בס"ד

Final Report

Project topic: Spoken language identification using deep learning.

Presented by: Akiva Gubbay – ID: 319210860.

Abstract

In this paper, I propose a system for language identification in speech audio files using recurrent neural network. Utilizing Mel-frequency cepstral coefficients(MFCC) obtained from the VoxForge data set. I trained a network architecture for two and four language classifiers.

Introduction

Spoken language identification(LID) is the problem of mapping continuous speech to the language it corresponds to. Deep learning models have yielded strong improvements in the fields of spoken language identification, machine translation and other language related tasks. Applications that arise directly from language identification include:

- Customer support centers that route people to their native agents.
- Law enforcement officials pinpoint suspects to certain geographic regions.
- Web information retrieval systems.

In this paper, I use a multi-layered recurrent neural network(RNN), combined with a fully connected hidden layer, as an approach to the task of language identification in speech audio files.

Related work

- This paper describes a project that classified urban sounds into categories using machine learning:
  https://aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/

- TopCoder held a contest to identify the spoken language in audio recordings. This is a project of a participant who finished 10th place and achieved an accuracy of 95% using convolutional networks: http://yerevann.github.io/2015/10/11/spoken-language-identification-with-deep-convolutional-networks/
- This paper describes a project that built a model that combines recurrent neural networks with CNNs to solve the LID: http://yerevann.github.io/2016/06/26/combining-cnn-and-rnn-for-spoken-language-identification/
- This is an LID project by Tom Herold, that used the Caffe and Tensorflow frameworks. During my project, I contacted Mr. Herold for advice about creating feature vectors from the audio files. https://github.com/twerkmeister/iLID

Project description

Dataset:

Four languages were used: English, German, French and Spanish. These languages were chosen because it is relatively easy to distinguish between them. I train and tested the model on audio from The VoxForge dataset.

The VoxForge dataset consists of multilingual speech samples available on the VoxForge website. This dataset contains short speech samples of 5–10 seconds. The audio samples are in wav format. Given that the speech samples are recorded by users with their own microphones, quality varies significantly between different samples. This dataset contains 80,000 English samples, 22,400 French samples, 29,653 German samples and 20,900 Spanish samples.

In contrast to song audio which combines background music, instruments and several singer's voices into a complex mix. Each audio file in the dataset records one speaker that speaks one language.

Feature extraction:

The first step in any automatic speech recognition system is to extract features i.e. identify the components of the audio signal that are good for identifying the linguistic content and discarding redundant information like background noise.

In the following, I present two possible features that can be extracted from audio files for machine learning:

Spectrograms:

Spectrograms are a visual representation of the audio signal after the application of the Discrete Fourier Transformation (DFT) that is an analysis of a window of amplitude measurements from the waveform that calculates the strength of frequencies inside that window. Figure 1 shows a spectrogram. Spectrograms show patterns that are visible to the human eye. These patterns are lines that are flat, rise, or drop over time and are usually present in multiple frequencies. In related work these patterns were used to recognize phonemes. The human voice mostly operates in frequencies below 10kHz. Thus, higher frequencies can be truncated.

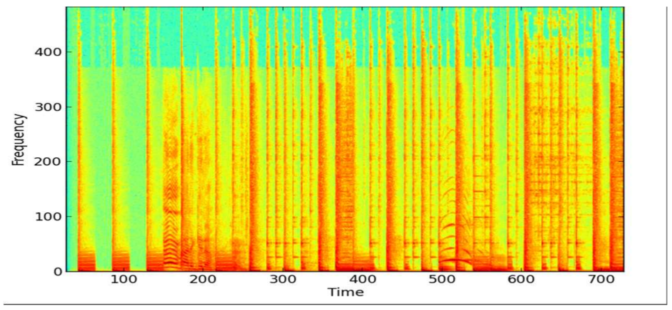


Figure 1 Spectrogram of an audio signal. The horizontal axis is the time and the vertical axis is the frequency. The darker the color the stronger the frequency at the given time.

MFCC Vectors:

In contrast to the prior feature, the Mel frequency cepstral coefficients (MFCC) are not a visual representation of the audio signal. Instead, they capture information about the audio signal in a 39-dimensional feature vector. By applying the DFT, mel filtering, and the inverse DFT the 39 coefficients can be extracted can be extracted.

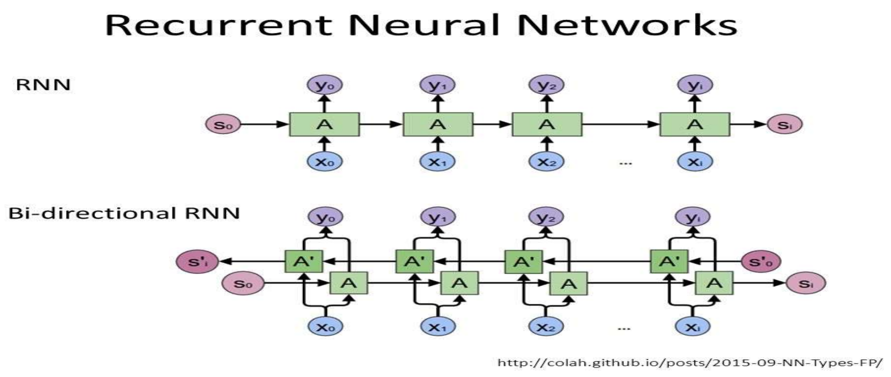
Many related projects that have succeeded with both feature extract techniqueuse. For this project, I have used the MFCC approach. I found that the MFCC's hold an accurate representation of the function represented in a spectrogram and preferred using it over deciphering relevant information from images of a spectrogram.

Based on trial and error, I cut 4 seconds of recording from each audio file. I took the 4 second recording from the middle of each audio file, to get the most amount of speech into each recording. Using Python scripts from the Librosa library, I created a MFCC vector from each wav file. Each feature vector the network receives as input hold 800 decimal values.

Network Architecture:

In order to take full advantage of the sequential characteristics of the audio data, I used a Recurrent neural network(RNN). Recurrent networks perform well on speech recognition tasks.

The RNN Is built from 128 bidirectional LSTM cells. Bidirectional RNNs are based on the idea that the output at any time may not only depend on the previous elements in the sequence, but also future elements. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.


Recurrent Neural Networks
RNN
Bi-directional RNN
http://colah.github.io/posts/2015-09-NN-Types-FP/

The network features one fully connected layer after the LSTM cells and results in two or four labels, depending on the number of languages used for training. Layers are activated using the ReLU function with 70% dropout.

The cross-entropy cost function is minimized using Adam optimizer with a learning rate of 0.001. Classification is achieved using SoftMax. The feature vectors are given to the network through Batch Gradient Descent (BGD).

Experiments/Simulations

Once I implemented the network, reaching 100% accuracy on the training set quit easy. Though signs of overfitting were very visible, as I the results on the validation set were no better than random ones. Overfitting was the most significant issue I had throughout the project.

The first thing I did to overcome the overfitting problem was introducing dropout to the NN. Though, this lead to another problem. Even with a dropout of only 0.9, the loss function was not decreasing properly, no matter the learning rate. The loss function manages to decrease nicely to a certain point and then it fluctuates. Understanding this meant the Network was trying to learn something, I dynamically lowered the learning rate as the loss function deceased. This helped with the loss minimization.

The next thing I tried, was to make the model more complex. I found that dropout on the LSTM cells themselves did not yield better results. So, I added 4 fully connected layers after the LSTM cells to preform dropout on them, but the accuracy on the training set dropped from 100% to 75%. So, I used Only one fully connected layer for the drop out and that improved the results.

In the end, the two changes that produced the best results were:

- changing the RNN cells from basic LSTM cells to bidirectional LSTM cells.
- Switching from Stochastic Gradient Descent (SGD) to BGD.

In addition to improving the accuracy, the switch to BGD made to program run much faster.

Results

Languages	Number of Training files (per language)	Number of Testing files (per language)	Accuracy
GER, FRE	150	100	77.5%
GER, FRE	1500	100	68.4%
GER, FRE	10000	5000	71%
ENG, GER, FRE, SPA	150	100	36%
ENG, GER, FRE, SPA	1500	100	40%
ENG, GER, FRE, SPA	10000	5000	39.3%

Unexplored options

- Introducing more traditional techniques, including n-gram and Chunking.
- Using a model that combines CNN and RNN.
- Using spectrograms as input to the Network.

Information

Project Github: https://github.com/AkivaGubbay/Spoken-Language-Identification

Programming language: Python 3.4.

Libraries: Tensorflow, Librosa, numpy.