1 Project Overview and Exploratory Data Analysis ¶

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/)](https://toopster.github.io/)

1.1 Business Case, Project Purpose and Approach

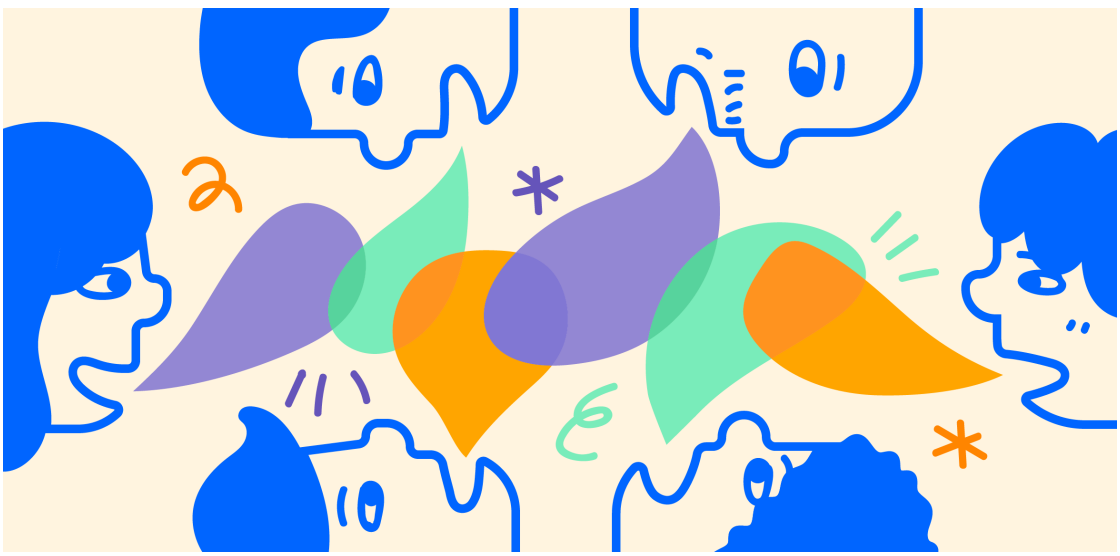
1.1.1 The importance of communication for people with severe learning disabilities

Communication is vital in ensuring that people can express themselves, control their environment and make sense of the world around them.

This is equally if not more important when that person has a learning disability and may not be able to interpret their environment as easily as others.

Communicating with children and young people who have a severe learning disability can be extremely challenging, if a child is not able to communicate by traditional methods they may become frustrated and use behaviours that might be seen as challenging.

For example, in self-harm behaviour, the child may be wanting to communicate that they are in pain or discomfort or that they are hungry, thirsty etc. If the behaviour is effective in getting what the child needs it is more likely to be repeated.



1.1.2 Types of communication

There are various forms and stages of communication but, broadly speaking, communication can be split into two types:

Expressive Language

Expressive language is the use of words to form sentences in order to communicate with other people.

Difficulties in using expressive language to communicate can range from experiencing difficulties putting words in the right order to being unable to form words in a meaningful way that others can understand.

When someone is unable to make use of expressive language, it can lead to frustration at not being able to express their needs and difficulty interacting with other people.

Receptive Language

Receptive language is the understanding of expressive language. The use of receptive language is not dependent on being able to use expressive language. Some people may not be able to form words and sentences themselves, but are able to understand expressive language when it is used by others.

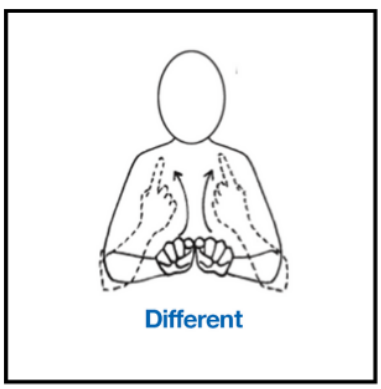
This can range from being able to easily understand what others say, to being able to only understand key words and phrases, and then only when they are spoken clearly and slowly. Everyone is different; some people may be able to use both receptive and expressive language to different degrees, whilst others may be able to use one or neither.

1.1.3 Communication techniques for people with learning disabilities

Some people with learning disabilities have difficulties communicating with others when solely making use of expressive and receptive language.

There are a variety of other techniques which have been developed to help support people for whom speech is difficult, for example:

- Communication systems such as Makaton (https://makaton.org/) (based on British sign language) or PECS (https://pecs-unitedkingdom.com/pecs/) (Picture Exchange communication system)
- Easy read symbols
- Speech and language therapy



Example of Makaton signing

Example of PECS symbols

People with learning disabilities often interpret body language and non-verbal communication in understanding simple everyday interactions.

It is essential when communicating with someone with a learning disability to give them time to take in what is being said, and to communicate more slowly than you may normally in order to allow them to process what it is that you are communicating.

It is often hard to know what support is available to help people with learning disabilities communicate more easily.

Using visual guides or cues to aide communication is one important way of supporting people to have a greater understanding of what is being conveyed to them.

1.1.4 Project purpose

> **Supporting communication in children with non-typical speech using Deep Learning**

Meet Martha...

My daughter Martha was born with a rare genetic syndrome, called Jacobsen's Syndrome, as a result of her missing over 14 million pieces of genetic material on her eleventh chromosome.

In addition to a large ventricular septal defect, a duplex kidney and a bleeding disorder, Martha has global development delay which has significantly affected her progress towards reaching milestones associated with typical child development such as walking and talking.

The purpose of this project is to use Data Science and deep learning techniques to build a model that can suitably classify audio samples from an individual with a speech disorder or with a severe learning disability that impacts speech.

The eventual aim being a tool or app that helps parents of children like Martha with a speech disorder or severe learning disability support their child's communication.

With adequately supported communication, their child will be better able to:

- Express their needs, feelings or opinions
- Make choices
- Interact with others and form relationships.
- Develop skills which will help them access learning opportunities.

Parents would be able to use the app, installed on a mobile device, to record their child speaking, receiving suggestions as to the word that the child is trying to voice, similar to the Google Recorder app (https://recorder.withgoogle.com/) but specifically configured to pick up and trained on atypical speech.

Beyond this, the app would also present an opportunity for parents to add to the dataset by submitting anonymised, labelled audio samples of their child speaking.

1.1.5 Approach

Classification using big data struggles to cope with the individual uniqueness of disabled people, and while

developers tend to design for the majority, so ignoring outliers, designing for edge cases would be a more inclusive approach.

Whilst similar to Project Euphonia by Google AI (https://sites.research.google/euphonia/about/) which is focused on "on helping people with atypical speech be better understood" and providing more equitable access to voice activated technology for users with a speech disability, this project will focus more on understanding and interpreting single words rather than whole sentences for the simple reason that most of the affected children will not be able to speak in full sentences and will have a limited vocabulary. It also also specifically aimed at parents or carers rather than the child themselves.

We will be using two datasets for the purpose of this project. (More detailed information can be found in the exploratory data analysis (2_eda.ipynb) section.

- Speech Commands (https://arxiv.org/abs/1804.03209): A dataset for limited-vocabulary speech recognition
- Ultrasuite (https://ultrasuite.github.io/): A collection of ultrasound and acoustic speech data from child speech therapy sessions

The initial plan was to use both the Speech Commands dataset combined with the Ultrasuite dataset to train each model but, after some testing and additional research, it was decided to use the Speech Commands dataset to create a "control" model against which the model based on the Ultrasuite data could be compared. This decision was principally so as not to introduce a "typical speech bias" into the final model.

Sources:

* Foundation for People with Learning Disabilities (https://www.learningdisabilities.org.uk/)
* National Health Service (NHS) UK (https://www.nhs.uk/)
* Me First: Helping health and care professionals communicate more effectively with children and young people (https://www.mefirst.org.uk/)
* Project Euphonia (by Google AI) (https://sites.research.google/euphonia/about/)
* Google AI Blog - Project Euphonia's Personalized Speech Recognition for Non-Standard Speech (https://ai.googleblog.com/2019/08/project-euphonias-personalized-speech.html)

1.2 Exploratory Data Analysis

This section presents an initial step to investigate, understand and document the available data fields and relationships, highlighting any potential issues / shortcomings within the datasets supplied.

IMPORTANT NOTE:

- Code and instructions for downloading each dataset are contained within the data acquisition notebook (2_data_acquisition.ipynb) entitled `2_data_acquisition.ipynb`.
- Code and instructions for processing the datasets is contained within the data preprocessing notebook (3_preprocessing.ipynb) entitled `3_preprocessing.ipynb`

The code cells below assume that the data has already been downloaded and processed in order to run successfully.

In [1]:

```
# Import required libraries and modules for eda
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import librosa, librosa.display
import IPython.display as ipd
import os
import pathlib
from pathlib import Path

import shared_functions.preprocessing as preprocess
```

In [2]:

```
def plot_spectrogram(Y, sr, hop_length, y_axis='linear'):
    '''
    Plot and visualise the spectrogram

        Params:
            Y (ndarray): Spectrogram to display
            sr (int): Sample Rate
            hop_length (int): Hop Length
            y_axis (str): Range for the y-axis
    '''
    # Portrait - presentation format
#     fig = plt.figure(figsize=(10, 8))
    # Landscape - Jupyter notebook format
    fig = plt.figure(figsize=(25, 10))
    librosa.display.specshow(Y,
                             sr=sr,
                             hop_length=hop_length,
                             x_axis='time',
                             y_axis=y_axis)
    fig.gca().set_xlabel('Time (seconds)', fontsize=18, labelpad=20)
    fig.gca().set_ylabel('Frequency (Hz)', fontsize=18, labelpad=20)
    plt.colorbar(format='%+2.f')
```

1.2.1 Speech Commands dataset

A dataset for limited-vocabulary speech recognition by Pete Warden, TensorFlow team at Google.

https://arxiv.org/abs/1804.03209 (https://arxiv.org/abs/1804.03209)

The Speech Commands dataset is an attempt to build a standard training and evaluation dataset for a class of simple speech recognition tasks. Its primary goal is to provide a way to build and test small models that detect when a single word is spoken, from a set of ten or fewer target words, with as few false background noise or unrelated speech.

Overview

The Speech Commands dataset contains `105,829` audio samples of `35` spoken keywords in `.wav` format, no longer than 1 second and with a sample rate of `16000`. The dataset has been organised such that each audio sample is stored within a folder that has been named so that is corresponds with it's associated

label as per the following example:

```
└── speech_commands_v0.02
    ├── backward
    │   ├── 0a2b400e_nohash_0.wav
    │   ├── 0a2b400e_nohash_1.wav
    │   ├── 0a2b400e_nohash_2.wav
    │   ├── 0a2b400e_nohash_3.wav
    │   ├── 0a396ff2_nohash_0.wav
    │   └── ...
    ├── bed
    │   ├── 0a7c2a8d_nohash_0.wav
    │   └── ...
```

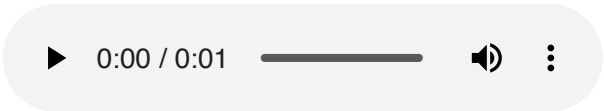
Preview audio samples and waveforms from Speech Commands dataset

In [3]:

```
# Load the audio sample and preview
target_sample = 'data/speech_commands_v0.02/zero/0a2b400e_nohash_0.wav'
sc_sample, sr = librosa.load(target_sample)
print('Audio sample: Zero')
ipd.Audio(sc_sample, rate=sr)
```

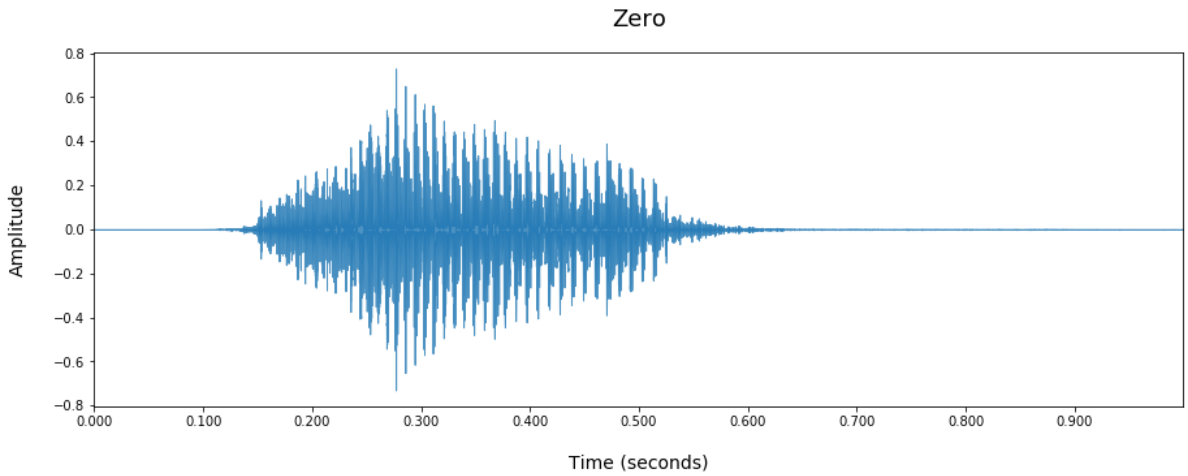
```
Audio sample: Zero
```

Out[3]:



In [4]:

```
# Plot the waveform for the specific audio sample
plt.figure(figsize=(15, 5))
plt.title('Zero', fontsize=18, pad=20)
librosa.display.waveplot(sc_sample, sr, alpha=0.8)
plt.xlabel('Time (seconds)', fontsize=14, labelpad=20)
plt.ylabel('Amplitude', fontsize=14, labelpad=20)
plt.show();
```



In [5]:

```
# Extract the short time Fourier transform and preview the array shape
hop_length = 512
n_fft = 2048

S_sc_sample = librosa.stft(sc_sample, n_fft=n_fft, hop_length=hop_length)
S_sc_sample.shape
```

Out[5]:

```
(1025, 44)
```

In [6]:

```
type(S_sc_sample[0][0])
```

Out[6]:

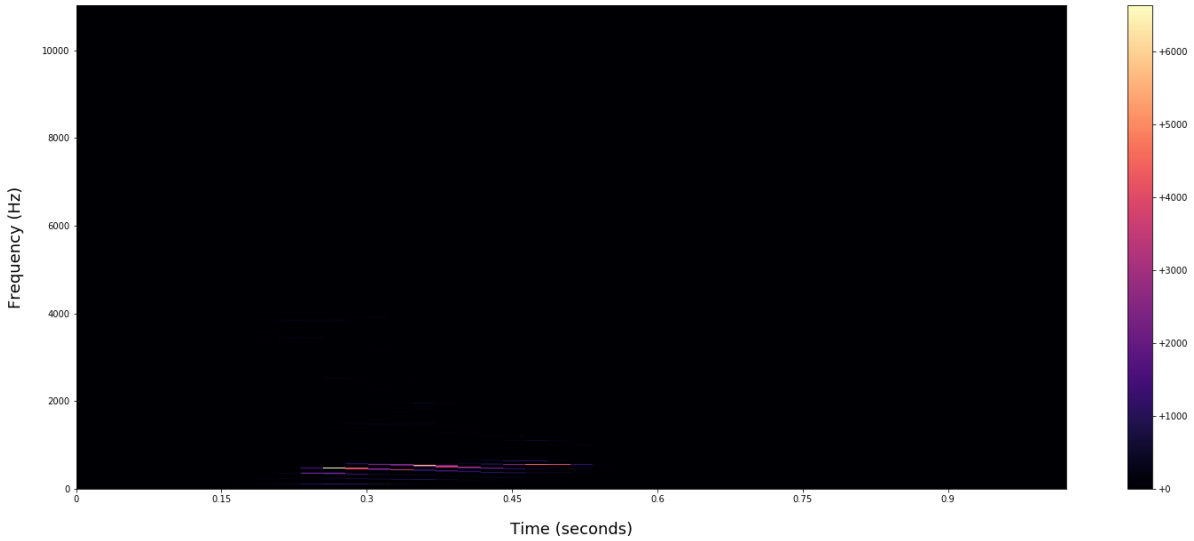
```
numpy.complex64
```

In [7]:

```
# Calculating the spectrogram
Y_sc_sample = np.abs(S_sc_sample) ** 2
```

In [8]:

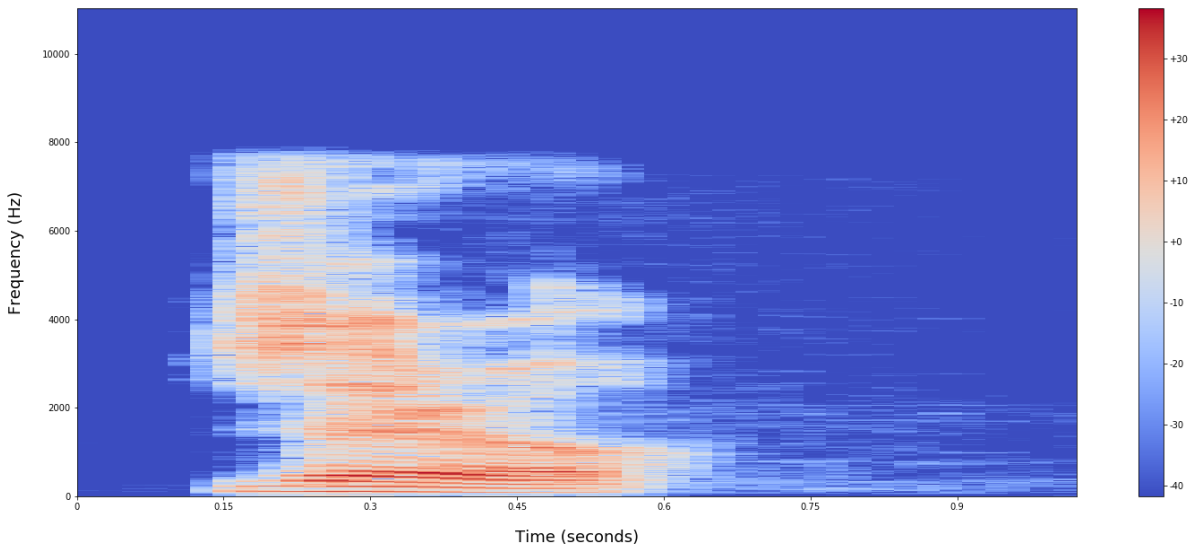
```
# Plotting the spectrogram for our example from the Speech Commands dataset
plot_spectrogram(Y_sc_sample, sr, hop_length)
```



The human perception of sound intensity is logarithmic in nature so we are interested in the log amplitude.

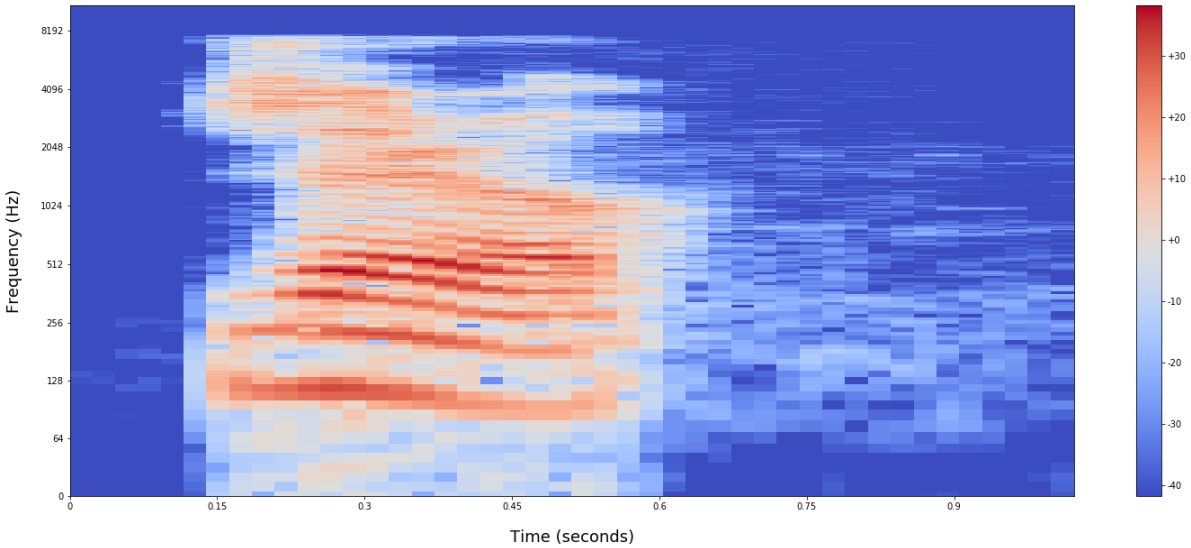
In [9]:

```
# Display spectrogram using log amplitude
Y_log_sc_sample = librosa.power_to_db(Y_sc_sample)
plot_spectrogram(Y_log_sc_sample, sr, hop_length)
```



In [10]:

```
# Display spectrogram using log frequency
plot_spectrogram(Y_log_sc_sample, sr, hop_length, y_axis='log')
```



In [11]:

```
# Get the audio sample file information for the Speech Commands dataset
speechcommands_filestats = preprocess.get_filestats('data/speech_commands_v0.02
speechcommands_filestats.head()
```

Out[11]:

	sample_speaker	sample_filename	sample_duration	sample_samplerate
0	right	8e523821_nohash_2.wav	1.000000	16000
1	right	bb05582b_nohash_3.wav	1.000000	16000
2	right	988e2f9a_nohash_0.wav	1.000000	16000
3	right	a69b9b3e_nohash_0.wav	0.938625	16000
4	right	1eddce1d_nohash_3.wav	1.000000	16000

In [12]:

```
len(speechcommands_filestats)
```

Out[12]:

```
105829
```

In [13]:

```
# Check how many samples are longer than 1 second in duration
sc_long_samples = speechcommands_filestats[
    (speechcommands_filestats['sample_duration'] > 1)
]
len(sc_long_samples)
```

Out[13]:

```
0
```

1.2.2 Ultrasuite dataset

A collection of ultrasound and acoustic speech data from child speech therapy sessions – University of Edinburgh, School of Infomatics

https://ultrasuite.github.io/ (https://ultrasuite.github.io/)

Ultrasuite is a collection of ultrasound and acoustic speech data from child speech therapy sessions. The current release includes three datasets, one from typically developing children and two from speech disordered children:

- **Ultrax Typically Developing (UXTD) (https://ultrasuite.github.io/data/uxtd/)** - A dataset of 58 typically developing children.
- **Ultrax Speech Sound Disorders (UXSSD) (https://ultrasuite.github.io/data/uxssd/)** - A dataset of 8 children with speech sound disorders.
- **UltraPhonix (UPX) (https://ultrasuite.github.io/data/upx/)** - A second dataset of children with speech sound disorders, collected from 20 children.

Source:
Eshky, A., Ribeiro, M. S., Cleland, J., Richmond, K., Roxburgh, Z., Scobbie, J., & Wrench, A. (2018) Ultrasuite: A repository of ultrasound and acoustic data from child speech therapy sessions. Proceedings of INTERSPEECH. Hyderabad, India. [paper (https://ultrasuite.github.io/papers/ultrasuite_IS18.pdf)]

Overview

In its raw format (i.e. as downloaded), the Ultrasuite dataset contains `14,286` audio samples from `86` different children from multiple speech therapy sessions in `.wav` format with a sample rate of `22050`. The audio samples are of varying lengths containing spoken words from both the child and, in some cases, the speech therapist.

Each audio sample is given a "grading", A - E that denotes the decreasing quality of the audio sample and capture. As a result, labels are not typically supplied for grades D and E.

In order to prepare the datasets prior to training the deep learning models, we need to download, splice and label each of the audio samples from the Ultrasuite datasets. We also need to transform them into the same structure as the Speech Commands dataset.

Once spliced and cleansed, as detailed below, the transformed Ultrasuite dataset contains `33,800` audio samples with a sample rate of `16000` lasting at least 1 second with `991` different words spoken.

Currently each dataset is organised by a child subject ID, session, and then sample ID, for example:

```
└── ultrasuite
    ├── core-uxssd
        ├── core
            ├── 01M  (Child ID)
                ├── BL1  (Session ID)
                    ├── 001A.wav (Sample ID)
                    ├── 002A.wav
                    ├── 003A.wav
                    ├── 004A.wav
                    ├── 005A.wav
                    └── ...
                ├── Maint1
                    ├── 001A.wav
                    ├── 002A.wav
                    ├── 003A.wav
                    └── ...
```

Preview audio samples, waveforms, spectrograms and labels from Ultrasuite dataset

In [14]:

```
# Load the Ultrasuite raw file information into a dataframe
raw_ultrasuite_filestats = preprocess.get_filestats('data/ultrasuite')
raw_ultrasuite_filestats.head()
```

Out[14]:

	sample_speaker	sample_filename	sample_duration	sample_samplerate
0	39F	037D.wav	21.733878	22050
1	39F	021D.wav	3.622313	22050
2	39F	017D.wav	2.414875	22050
3	39F	040D.wav	9.659501	22050
4	39F	041D.wav	3.018594	22050

In [15]:

```
# Preview the number of audio samples in the raw Ultrasuite dataset
len(raw_ultrasuite_filestats)
```

Out[15]:

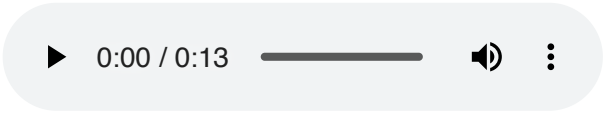
```
14286
```

In [16]:

```
# Load the audio sample and preview
target_sample = 'data/ultrasuite/core-uxssd/core/06M/BL1/002A.wav'
us_sample, sr = librosa.load(target_sample)
print('Audio sample: Car | Girl | Moon | Knife')
ipd.Audio(us_sample, rate=sr)
```

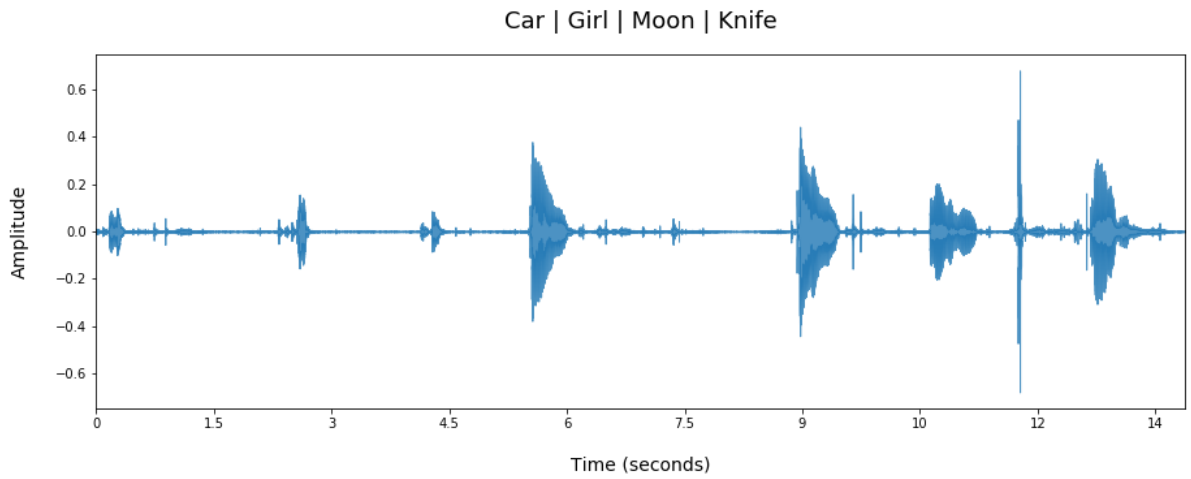
```
Audio sample: Car | Girl | Moon | Knife
```

Out[16]:



In [17]:

```
# Plot the waveform for the specific audio sample
plt.figure(figsize=(15, 5))
plt.title('Car | Girl | Moon | Knife', fontsize=18, pad=20)
librosa.display.waveplot(us_sample, sr, alpha=0.8)
plt.xlabel('Time (seconds)', fontsize=14, labelpad=20)
plt.ylabel('Amplitude', fontsize=14, labelpad=20)
plt.show();
```



In [18]:

```
def all_ultrasuite_word_labels(src_directory, src_dataset):
    '''
    Extracts and combines the labels from all *.lab files into a single
    DataFrame

        Params:
            src_directory (str): Target directory containing the *.wav files
                                 for generating stats on

        Returns:
            all_labels_df (pandas.core.frame.DataFrame):
            DataFrame containing all labels from the Ultrasuite dataset
    '''
    directory = src_directory + src_dataset + '/word_labels/lab/'
    columns = ['start_time', 'end_time', 'utterance']
    all_labels_df = pd.DataFrame()

    for filename in os.listdir(directory):

        filepath = directory + filename

        labels_df = pd.read_csv(filepath, sep=" ", header=None, names=columns)

        # Extract the speaker, session and speech data from the filename and
        # add to the dataframe
        labels_df['dataset'] = src_dataset
        labels_df['speaker'] = filename[0:3]
        if len(filename[4:-9]) == 0:
            labels_df['session'] = None
        else:
            labels_df['session'] = filename[4:-9]
        labels_df['speech_waveform'] = filename[-8:-4]

        # Tidy up data formatting and correct time based units
        labels_df['utterance'] = labels_df['utterance'].str.lower()
        labels_df['start_time'] = pd.to_timedelta(labels_df['start_time'] * 100)
        labels_df['end_time'] = pd.to_timedelta(labels_df['end_time'] * 100)

        # Append incoming labels to existing dataframe
        all_labels_df = all_labels_df.append(labels_df, ignore_index=True)

    return all_labels_df
```

In [19]:

```
# Load the labels for the Ultrax Speech Sound Disorders dataset
uxssd_df = all_ultrasuite_word_labels('data/ultrasuite/labels-uxtd-uxssd-upx/',
                                      'uxssd')
```

In [20]:

```
# Preview the data
uxssd_df.head()
```

Out[20]:

	start_time	end_time	utterance	dataset	speaker	session	speech_waveform
0	00:00:01.340000	00:00:02.040000	th	uxssd	02M	BL1	069B
1	00:00:02.460000	00:00:03.350000	atha	uxssd	02M	BL1	069B
2	00:00:03.790000	00:00:04.650000	eethee	uxssd	02M	BL1	069B
3	00:00:05.210000	00:00:06.110000	otho	uxssd	02M	BL1	069B
4	00:00:00.970000	00:00:01.480000	core	uxssd	04M	Maint1	017A

In [21]:

```
# Load the labels for the Ultrax Typically Developing dataset
uxtd_df = all_ultrasuite_word_labels('data/ultrasuite/labels-uxtd-uxssd-upx/',
                                     'uxtd')
```

In [22]:

```
# Preview the data
uxtd_df.head()
```

Out[22]:

	start_time	end_time	utterance	dataset	speaker	session	speech_waveform
0	00:00:07.300000	00:00:08.180000	watch	uxtd	37M	None	001A
1	00:00:08.270000	00:00:09.219999	fishing	uxtd	37M	None	001A
2	00:00:09.539999	00:00:10.500000	gloves	uxtd	37M	None	001A
3	00:00:10.640000	00:00:11.520000	spider	uxtd	37M	None	001A
4	00:00:01.170000	00:00:02.020000	r	uxtd	30F	None	010B

In [23]:

```
# Preview the dataframe info
uxtd_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6094 entries, 0 to 6093
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype          
---  ------           --------------  -----          
 0   start_time       6094 non-null   timedelta64[ns]
 1   end_time         6094 non-null   timedelta64[ns]
 2   utterance        6094 non-null   object         
 3   dataset          6094 non-null   object         
 4   speaker          6094 non-null   object         
 5   session          0 non-null      object         
 6   speech_waveform  6094 non-null   object         
dtypes: object(5), timedelta64[ns](2)
memory usage: 333.4+ KB
```

In [24]:

```
# Load the labels for the Ultraphonix dataset
upx_df = all_ultrasuite_word_labels('data/ultrasuite/labels-uxtd-uxssd-upx/',
                                    'upx')
```

In [25]:

```
# Preview the data
upx_df.head()
```

Out[25]:

	start_time	end_time	utterance	dataset	speaker	session	speech_waveform
0	00:00:00	00:00:01.400000	sigh	upx	20M	Post	012A
1	00:00:01.639999	00:00:02.910000	sausages	upx	20M	Post	012A
2	00:00:03.140000	00:00:03.970000	snail	upx	20M	Post	012A
3	00:00:04.890000	00:00:05.699999	beige	upx	20M	Post	012A
4	00:00:00.510000	00:00:01.240000	sack	upx	16M	BL3	016A

Post transformation

In [26]:

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

Out[26]:

	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 [27]:

```
# Get the file information the audio samples for 'book'
ultrasuite_book = ultrasuite_filestats[
    (ultrasuite_filestats['sample_utterance'] == 'book')
]
len(ultrasuite_book)
```

Out[27]:

```
114
```

In [28]:

```
# Preview the dataframe
ultrasuite_book.head(20)
```

Out[28]:

	sample_utterance	sample_filename	sample_duration	sample_samplerate
22389	book	book_uxssd-07F-Post-015A.wav	1.001000	16000
22390	book	book_uxtd-13F-045A.wav	1.000938	16000
22391	book	book_uxssd-06M-Mid-011A.wav	1.000938	16000
22392	book	book_uxssd-02M-Maint2-013A.wav	1.000938	16000
22393	book	book_uxssd-06M-Post-047A.wav	1.001000	16000
22394	book	book_uxssd-05M-Post-020A.wav	1.000938	16000
22395	book	book_upx-08M-Suit-014A.wav	1.000875	16000
22396	book	book_uxtd-16F-046A.wav	1.000938	16000
22397	book	book_upx-03F-Therapy_04-011A.wav	1.000938	16000
22398	book	book_uxssd-02M-BL2-027A.wav	1.001000	16000
22399	book	book_uxssd-04M-Mid-013A.wav	1.000875	16000
22400	book	book_uxssd-04M-BL2-013A.wav	1.001000	16000
22401	book	book_upx-15M-Post-052A.wav	1.000875	16000
22402	book	book_uxtd-23F-046A.wav	1.000938	16000
22403	book	book_uxssd-03F-BL1-027A.wav	1.000875	16000
22404	book	book_uxssd-07F-BL2-014A.wav	1.000938	16000
22405	book	book_uxssd-07F-Mid-014A.wav	1.001000	16000
22406	book	book_uxtd-25M-046A.wav	1.000938	16000
22407	book	book_upx-07M-Suit-015A.wav	1.000875	16000
22408	book	book_uxssd-06M-Maint1-041A.wav	1.000938	16000

In [29]:

```
# Check the total number of samples after preprocessing
len(ultrasuite_filestats)
```

Out[29]:

```
33800
```

In [30]:

```
# Check how many samples that are longer than 1 second in duration
us_long_samples = ultrasuite_filestats[
    (ultrasuite_filestats['sample_duration'] > 1.0)
]
len(us_long_samples)
```

Out[30]:

33800

In [31]:

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

Out[31]:

	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 [32]:

```
1 len(us_summary)
```

Out[32]:

991

In [33]:

```
# Get the top 35 words with the largest number of samples
us_top35 = us_summary.head(35)
us_top35.sort_values('sample_utterance', ascending=True)
```

Out[33]:

	sample_utterance	count
75	boy	148
84	bridge	198
106	cab	123
176	crab	122
237	ear	159
249	elephant	233
274	fishing	222
275	five	130
282	four	146
290	frog	178
314	gloves	210
322	got	229
323	gown	162
366	helicopter	292
397	in	211
412	ken	143
505	monkey	135
538	on	154
542	or	142
548	pack	132
700	say	290
704	school	142
705	scissors	222
744	sheep	166
814	spider	217
884	teeth	128
892	thank	204
905	tie	123
906	tiger	133
908	times	135
946	umbrella	222
958	was	171

	sample_utterance	count
961	watch	235
980	yellow	163
984	zebra	141

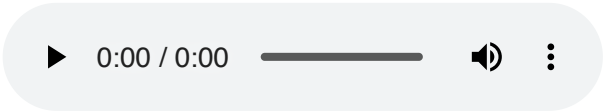
Preview audio samples, waveforms, spectrograms and labels from Ultrasuite dataset post transformation

In [34]:

```
# Load the audio sample and preview post transformation
target_sample_isolated = 'data/ultrasuite_isolated/uxssd/06M/BL1/002A/girl.wav'
us_sample_isolated, sr = librosa.load(target_sample_isolated)
print('Audio sample: Girl')
ipd.Audio(us_sample_isolated, rate=sr)
```

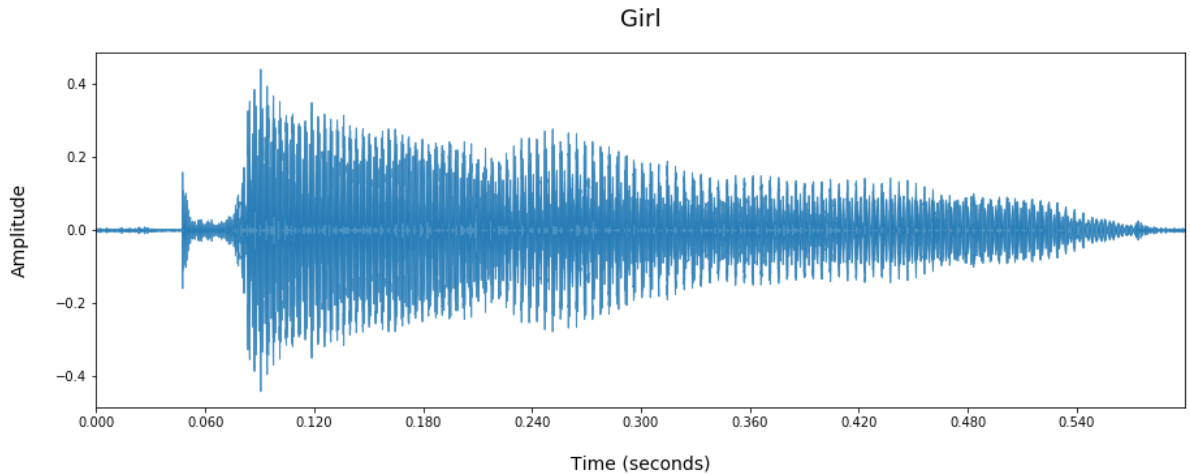
```
Audio sample: Girl
```

Out[34]:



In [35]:

```
# Plot the isolated waveform for the specific audio sample
plt.figure(figsize=(15, 5))
plt.title('Girl', fontsize=18, pad=20)
librosa.display.waveplot(us_sample_isolated, sr, alpha=0.8)
plt.xlabel('Time (seconds)', fontsize=14, labelpad=20)
plt.ylabel('Amplitude', fontsize=14, labelpad=20)
plt.show();
```

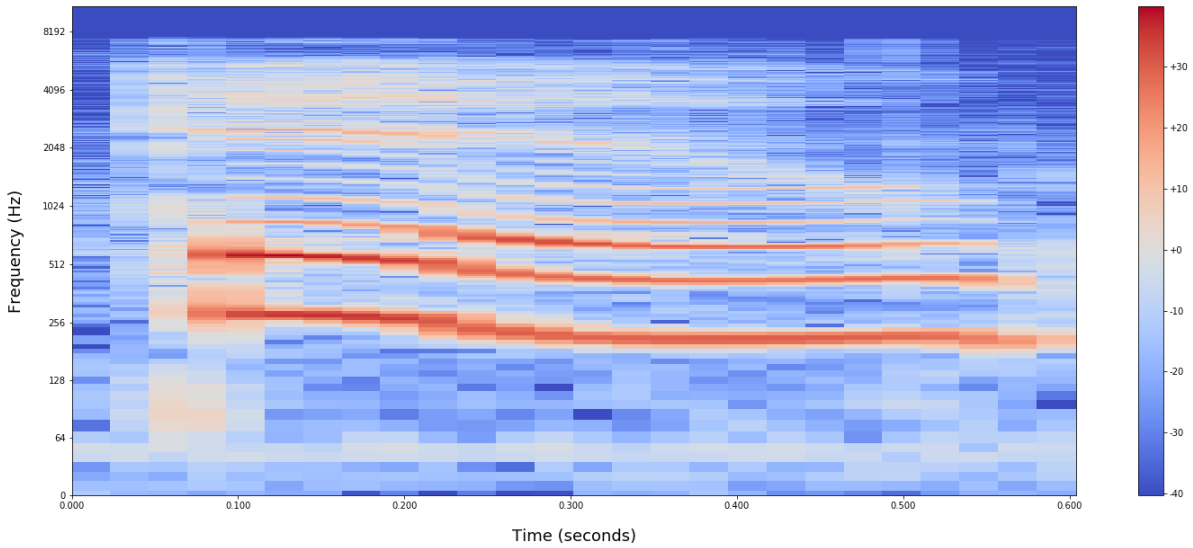


In [36]:

```
S_us_sample_iso = librosa.stft(us_sample_isolated,
                               n_fft=n_fft,
                               hop_length=hop_length)
Y_us_sample_iso = np.abs(S_us_sample_iso) ** 2

Y_log_us_sample_iso = librosa.power_to_db(Y_us_sample_iso)

# Display spectrogram using log frequency
plot_spectrogram(Y_log_us_sample_iso, sr, hop_length, y_axis='log')
```



Sources / Code adapted from:

* Audio Signal Processing for ML - Valerio Velardo - The Sound of AI (https://github.com/musikalkemist/AudioSignalProcessingForML)