

# 3 Data Preprocessing

## Data Science - Capstone Project Submission

- Student Name: **James Toop**
- Student Pace: **Self Paced**
- Scheduled project review date/time: **29th October 2021 @ 21:30 BST**
- Instructor name: **Jeff Herman / James Irving**
- Blog URL: <https://toopster.github.io/> (<https://toopster.github.io/>)

### IMPORTANT NOTE:

This section presents code and instructions for preprocessing each dataset for training the models.

The datasets and transformed JSON files have not been included in the GitHub repository with this notebook and will need to be downloaded and stored in the local repository for the code to run correctly.

The code in the [notebook \(2\\_data\\_acquisition.ipynb\)](#) entitled `2_data_acquisition.ipynb` contains code for downloading the datasets.

To ensure ease of use, however, it is also possible to download the raw and transformed datasets using [this link](https://drive.google.com/file/d/11KIYIzWtEQJ-pp0G1bJPHXLJLj8uKPqW/view?usp=sharing) (<https://drive.google.com/file/d/11KIYIzWtEQJ-pp0G1bJPHXLJLj8uKPqW/view?usp=sharing>).

In [1]:

```
1 # Import required libraries and modules for data preprocessing
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns
6
7 from pydub import AudioSegment
8 import wave
9 import soundfile as sf
10 import librosa, librosa.display
11 import IPython.display as ipd
12
13 import tensorflow as tf
14 from tensorflow.keras.layers.experimental import preprocessing
15 from tensorflow.keras import layers
16 from tensorflow.keras import models
17
18 import os
19 import pathlib
20 from pathlib import Path
21 import shutil
22 import collections
23 import nltk
24
25 import shared_functions.preprocessing as preprocess
```

### 3.1 Stage 1 - Transforming the Ultrasuite dataset

## 3.1 Stage 1 - Transforming the Ultrasuite dataset

### 3.1.1 Isolating utterances from the original audio samples and labelling

Unlike the Speech Commands dataset, each audio sample in it's raw format contains multiple utterances that are spoken by both the Speech Therapist and the child subject.

The labels for each `.wav` file have been stored in a separate `.lab` file together with timestamps for the start and end of each utterance. The following is an example for the audio clip previewed earlier:

```
54700000 60900000 CAR
88800000 94800000 GIRL
109500000 112999999 MOON
126100000 136400000 KNIFE
```

In order to get the audio samples from the Ultrasuite datasets into the appropriate format for the Deep Learning models, we will need to splice the raw audio samples according to these timestamps.

In [2]:

```
1 def ultrasuite_word_labels(src_dataset, src_file):
2     '''
3     Extracts the labels from a single *.lab file into a single DataFrame
4
5     Params:
6         src_dataset (str): Dataset name of the target *.lab file
7         src_file (str): Filename of the target *.lab file
8
9     Returns:
10        word_labels_df (pandas.core.frame.DataFrame):
11        DataFrame containing the labels (utterances), timestamps, speaker
12        and session from the *.lab file
13    '''
14    filepath = 'data/ultrasuite/labels-uxtd-uxssd-upx/'
15    filepath = filepath + src_dataset + '/word_labels/lab/' + src_file
16
17    columns = ['start_time', 'end_time', 'utterance']
18    word_labels_df = pd.DataFrame()
19    word_labels_df = pd.read_csv(filepath,
20                                sep=" ",
21                                header=None,
22                                names=columns)
23
24    # Extract the speaker, session and speech data from the filename
25    word_labels_df['dataset'] = src_dataset
26    word_labels_df['speaker'] = src_file[0:3]
27    if len(src_file[4:-9]) == 0:
28        word_labels_df['session'] = None
29    else:
30        word_labels_df['session'] = src_file[4:-9]
31    word_labels_df['speech_waveform'] = src_file[-8:-4]
32
33    # Tidy up data formatting and correct time based units
34    word_labels_df['utterance'] = word_labels_df['utterance'].str.lower()
35    word_labels_df['start_time'] = pd.to_timedelta(word_labels_df['start_time']
36                                                    * 100)
37    word_labels_df['end_time'] = pd.to_timedelta(word_labels_df['end_time']
38                                                  * 100)
39
40    return word_labels_df
```

In [3]:

```
1 # Quick test to check function works for a single labels file
2 upx_01F_df = ultrasuite_word_labels('upx', '01F-BL1-005A.lab')
3 upx_01F_df.head()
```

Out[3]:

	start_time	end_time	utterance	dataset	speaker	session	speech_waveform
0	00:00:00	00:00:00.620000	teeth	upx	01F	BL1	005A
1	00:00:03.860000	00:00:04.650000	watch	upx	01F	BL1	005A
2	00:00:06.160000	00:00:06.780000	orange	upx	01F	BL1	005A
3	00:00:09.050000	00:00:09.980000	school	upx	01F	BL1	005A

In [4]:

```
1 def extract_segments(y, sr, segments, dataset):
2     '''
3     Extracts audio segments from the source *.wav file based on timestamps
4     contained within the associated *.lab file
5
6     Params:
7         y (str): Path to input file
8         sr (int): Sample Rate
9         segments (DataFrame): DataFrame containing timestamps, labels,
10                             speaker and session data
11         dataset (str): Specific ultrasuite dataset to process can be
12                       'upx', 'uxtd' or 'uxssd'
13     '''
14     # Compute segment regions in number of samples
15     starts = np.floor(segments.start_time.dt.total_seconds() * sr).astype(int)
16     ends = np.ceil(segments.end_time.dt.total_seconds() * sr).astype(int)
17
18     isolated_directory = 'data/ultrasuite_isolated/' + dataset + '/'
19
20     if not os.path.isdir(isolated_directory):
21         os.makedirs(isolated_directory.strip('/'))
22
23     i = 0
24     # Slice the audio into segments
25     for start, end in zip(starts, ends):
26         audio_seg = y[start:end]
27         print('extracting audio segment:', len(audio_seg), 'samples')
28
29         # Set the file path for the spliced audio file
30         file_path = isolated_directory + str(segments.speaker[i]) + '/'
31         if segments.session[i] != None:
32             file_path = file_path + str(segments.session[i]) + '/'
33         file_path = file_path + str(segments.speech_waveform[i]) + '/'
34
35         if not os.path.isdir(file_path):
36             os.makedirs(file_path.strip('/'))
37
38         file_name = file_path + str(segments.utterance[i]) + '.wav'
39
40         sf.write(file_name, audio_seg, sr)
41         i += 1
```

In [5]:

```
1 def process_ultrasuite_wav_files(src_dataset, src_speaker, src_session):
2     '''
3     Processes and extracts audio segments for all Ultrasuite *.wav files
4
5     Params:
6         src_dataset (str): Ultrasuite dataset to process can be
7                             'upx', 'uxtd' or 'uxssd'
8         src_speaker (str): Speaker to process
9         src_session (str): Session to process
10
11     '''
12     directory = 'data/ultrasuite/core-'
13     directory = directory + src_dataset + '/core/' + src_speaker + '/'
14
15     # Set the target directory based on session if available
16     if src_session != False:
17         directory = directory + src_session + '/'
18
19     # Loop through files in directory, splice and rename based on labels
20     for filename in os.listdir(directory):
21
22         if not filename[-5:-4] == 'E' or filename[-5:-4] == 'D':
23             # Fetch the corresponding word labels and load into a DataFrame
24             # Handle errors for when no labels exist
25             # Labels only available for high quality samples
26             try:
27                 if src_session != False:
28                     labels_filename = src_speaker + '-' + src_session + '-'
29                     labels_filename = labels_filename + filename[-8:-4]
30                 else:
31                     labels_filename = src_speaker + '-' + filename[-8:-4]
32
33                 labels_filename = labels_filename + '.lab'
34
35                 labels_df = ultrasuite_word_labels(src_dataset,
36                                                    labels_filename)
37
38                 wav_path = directory + filename
39                 y, sr = librosa.load(wav_path, sr=16000)
40                 extract_segments(y,
41                                sr,
42                                labels_df,
43                                src_dataset)
44
45             except IOError:
46                 if src_session != False:
47                     print('\n',
48                           src_speaker,
49                           '-',
50                           src_session,
51                           '-',
52                           filename[-8:-4],
53                           '.lab not found \n')
54                 else:
55                     print('\n',
56                           src_speaker,
57                           '-',
58                           filename[-8:-4],
59                           '.lab not found \n')
```

In [6]:

```
1 def process_all_wav_files(datasets):
2     '''
3     Processes and extracts audio segments for all Ultrasuite *.wav files
4
5     Params:
6         datasets (list): Ultrasuite dataset to process can be any or all
7                           of 'upx', 'uxtd', 'uxssd'
8     '''
9     # Loop through the datasets
10    for dataset in datasets:
11        current_dataset_dir = 'data/ultrasuite/core-' + dataset + '/core/'
12        speakers = os.listdir(current_dataset_dir)
13
14        # Loop through the speakers
15        for speaker in speakers:
16            current_speaker_dir = 'data/ultrasuite/core-' + dataset + '/core/'
17            current_speaker_dir = current_speaker_dir + speaker + '/'
18            sessions = os.listdir(current_speaker_dir)
19
20            # If multiple therapy sessions loop through and process files
21            for session in sessions:
22                if os.path.isdir(os.path.join(current_speaker_dir, session)):
23                    process_ultrasuite_wav_files(dataset, speaker, session)
24                else:
25                    process_ultrasuite_wav_files(dataset, speaker, False)
```

In [ ]:

```
1 # Splice all *.wav files for all datasets
2 # NOTE: This takes a long time to run
3 process_datasets = ['upx', 'uxssd', 'uxtd']
4 process_all_wav_files(process_datasets)
```

### 3.1.2 Standardising the audio samples and folder structure

In [12]:

```
1 def pad_silence(target_length, input_filepath, output_filepath):
2     '''
3     Pad the spliced audio samples with silence so that they are all at least
4     1 second in length
5
6     Params:
7         target_length (int): Target length of final audio sample in
8                             milliseconds
9         input_filepath (str): File path to input / original *.wav file
10        output_filepath (str): File path to output / padded *.wav file
11    '''
12    target_length = target_length
13    audio = AudioSegment.from_wav(input_filepath)
14    if len(audio) > target_length:
15        print(str(input_filepath) ,
16              'is longer than 1 second, no padding required.')
17        silence = AudioSegment.silent(duration=0)
18    else:
19        silence = AudioSegment.silent(duration=target_length - len(audio) + 1)
20
21    padded = audio + silence
22    padded.export(output_filepath, format='wav')
```

In [13]:

```
1 def standardise_filing(datasets):
2     '''
3     Standardise filing structure for isolated samples, padding and renaming
4     files in the process
5
6     Params:
7         datasets (list): Ultrasuite dataset to process can be any or all
8                          of 'upx', 'uxtd', 'uxssd'
9     '''
10    # Loop through the datasets
11    for dataset in datasets:
12
13        isolated_files = Path.cwd() / 'data/ultrasuite_isolated' / dataset
14
15        for isolated_file in isolated_files.glob('**/*'):
16
17            if isolated_file.is_file():
18
19                # Rename the file but don't lose the original references
20                filename = isolated_file.stem
21                extension = isolated_file.suffix
22                sourcedata = dataset
23                sourcefile = isolated_file.parent.parts[-1]
24
25
26                # Handle the different folder structures
27                if dataset == 'uxtd':
28                    speaker = isolated_file.parent.parts[-2]
29                    new_filename = f'{filename}_{dataset}-{speaker}-{sourcefile}'
30
31                else:
32                    session = isolated_file.parent.parts[-2]
33                    speaker = isolated_file.parent.parts[-3]
34                    new_filename = f'{filename}_{dataset}-{speaker}-{session}-{sourcefile}'
35
36                # Define the new file path creating directory if it doesn't exist
37                new_path = Path.cwd() / 'data/ultrasuite_transformed' / filename
38
39                if not new_path.exists():
40                    new_path.mkdir(parents=True, exist_ok=True)
41
42                new_file_path = new_path.joinpath(new_filename)
43
44                # Pad audio sample if required and move to new location
45                if extension == '.wav':
46                    pad_silence(1000, str(isolated_file), str(new_file_path))
```

In [ ]:

```
1 # Run the function to standardise the filing for all Ultrasuite datasets
2 standardise_filing(['upx', 'uxssd', 'uxtd'])
```

### 3.1.3 Cleansing the dataset

1. Only keep audio samples of actual words using NLTK WordNet as a source corpus
2. Remove audio samples of simple phonetic letters (from the manual remove list)



3. Only keep audio samples that have more than 5 different samples

In [15]:

```
1  # Check to see if the WordNet corpus is available, download if not and import
2  try:
3      nltk.data.find('corpora/wordnet')
4  except LookupError:
5      nltk.download('wordnet')
6
7  from nltk.corpus import wordnet as wn
8
9  def remove_invalid_samples():
10     '''
11     Function to remove all 'invalid' audio samples based on predetermined
12     criteria
13     '''
14     transformed_files = 'data/ultrasuite_transformed/'
15
16     manual_remove = ['a']
17
18     for name in sorted(os.listdir(transformed_files)):
19
20         path = os.path.join(transformed_files, name)
21
22         if os.path.isdir(path):
23             num_samples = len(os.listdir(path))
24
25             # Remove audio samples of words not listed in NLTK WordNet corpus
26             if not wn.synsets(name) or len(name)==1:
27                 print(name,
28                       'is NOT a valid word, removing',
29                       num_samples,
30                       'samples')
31                 shutil.rmtree(path)
32             # Remove audio samples where there are 5 or less samples
33             elif num_samples <= 5:
34                 print(name,
35                       'does NOT have enough samples, removing',
36                       num_samples,
37                       'samples')
38                 shutil.rmtree(path)
39             # Remove audio samples based on our manually constructed list above
40             elif name in manual_remove:
41                 print(name,
42                       'is being manually removed',
43                       num_samples,
44                       'samples')
45                 shutil.rmtree(path)
46             else:
47                 print('---')
48                 print(name,
49                       'is a valid word and there are',
50                       num_samples,
51                       'samples:\n')
```

In [ ]:

```
1 # Remove invalid audio samples from the transformed dataset
2 remove_invalid_samples()
```

### 3.1.4 Subsetting the dataset

In [17]:

```
1 # Get the audio sample file information for the Ultrasuite dataset
2 ultrasuite_filestats = preprocess.get_filestats('data/ultrasuite_transformed')
3 ultrasuite_filestats.head()
```

Out[17]:

	sample_utterance	sample_filename	sample_duration	sample_samplerate
0	parch	parch_upx-05M-BL2-017A.wav	1.000938	16000
1	parch	parch_upx-05M-Mid-016A.wav	1.001000	16000
2	parch	parch_upx-05M-BL3-016A.wav	1.000875	16000
3	parch	parch_upx-05M-BL4-016A.wav	1.000875	16000
4	parch	parch_upx-05M-Maint-016A.wav	1.000875	16000

In [18]:

```
1 # Summarise the number of samples for each utterance
2 us_summary = (ultrasuite_filestats.groupby(['sample_utterance'])
3             .size()
4             .reset_index(name='count')
5             .sort_values('count', ascending=False))
6 us_summary.head(35)
```

Out[18]:

	sample_utterance	count
366	helicopter	292
700	say	290
961	watch	235
249	elephant	233
322	got	229
705	scissors	222
946	umbrella	222
274	fishing	222
814	spider	217
397	in	211
314	gloves	210
892	thank	204
84	bridge	198
290	frog	178
958	was	171
744	sheep	166
980	yellow	163
323	gown	162
237	ear	159
538	on	154
75	boy	148
282	four	146
412	ken	143
542	or	142
704	school	142
984	zebra	141
908	times	135
505	monkey	135
906	tiger	133
548	pack	132

	sample_utterance	count
275	five	130
884	teeth	128
905	tie	123
106	cab	123
176	crab	122

In [19]:

```

1  def copy_keywords(num_keywords, keywords):
2      '''
3      Copys the 'top' keywords based on number of samples to a new folder
4
5      Params:
6          num_keywords (int): Number of 'top' keywords to copy based on
7                              number of samples available
8          keywords (DataFrame): DataFrame containing keywords sorted by
9                               number of samples
10     '''
11     src_directory = 'data/ultrasuite_transformed/'
12     top_directory = 'data/ultrasuite_top' + str(num_keywords) + '/'
13
14     sorted_keywords = keywords.reset_index()
15
16     if not os.path.isdir(top_directory):
17         os.makedirs(top_directory.strip('/'))
18
19     i = 0
20     while (i < num_keywords):
21         src_folder = src_directory + sorted_keywords.sample_utterance[i]
22         dest_folder = top_directory + sorted_keywords.sample_utterance[i]
23
24         if not os.path.isdir(dest_folder):
25             shutil.copytree(src_folder, dest_folder)
26
27             print(sorted_keywords.sample_utterance[i], 'copied')
28         else:
29             print(sorted_keywords.sample_utterance[i], 'already exists')
30         i += 1

```

In [20]:

```
1 copy_keywords(35, us_summary)
```

```
helicopter copied  
say copied  
watch copied  
elephant copied  
got copied  
scissors copied  
umbrella copied  
fishing copied  
spider copied  
in copied  
gloves copied  
thank copied  
bridge copied  
frog copied  
was copied  
sheep copied  
yellow copied  
gown copied  
ear copied  
on copied  
boy copied  
four copied  
ken copied  
or copied  
school copied  
zebra copied  
times copied  
monkey copied  
tiger copied  
pack copied  
five copied  
teeth copied  
tie copied  
cab copied  
crab copied
```

## 3.2 Stage 2 - Feature extraction

In [21]:

```
1  # Function to extract features to use in the models and store in JSON file
2  def preprocess_dataset(dataset_path,
3                        json_path,
4                        feature,
5                        num_samples,
6                        num_mfcc=13,
7                        n_fft=2048,
8                        hop_length=512):
9      '''
10     Code adapted from Deep Learning Audio Application from Design
11     to Deployment - Valerio Velardo - The Sound of AI
12
13     Extract Mel Spectrograms and MFCCs to use in the models and store in JSON
14     file
15
16     Params:
17         dataset_path (str):
18             Path to dataset containing audio samples
19         feature (str):
20             Specific feature requested, accepts either 'MFCCs' or 'mel_specs'
21         json_path (str):
22             Output path to JSON file
23         num_samples (int):
24         num_mfcc (int):
25         n_fft (int):
26         hop_length (int):
27     '''
28     # Dictionary to temporarily store mapping, labels, MFCCs and filenames
29     if feature == 'mel_specs':
30         data = {
31             'mapping': [],
32             'labels': [],
33             'mel_specs': [],
34             'files': []
35         }
36     else:
37         data = {
38             'mapping': [],
39             'labels': [],
40             'MFCCs': [],
41             'files': []
42         }
43
44     # Loop through all sub directories
45     for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
46
47         # Ensure we're at sub-folder level
48         if dirpath is not dataset_path:
49
50             # Save label in the mapping
51             label = dirpath.split('/')[-1]
52             data['mapping'].append(label)
53             print("\nProcessing: {}".format(label))
54
55             # Process all audio files in the sub directory and store features
56             for f in filenames:
57                 file_path = os.path.join(dirpath, f)
58
59                 # Load audio file and slice it to ensure length consistency
```

```

60     signal, sample_rate = librosa.load(file_path)
61
62     # Drop audio files with less than pre-decided number of samples
63     if len(signal) >= num_samples:
64
65         # Ensure consistency of the length of the signal
66         signal = signal[:num_samples]
67
68         # Extract MFCCs
69         if feature == 'mel_specs':
70             mel_specs = librosa.feature.melspectrogram(signal,
71                                                         sample_rate,
72                                                         n_fft=n_fft,
73                                                         hop_length=hop_length)
74
75             data['mel_specs'].append(mel_specs.T.tolist())
76
77         else:
78             MFCCs = librosa.feature.mfcc(signal,
79                                         sample_rate,
80                                         n_mfcc=num_mfcc,
81                                         n_fft=n_fft,
82                                         hop_length=hop_length)
83
84             data['MFCCs'].append(MFCCs.T.tolist())
85
86         # Append data in dictionary
87         data['labels'].append(i-1)
88         data['files'].append(file_path)
89         print("{}: {}".format(file_path, i-1))
90
91     # Save data in JSON file for re-using later
92     with open(json_path, 'w') as file_path:
93         json.dump(data, file_path, indent=4)

```

In [22]:

```

1  # Set the parameters for the Speech Commands dataset for preprocessing
2  sc_dataset_path = 'data/speech_commands_v0.02'
3  sc_json_path = 'speech_commands_data.json'
4  num_samples = 22050

```

In [ ]:

```

1  # Preprocess the Speech Commands dataset extracting MFCCs
2  preprocess_dataset(sc_dataset_path, sc_json_path, 'MFCCs', num_samples)

```

In [24]:

```

1  # Set the parameters for the Ultrasuite dataset for preprocessing
2  us_dataset_path = 'data/ultrasuite_top35'
3  us_json_path = 'ultrasuite_top35_data.json'
4  num_samples = 22050

```

In [ ]:

```
1 # Preprocess the Ultrasuite dataset extracting MFCCs
2 preprocess_dataset(us_dataset_path,
3                   us_json_path,
4                   'MFCCs',
5                   num_samples)
```

In [25]:

```
1 # Set the parameters for the Ultrasuite dataset for preprocessing
2 # based on Mel Spectrograms
3 us_dataset_path = 'data/ultrasuite_top35'
4 us_melspec_json_path = 'ultrasuite_top35_data_melspec.json'
5 num_samples = 22050
```

In [ ]:

```
1 # Preprocess the Ultrasuite dataset extract Mel Spectrograms
2 preprocess_dataset(us_dataset_path,
3                   us_melspec_json_path,
4                   'mel_specs',
5                   num_samples)
```

---

#### Sources / Code adapted from:

\* [Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow](https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/) - Aurélien Géron

(<https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>).

\* [Deep Learning Audio Application from Design to Deployment](https://github.com/musikalkemist/Deep-Learning-Audio-Application-From-Design-to-Deployment) - Valerio Velardo - The Sound of AI

(<https://github.com/musikalkemist/Deep-Learning-Audio-Application-From-Design-to-Deployment>).