float(freq['energy'])
102
103
104 def freq_business_feature(text, freq):
105     return float(freq['business'])
106
107
108 def freq_message_feature(text, freq):
109     return float(freq['message'])
110
111
112 def freq_volumes_feature(text, freq):
113     return float(freq['volumes'])
114
115
116 def freq_revision_feature(text, freq):
117     return float(freq['revision'])
118
119
```

```
120 def freq_path_feature(text, freq):
121     return float(freq['path'])
122
123
124 def freq_meter_feature(text, freq):
125     return float(freq['meter'])
126
127
128 def freq_memo_feature(text, freq):
129     return float(freq['memo'])
130
131
132 def freq_planning_feature(text, freq):
133     return float(freq['planning'])
134
135
136 def freq_pleased_feature(text, freq):
137     return float(freq['pleased'])
138
139
140 def freq_record_feature(text, freq):
141     return float(freq['record'])
142
143
144 def freq_out_feature(text, freq):
145     return float(freq['out'])
146
147 # Features that look for certain characters
148
149
150 def freq_semicolon_feature(text, freq):
151     return text.count(';')
152
153
154 def freq_dollar_feature(text, freq):
155     return text.count('$')
156
157
158 def freq_sharp_feature(text, freq):
159     return text.count('#')
160
161
162 def freq_exclamation_feature(text, freq):
163     return text.count('!')
164
165
166 def freq_para_feature(text, freq):
167     return text.count('(')
168
169
170 def freq_bracket_feature(text, freq):
171     return text.count('[')
172
173
174 def freq_and_feature(text, freq):
175     return text.count('&')
176
177 # ----- Add your own feature methods -----
178
179
```

```
180 def example_feature(text, freq):
181     return int('example' in text)
182
183 # Generates a feature vector
184
185
186 def generate_feature_vector(text, freq):
187     feature = []
188     feature.append(freq_pain_feature(text, freq))
189     feature.append(freq_private_feature(text, freq))
190     feature.append(freq_bank_feature(text, freq))
191     feature.append(freq_money_feature(text, freq))
192     feature.append(freq_drug_feature(text, freq))
193     feature.append(freq_spam_feature(text, freq))
194     feature.append(freq_prescription_feature(text, freq))
195     feature.append(freq_creative_feature(text, freq))
196     feature.append(freq_height_feature(text, freq))
197     feature.append(freq_featured_feature(text, freq))
198     feature.append(freq_differ_feature(text, freq))
199     feature.append(freq_width_feature(text, freq))
200     feature.append(freq_other_feature(text, freq))
201     feature.append(freq_energy_feature(text, freq))
202     feature.append(freq_business_feature(text, freq))
203     feature.append(freq_message_feature(text, freq))
204     feature.append(freq_volumes_feature(text, freq))
205     feature.append(freq_revision_feature(text, freq))
206     feature.append(freq_path_feature(text, freq))
207     feature.append(freq_meter_feature(text, freq))
208     feature.append(freq_memo_feature(text, freq))
209     feature.append(freq_planning_feature(text, freq))
210     feature.append(freq_pleased_feature(text, freq))
211     feature.append(freq_record_feature(text, freq))
212     feature.append(freq_out_feature(text, freq))
213     feature.append(freq_semicolon_feature(text, freq))
214     feature.append(freq_dollar_feature(text, freq))
215     feature.append(freq_sharp_feature(text, freq))
216     feature.append(freq_exclamation_feature(text, freq))
217     feature.append(freq_para_feature(text, freq))
218     feature.append(freq_bracket_feature(text, freq))
219     feature.append(freq_and_feature(text, freq))
220
221     # ----- Add your own features here - -----
222     # Make sure type is int or float
223
224     return feature
225
226 # This method generates a design matrix with a list of filenames
227 # Each file is a single training example
228
229
230 def generate_design_matrix(filenames, vectorizer=None):
231     texts = []
232     for filename in tqdm(filenames):
233         with open(filename, 'r', encoding='utf-8', errors='ignore') as f:
234             try:
235                 text = f.read()
236             except Exception as e:
237                 continue
238             text = text.replace('\r\n', ' ') # Remove newline character
239             texts.append(text)
```

```
240
241     if not vectorizer:
242         vectorizer = TfidfVectorizer(stop_words='english', max_features=2000)
243         vectorizer.fit(texts)
244
245     design_matrix = vectorizer.transform(texts)
246
247     # design_matrix = []
248     # for text in texts:
249     #     words = re.findall(r'\w+', text)
250     #     word_freq = defaultdict(int) # Frequency of all words
251     #     for word in words:
252     #         word_freq[word] += 1
253
254     #     # Create a feature vector
255     #     feature_vector = vectorizer.transform(text)
256     #     design_matrix.append(feature_vector)
257
258     return design_matrix.todense(), vectorizer
259
260 # ***** Script starts here *****
261 # DO NOT MODIFY ANYTHING BELOW
262
263
264 spam_filenames = glob.glob(BASE_DIR + SPAM_DIR + '*.txt')
265 ham_filenames = glob.glob(BASE_DIR + HAM_DIR + '*.txt')
266 test_filenames = [BASE_DIR + TEST_DIR + str(x) + '.txt' for x in
267                   range(NUM_TEST_EXAMPLES)]
268 n_spam = len(spam_filenames)
269 n_ham = len(ham_filenames)
270 n_test = len(test_filenames)
271 all_filenames = spam_filenames + ham_filenames + test_filenames
272 all_design_matrix, vectorizer = generate_design_matrix(all_filenames)
273 # Important: the test_filenames must be in numerical order as that is the
274 # order we will be evaluating your classifier
275
276 X = all_design_matrix[: (n_spam + n_ham)]
277 Y = np.array([1] * n_spam + [0] * n_ham).reshape((-1, 1))
278 test_design_matrix = all_design_matrix[(n_spam + n_ham):]
279
280 print(f'X:{X.shape}')
281 print(f'X:{test_design_matrix.shape}')
282
283 file_dict = {}
284 file_dict['training_data'] = X
285 file_dict['training_labels'] = Y
286 file_dict['test_data'] = test_design_matrix
287 scipy.io.savemat(BASE_DIR + 'spam_data.mat', file_dict)
```