Machine Learning Nanodegree Capstone Proposal

Vivek Krishna Choppa

December 27, 2017

1. Domain Background

Speech recognition and language understanding is the problem domain with huge number of solutions associated with it. Instead of typing your question in the search engine and reading through lot of articles to arrive at your answer, if a machine could listen to our speech and be able to understand our question to answer it appropriately, then that would be of great value.

I know of two different theories of human hearing. One is Place theory which is frequency based and the other one is Temporal theory. Two main tendencies of input for speech recognition are spectrogram and MFCC - Mel-Frequency Cepstral Coefficients.

Historically, neural networks has been used for both deducing features from the speech signal as well as predicting output probabilities of the speech. (C. P. Lim, 2000; 1)

Gaussian mixture models are relatively more efficient to make the system more speaker adaptive compared to neural networks. (Rose)

Hybrid networks generated by combining neural networks with hidden markov models are also explored.

Amazon's Alexa, Google Home and many other corporate giants are trying to tackle this problem in their own way. I would like to explore this problem domain more and see how state of the art Deep learning techniques have been used to solve this problem.

This problem is important to solve because only if the speech recognition is robust, then the system can attempt to understand language from speech.

2. Motivation

We might be on the verge of too many screens. It seems like every day, new versions of common objects are "re-invented" with built-in wifi and bright touchscreens. A promising antidote to our screen addiction are voice interfaces. My personal motivation for investigating this area more in depth is that I want to build my own speech recognition system and use this as a base for my further explorations in language understanding.

3. Problem Statement

I am using a kaggle competition, where you're challenged to use the Speech Commands Dataset to build an algorithm that understands simple spoken commands. By improving the recognition accuracy of open-sourced voice interface tools, we can improve product effectiveness and their accessibility.

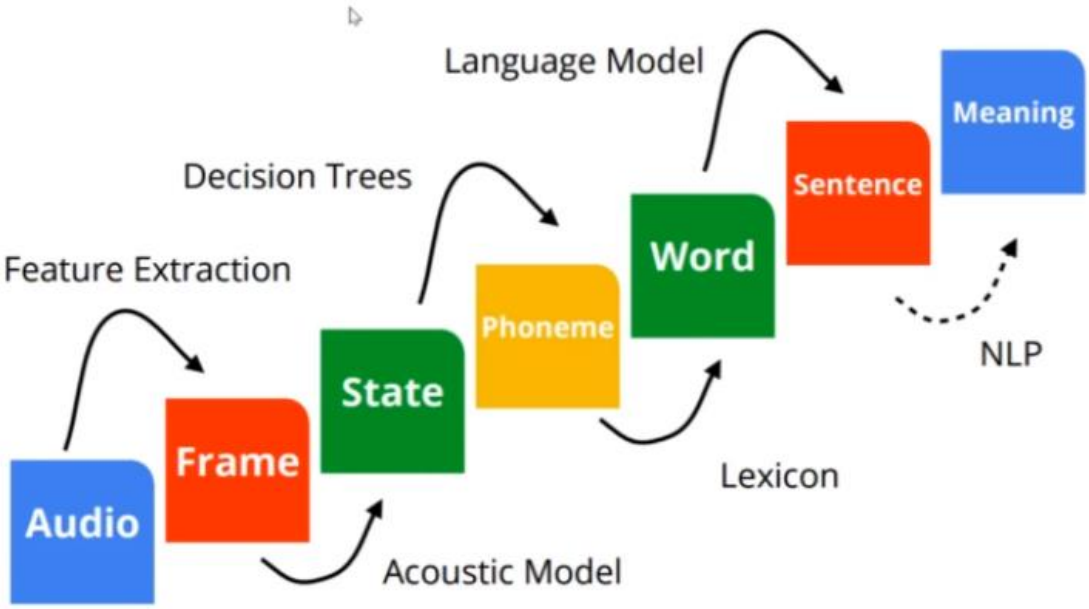
4. Datasets and Inputs

TensorFlow recently released the Speech Commands Datasets. It includes 65,000 one-second long utterances of 30 short words, by thousands of different people. Among the 30 short word categories, each word has around 1700 to 2400 examples and that should be good enough to train the model for each label.

5. Solution statement

Speech representations: We want a low-dimensionality representation, invariant of speaker, background noise, rate of speaking etc.

1. FFT (Fast Fourier transform) – Represents energy in different frequency bands. Generating an image of this graph could potentially change it from audio problem to image problem. FFT is still too high dimensional and we have to downsample by local weighted averages on mel scale non-linear spacing, and take a log.
2. MFCC (Mel frequency Cepstral Coefficients) – Discrete cosine transformation of the Mel filter bank energies. Similar to principal components.



I will be using Deep learning and Tensor-flow techniques to accurately predict speech commands from audio track snippets. Neural networks can handle high-dimensional features with correlated features.

Among different neural network architectures like fully connected model, CNNs and RNNs,

CNNs with pooling layers throws away the timing information that is quite important for speech. Pooling in the frequency domain can throw away some variation in pitch between different speakers. We want to recognize same patterns whether they occur at high frequency or low frequency,

RNNs have given much better results for speech recognition. RNNs are a powerful model for sequential data. Deep, bidirectional Long Short-term Memory RNNs with end to end training and weight noise gives state of the art results in phenome recognition. (Alex Graves)

I will train the dataset with SVM classifier to use that performance as a benchmark model for my use case. It has been tried already on a similar dataset of 11 vowels spoken in isolation and has produced comparable results to Artificial neural networks.

6. Evaluation metrics

Submissions are evaluated on Multiclass Accuracy, which is simply the average number of observations with the correct label.

Note: There are only 12 possible labels for the Test set: `yes`, `no`, `up`, `down`, `left`, `right`, `on`, `off`, `stop`, `go`, `silence`, `unknown`.

The `unknown` label should be used for a command that is not one of the first 10 labels or that is not `silence`.

This is the optimal evaluation metric because high accuracy means that the model is able to recognize words rightly. I will likely use a categorical cross-entropy loss function for error.

Submission File

For audio clip in the test set, you must predict the correct `label`. The submission file should contain a header and have the following format:

```
fname,label
clip_000044442.wav,silence
clip_0000adecb.wav,left
clip_0000d4322.wav,unknown
etc.
```

7. Project design

Programming Language and Libraries

1. Python 3
2. TensorFlow
3. Keras
4. Scikit-learn

8. Machine Learning Design

Type of training experience: Training against labelled data.

Target function: Labelled words.

Representation of Learned function: Deep neural network of RNN kind.

Learning algorithm: Back propagation

First I will clean the data (audio files) to capture only the speech signal in that that is required and trim out the rest. For this, some VOICE ACTIVITY DETECTION will be really helpful. It is impossible to cut all the files manually and do this basing on plots. But we can use for example *webrtcvad* package to have a good *VAD*.

Then, I will create features from these audio files such as FFT and MFCC.

I will experiment with few neural network architectures which might have CNN, LSTM/RNN and fully connected layers.

In LSTMs, the hidden layers are not just collection of neurons but a collection of memory cells. It learns what to remember and what to forget as part of training.

I will be using a CNN layer (helpful in reducing number of parameters that needs to be trained) after the input layer, followed by three LSTM layers and fully connected layer at the end.

I will optimize to one such network which gives optimal performance.