Gridspace IAP 2023
Projects 3 and 4

January 22, 2023

Week 3: Core Models

Due Friday 27th Jan, Presentations Monday 30th

Simple Speech Recognition

Week 3 focused on the core pillars of speech technology; Recognition (Automatic Speech Recognition or ASR), Synthesis (Text-To-Speech or TTS) and Understanding (Dialog Systems). One of our key products, and one of the most commercially-successful speech technologies in industry, is speech transcription. Think about everything from automatic captioning, to asking Siri to set a reminder. All of these require the ability to map an audio signal to it's corresponding word.

Unfortunately, training a fully functional ASR, with a large and diverse vocabulary, is both computationally and time intensive, and not feasible for our project. Therefore, for this Pset, we will be training a much smaller version, to recognise a small subset of words. Let's start with commands!

Exercises

Exercise 1: Speech Recognition

Our model takes in a wave form and a sample rate, and uses convolutional layers in the neural network to process the audio. You might be familiar with

convolutional layers as apply to 2D signals, such as images. The classical example being AlexNet, the CNN image recognition network. Applying CNNs to audio data works in much the same way, just in 1D.

Exercise 2: Audio Features

Explore what other audio features may improve the performance of your speech recognition model. For example, a binary indication of whether the voice was female or male, given as a feature, may allow the model to learn faster to differentiate pitch groups. (Note I say 'faster'. In theory, if allocating a weight within the network to switch weights on pitch groups is a helpful representation, then our original network (which has access to the general pitch of the voice), should be able to learn this. If our data is skewed, or our dataset is too small, giving it this type of information as a helping hand can improve performance).

Exercise 3: Open Ended Exercise

You've trained a network to recognise command words. Earlier this month you wrote a wake word detection algorithm.

Can we combine these? It's time to play! Create an app that uses both these technologies to perform functions based on wake word detection and specific commands. Spotify has a reasonable API (spotipy), although authentication can be a pain. If you need advice or help to realise or discuss the potential of an idea, reach out.

Optional Extra Exercise: Spectrograms

We talked about CNNs on audio signals being a 1D variation of image recognition networks. However, if you remember back to our DSP lecture, we also had a 2D signal representation in the form of spectrograms. That is, frequency variation over time. What happens if we modify our network to take in these 2D signals as input instead, and change our convolutional layers to work in 2D. Have we lost information? Does it perform better or worse?

Materials Provided

- Speech Recognition Training code

Week 4: Applied ML

Due Friday 3rd Feb, Presentations Same Day

Utterance trait classification (Larger overall project)

So, we can model language. But what about meaning? Emotion? Intent? Language is dense, to put it lightly, and humans have been struggling to find patterns in it since before texts even existed. There are plenty of jokes and memes about how we struggle to interpret even the most basic of sentences. "I'm OK" has been proven to have at least 34 different meanings. If you want to know what they are, ask. Or just try and say it 34 different ways, I bet you can.

So, if humans struggle to identify concrete rules, that's kind of why we use ML right? To find the patterns we're blind to, and to learn characteristics ignored or abstracted by our conscious brains. So lets throw some good old classification models at a bunch of speech utterances and see what we can learn.

The other aspect of this final project, is we will not be providing starter code. We expect you to use off the shelf models, and would be surprised if you did not use code online as a starting point. Look for classifier tutorials, BERT sentiment analysis, RNN networks. We are building a simple model that will have similarities to a lot of the tutorials online, however, using our own data and with our own goal in mind. This initial step of evaluating what currently exists, building off pre-trained models, and deciding where we need custom solutions, is a core skill in developing new ML technologies. Building a model from scratch is out of the scope of this series but worth pursuing and learning.

Exercises

Exercise 1: Data

The main limiting factor, as ever, is labelled data. In order to learn emotion, a model needs to be able to train on some data with known emotions. We are providing our Harper Valley dataset of synthetic data, labelled with a few different traits. Inspect the data using the provided code, and choose

what labels you want to build your classifier around. Of course, you are free to source your own dataset, but cite the source in your project.

Exercise 2: Data again

Having identified the trait you intend to build your classifier on, you will need to get the data into a format that a classifier can take as input. If you are using the Harper Valley dataset, the github repository provides from classes for data formatting and manipulation. Other datasets may take some more work, and feel free to reach out to us if you need help at this stage. Take pains on this step. This is the most powerful stage of the process. Your data will direct your model, and many models have achieved incredible results (NLU, Question forming, list generation) by virtue of careful data engineering.

Exercise 3: Model

Feed your model! Choose an off-the-shelf classifier (such as one from Hugging Face), or build your own if you're feeling ambitious. Initialize your model, put the data into a DataLoader, and feed that DataLoader in to the training pipeline. The model may take a while to train, however, if it seems unreasonably slow, make sure you are using a GPU (change Colab runtime as needed). You may have to edit some of your parameters to get it to train in a reasonable time.

It is often helpful to print stats at each step to be able to keep an eye on how the training process is progressing, and make sure you don't waste 4 hours when a simple loss plot could have shown you that the model was failing.

Look for a loss plot with a sharp initial decrease that stabilises out. Cancel training early if the loss does not start steadily dropping in the first few iterations. Fail early and iterate quick.

Exercise 4: Evaluation

How well does it perform? By human inspection, as well as standard industry metrics (precision, accuracy, recall). Qualitatively, if you feed it a new utterance, does it classify it correctly?

Materials Provided

- Harper Valley Dataset
- The Internet

End of IAP

Congrats! You have made it to the end of the projects. It's a lot to squeeze in to 4 weeks and, in the pursuit of breadth, some depth had to be sacrificed. We have built a wake word detector, fine tuned a language model, and trained speech recognition model and language trait classification models, but these are just touching the surface of each discipline in the landscape of NLP. We hope primarily to have sparked the curiosity to pursue more projects in this vein and will be sending out a list of resources for further exploration. If you're an MIT student, 6.864 (Advanced Natural Language Processing), is a fantastic class, as is 6.341 (Discrete Time Signal Processing) and 6.867 (Graduate Machine Learning). If you wish to discuss NLP in industry, email us at IAP@gridspace.com. We hope you had fun!

What Next?

The recent revolution in Large Language Models has changed the landscape of NLP forever, and, as a company, we are excited to be on the front line of that charge.

We will be hanging out, demoing and recruiting at MIT's XFair this February 20th, so come swing by and say hi!

Exercise To The Reader:

Can you combine all four projects? We'd love to see it!