

In [8]:

```

1  # Para instalar python_speech_features utilice en la consola de Python:
2  # pip install python_speech_features
3
4  # Para instalar el paquete de reconocimiento de voz
5  # pip3 install hmmlearn
6
7  # EJECUTAR EL CÓDIGO
8  # Este programa se debe ejecutar en la consola de Python
9  # Para ello utilice el siguiente comando:
10
11 # python reconocer_texto.py --input-folder data
12
13 # NOTA: El programa: reconocer_texto.py se incluye en la carpeta
14 # NOTA: Al final de este documento se incluye una imagen de ejecución
15
16 import os
17 import argparse
18 import warnings
19 import numpy as np
20 from scipy.io import wavfile
21 from hmmlearn import hmm
22 from python_speech_features import mfcc
23
24 # Define una función para analizar los argumentos de entrada
25 def build_arg_parser():
26     parser = argparse.ArgumentParser(description='Trains the HMM-based speech
27         recognition system')
28     parser.add_argument("--input-folder", dest="input_folder", required=True,
29         help="Input folder containing the audio files for training")
30     return parser
31
32 # Define una clase para entrenar el HMM
33 class ModelHMM(object):
34     def __init__(self, num_components=4, num_iter=1000):
35         self.n_components = num_components
36         self.n_iter = num_iter
37         self.cov_type = 'diag'
38         self.model_name = 'GaussianHMM'
39         self.models = []
40         self.model = hmm.GaussianHMM(n_components=self.n_components,
41             covariance_type=self.cov_type, n_iter=self.n_iter)
42
43     # 'training_data' es un array numpy 2D donde cada fila es 13-dimensional
44     def train(self, training_data):
45         np.seterr(all='ignore')
46         cur_model = self.model.fit(training_data)
47         self.models.append(cur_model)
48     # corre el modelo HMM para realizar inferencia sobre la entrada de datos
49     def compute_score(self, input_data):
50         return self.model.score(input_data)
51
52 # Define una función para construir un modelo para cada palabra
53 def build_models(input_folder):
54
55     # Inicializar la variable para almacenar todos los modelos
56     speech_models = []

```

```

57 # Analiza el directorio de entrada
58 for dirname in os.listdir(input_folder):
59     # Obtiene el nombre del subfolder
60     subfolder = os.path.join(input_folder, dirname)
61     if not os.path.isdir(subfolder):
62         continue
63
64     # Extrae la etiqueta
65     label = subfolder[subfolder.rfind('/') + 1:]
66     # Inicializa las variables
67     X = np.array([])
68     # Crea una lista de archivos a ser utilizados para el entrenamiento
69     # Se deja un archivo por folder para validación
70     training_files = [x for x in os.listdir(subfolder) if x.endswith('.v
71
72     # Se itera a través de los archivos de entrenamiento y se construyen
73     for filename in training_files:
74         # Se extrae el path actual
75         filepath = os.path.join(subfolder, filename)
76         # Se lee la señal lde audio desde el archivo de entrada
77         sampling_freq, signal = wavfile.read(filepath)
78
79         # Se extraen las características MFCC
80         with warnings.catch_warnings():
81             warnings.simplefilter('ignore')
82             features_mfcc = mfcc(signal, sampling_freq)
83         # Se agrega a la variable X
84         if len(X) == 0:
85             X = features_mfcc
86         else:
87             X = np.append(X, features_mfcc, axis=0)
88
89         # Se crea el modelo HMM
90         model = ModelHMM()
91         # Se entrena el HMM
92         model.train(X)
93         # Se almacena el modelo para la palabra actual
94         speech_models.append((model, label))
95         # Se reinicia la variable
96         model = None
97     return speech_models
98
99 # Define a function to run tests on input files
100 def run_tests(test_files):
101     # Classify input data
102     for test_file in test_files:
103         # Read input file
104         sampling_freq, signal = wavfile.read(test_file)
105         # Extract MFCC features
106         with warnings.catch_warnings():
107             warnings.simplefilter('ignore')
108             features_mfcc = mfcc(signal, sampling_freq)
109         # Define variables
110         max_score = -float('inf')
111         output_label = None
112         # Run the current feature vector through all the HMM
113         # models and pick the one with the highest score

```

```

114         for item in speech_models:
115             model, label = item
116             score = model.compute_score(features_mfcc)
117             if score > max_score:
118                 max_score = score
119                 predicted_label = label
120             # Print the predicted output
121             start_index = test_file.find('/') + 1
122             end_index = test_file.rfind('/')
123             original_label = test_file[start_index:end_index]
124             print('\nOriginal: ', original_label)
125             print('Predicted:', predicted_label)
126
127 if __name__ == '__main__':
128     args = build_arg_parser().parse_args()
129     input_folder = args.input_folder
130
131     # Build an HMM model for each word
132     speech_models = build_models(input_folder)
133     # Test files -- the 15th file in each subfolder
134     test_files = []
135     for root, dirs, files in os.walk(input_folder):
136         for filename in (x for x in files if '15' in x):
137             filepath = os.path.join(root, filename)
138             test_files.append(filepath)
139
140     run_tests(test_files)

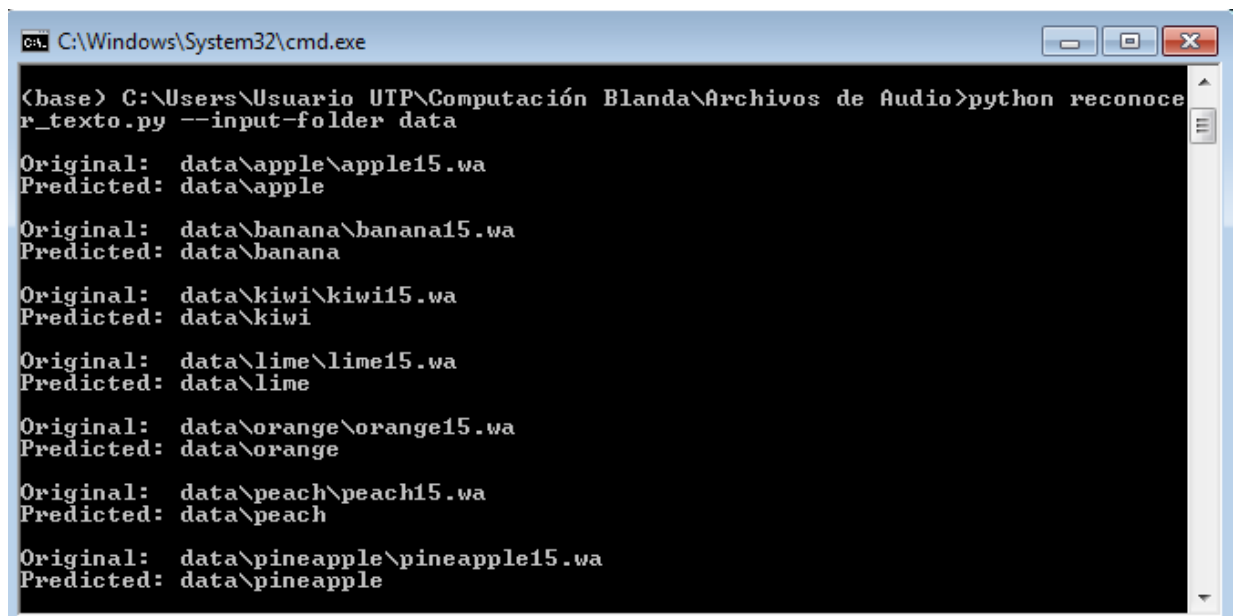
```

usage: ipykernel_launcher.py [-h] --input-folder INPUT_FOLDER

ipykernel_launcher.py: error: the following arguments are required: --input-folder

An exception has occurred, use %tb to see the full traceback.

SystemExit: 2



The screenshot shows a Windows command prompt window titled "C:\Windows\System32\cmd.exe". The user has run the command: `python reconoce_r_texto.py --input-folder data`. The output shows the script successfully recognizing words from audio files. For each file, it prints the "Original" label (the full path to the 15th file in a subfolder) and the "Predicted" label (the word itself).

```

Original: data\apple\apple15.wa
Predicted: data\apple

Original: data\banana\banana15.wa
Predicted: data\banana

Original: data\kiwi\kiwi15.wa
Predicted: data\kiwi

Original: data\lime\lime15.wa
Predicted: data\lime

Original: data\orange\orange15.wa
Predicted: data\orange

Original: data\peach\peach15.wa
Predicted: data\peach

Original: data\pineapple\pineapple15.wa
Predicted: data\pineapple

```