
# Gridspace IAP 2023
# Logistics and Project 1

January 9, 2023

## Intro, Logistics and Timeline

Welcome to the Gridspace X IAP Education Series!

In this lecture series, we aim to cover the breadth of spoken language topics beyond a typical introduction to machine learning or NLP.

The course will involve 12 lectures over the course of the month, along with weekly projects, lecture questions and the opportunity to present your work each week. Emphasis in these projects is on creativity and application of the principles taught during the week.

### Handing in Project Work

- Project 1 is included in this handout, and we will send out the rest of the projects as the course progresses.
- **Finished Projects can be emailed to iap@gridspace.com** on the due date, along with a brief description of what you implemented and how to run it.
- For each project, we will provide brief feedback and suggestions for improvement/further development. The emphasis should be on creativity and exploration of the concepts.
- Each Monday, we will give you the chance to present your work to the rest of the students (the final week this will be the Friday rather than Monday).

- If you wish to volunteer your project for a specific week (for example if you're particularly proud of what you've made), simply let us know when you submit it. Otherwise, projects will be selected each week.
- Note that we will not be following up on missing projects. If you wish to hand in a project after the deadline for comments, that is also fine, however, you will not be able to present that week.

This course is self-motivated, and whilst it is entirely possible to find the answers online, or ask ChatGPT to code it up for you, we recommend you implement everything from scratch. Unless you choose to make that the point of the project that week.

## Timetable

| | MONDAY | TUESDAY | WEDNESDAY | THURSDAY | FRIDAY | SATURDAY | SUNDAY |
|---|---|---|---|---|---|---|---|
| Fundamentals of Sound and Signals | Jan 9<br>Sound, Acoustics and Voice | 10 | 11<br>DSP and Information Theory | 12<br>Machine Learning and the JAX Library | 13 | 14 | 15 |
| Intro to Language and NLP | 16<br>Morphology and Words | 17 | 18<br>Syntax, Semantics, Embeddings and Lexers | 19 | 20<br>Large Language Models (LLMs) & InstructGPT | 21 | 22 |
| Pillars of Speech Technology | 23<br>ASR: Automatic Speech Recognition | 24 | 25<br>TTS: Text to Speech | 26<br>DS: Dialog Systems | 27 | 28 | 29 |
| Real World Applications | 30<br>Emotion, Dialog Acts, Personality and Lying | 31 | Feb 1<br>Conversation Design and AI Ethics | 2 | 3<br>Deploying ML and NLP in the Real World | 4 | 5 |

## Pset Deadlines

- Pset 1: Friday 13th Jan, Presentations Monday 15th
- Pset 2: Friday 20th Jan, Presentations Monday 22nd
- Pset 3: Friday 27th Jan, Presentations Monday 30th
- Pset 4: Friday 3rd Feb, Presentations same day

# Data: Harper Valley Dataset

Throughout the projects, we will be asking you to manipulate sentences and audio. To do that, you will need some data. Our own engineers created a public dataset of synthetic call center conversations, played out by actors, to simulate the data our models perform on in real life. This call center is for the fictional company "Harper Valley Insurance", and as such is called the 'Harper Valley Dataset'. Throughout the following weeks, if you need audio data to test against, train on, or manipulate, you are welcome to use anything (clean) from the web, including other public datasets or legally scraped audio. However, we recommend using the Harper Valley collection, and the problems are built around the information available in this dataset.

Note how the dataset is formatted. For each data point, there are the (2 channel) waveform audio files, along with a JSON file containing transcript and other metadata, such as start time, duration, and emotion valence. In the root folder is a README telling any would-be user of the data how it is structured.

Explore the repository. There are some training scripts, and experiments that you can run. For ease of data manipulation, there are also Database classes and some utility functions. You will not need these for this week, but it is good practise to get a holistic sense of what is available.

## Load and View Harper Valley Dataset

1. Open a terminal
2. Clone the repository to your local machine (‘git clone https://github.com/cricketclub/gridspace-stanford-harper-valley‘) and navigate into the directory.
3. Run ‘pip install notebook‘ to install jupyter notebook.
4. Run ‘jupyter notebook‘ to start the notebook on your local server. Navigate to 'localhost:8888' in a browser (you can replace 8888 with whichever port your notebook uses, although this is 8888 by default)

You may find the JSON python package useful in reading the data into a dictionary, and librosa, scipy.signal or torchaudio for reading your audio files. Play around with the data, and see what information you have available to you. Reach out early if you have any difficulty accessing this.

# Week 1: Sound and Signals

**Due Friday 13th Jan, Presentations Monday 15th**

## Wake up word detection

*"Alexa? *pause* Play that song I like"*
*"Hey Google *beat* how do I hula hoop?"*
*"Oi, Siri, is it going to rain today?" *no response**

We're all familiar with these phrases by now, although it wasn't long ago that the concept of simply speaking a command, and having a household device perform the desired task, would have been the equivalent of Hogwarts magic. These incantations even have similar limitations. Think of Harry and Ron, struggling to pronounce "Wingardium Leviosa" and their mounting frustration with each item that refused to levitate, and tell me you haven't experienced the same as you yell across the room at your alexa. Whilst there are many reasons for a voice controlled device to fail to parse a command (noise, mispronunciation, out-of-scope phrases etc etc), the first point of failure is that crucial ability to recognise it's wake word, spoken amongst a deluge of daily sounds. It must be able to dutifully ignore the surrounding private conversations, whilst remaining primed for it's call to action.

So how do we get its attention?

## Exercises

Code File: Colab Notebook
Other files: Drive Folder

### Exercise 0

Go through the notebook and familiarise yourself with the functions and signal processing techniques presented. Run them on your own audio and practise loading and manipulating audio data in python.

### Exercise 1

A wake word is simply a specific signal, or waveform, that, when observed by the device and matched to the 'wake word' pattern, tells the device to start listening for the command. Choose a wake word, record yourself saying it a few times, and plot the waveforms. Notice the specific shape and form of the audio in the time domain. Plot the spectrogram and observe its frequency content over time, see if you can identify the consonants and vowels. Get a few other people to do the same and compare them. Do you see pitch differences? What features are consistent for your word, regardless of speaker? Identify some distinguishing features for your wake word. Plot the Mel-spectrogram (librosa has an in-built function for this). Do you observe the difference?

### Exercise 2

Write a function that takes in a sound of the same length as your wake word, and calculates it's similarity to your recorded waveform. You can choose a simple time domain cosine similarity, or something more complex in the frequency domain. Think about the features you want it to be able to calculate similarity over.

### Exercise 3

Write a function to take in a longer audio segment, and compare the two signals to find the similarity of the signal to your template word at each point in time. This can be a cross-correlation in the time domain, or it can be a windowed/convolved application of your previous similarity function. You should now have a new time-domain signal representing some scaled likelihood, at each point in time, that your wake word was spoken. Test this against some inputs, do you observe the spike?

### Exercise 4

Write a function that takes in audio, and performs some action in response to hearing your wake word.

### Exercise 5: Open Ended Exercise

Evaluate and make an improvement. Notice how well your program reacts to your wake word, and make some code change to improve it's performance. Is it overfitting to your single example? Is it robust to different speakers?

What about different speeds? Another common function used to calculate similarity, that is robust to stretching and squishing of signals in the time domain, is Dynamic Time Warping (DTW). Implement the same procedure but with DTW. Is it more robust?

Stretch goal: Can you make it work in real time?

### Materials Provided

- Harper Valley Audio
- Guitar Audio Files
- Signal visualisation notebook and starter code

## Readings and Resources

Bartosz Ciechanowski's interactive presentation on sound: https://ciechanow.ski/sound/