Tejasvini Mavuleti

12/13/2022

**MIS 64061 – Advanced Machine Learning**

**Final Project Report**

Application of Deep Neural Network on Apple’s Siri Voice Detection

# Summary

In this assignment, I used a TensorFlow Speech Recognition dataset to test and present the application of DNN in Apple's Siri Voice Detection. The main problem I addressed was the accuracy of the wake word - "Hey Siri" and the underlying process of transforming a soundwave to an acoustic model. From what I found, the solution was to preprocess the audio inputs and remove the outliers like background noise, misspelled words, and grammatical errors. My contribution to this project is to understand and explain the role of Siri and its acoustic patterns in trigger voice detection.

Understanding

Apple's "Hey Siri" feature allows users to invoke Siri as a hands-free feature. A small speech recognizer runs all the time and listens for just these two words and gives answers to every topic under the sky. When it detects "Hey Siri," the rest of Siri parses the following speech as a command or query. The "Hey Siri" detector uses a Deep Neural Network (DNN) to convert the acoustic pattern of your voice at each instant into a probability distribution over speech sounds:

1. I worked on how it uses a temporal integration process to compute a confidence score that your phrase was "Hey Siri." If the score is high enough, Siri wakes up.
2. I explored ways to detect gender, language, and users' most frequent questions.
3. I tested the underlying technology, which can show more about machine learning than just speech recognition, and explained their importance in this project.

# Problem

PROBLEM - TRIGGER WORD DETECTION USING APPLE SIRI

Sound waves inside your body are transformed to make sequences of sounds in your device. A sound is created through tiny changes in air pressure, entering our ears as one continuous sound wave. Computers need to sense words and transform them into text. The device then measures the sound one point at a time and detects the probability of the trigger word accuracy. This happens within milliseconds, and the devices process the response. This process works with acoustic patterns, which is the soul of the idea I wanted to address. I wanted to dig deeper to look at human-machine interactions using real-world data. This project examines the power differentials between humans and Siri in the interaction between user and agent, the agent's responses to the user, and the agent's vocal characteristics, including gender. Zillion types of sounds in the universe are cut, reassembled, and spoken out. For this assignment, the main goal was to look at how these different sounds impact the output. Every word is transformed into sound waves and identified, cut, and reassembled within milliseconds to convert into spectrograms inside the device. As a text and speech, the answer is flown out in no more than 2 seconds.

Once a sound is recorded digitally, the computer has to figure out what sounds it has to pay attention to using algorithms. For example, to determine if chunks of digitized sound are words rather than sounds from a car engine or a radio, the computer applies mathematical operations to separate what is speech and what isn't. For example, there might be combinations of sounds, missed words, voice gravel, and background noise. And it is the algorithm's job to identify and remove the noise(outliers) and give accurate results.

The speed at which you speak contains most of the information. The user's speech rate adjustments ensure that missed words, background noise, and grammar mistakes are corrected. I got the data from Kaggle, where 65,000 one-second-long utterances of 30 short words by thousands of different people. Mind you, their slang, speeds, and utterances are way different. Overall, this project work has implications for human-computer interaction and linguistic adaptation.

To reduce the time and effort taken to collect this data, researchers at Apple developed a framework that leverages user engagement signals to create data-augmenting labels automatically. They report that incorporating strategies like multi-task learning and validation with an external knowledge base, the annotated data significantly improves the response accuracy in a production deep learning system. The other part was about the source which Siri gets its answers from. When you ask a question, Siri usually uses Google's search engine for her results. Alexa and Microsoft's Cortana use Bing when answering your questions. Siri does utilize Bing for some queries too.

Introducing Siri's best friend - NN

Trigger word detection is the technology that allows Apple devices to wake up upon hearing a certain word. For this exercise, our trigger word will be "Hey Siri." Whenever you hear "Hey Siri," it will make a "chiming" sound. First, small sounds of 0.01 seconds are recorded, and a cluster of 20 (0.2 seconds) at a time is continuously fed to a Deep Neural Network(DNN), which converts these into a probability density function with a threshold above which it activates the main processor.

The most interesting part I wanted to test and learn from this project was to see if AI can sense human emotions through voice modulation. "As per a new patent filed by the American tech giant, the future versions of its AI voice assistant Siri may be able to sense and interpret the facial expressions and emotions of the user when he or she is giving voice commands to it."

There can be many unusual word combinations in a single speech stream simply because many phonemes sound similar when said quickly. Sometimes the result can be a wacky sequence of words that don't make sense. To avoid this, the computer system applies models based on how people talk to determine how likely one word is to follow another. Emotional AI can read people's feelings through text, voice tone, facial expressions, and gestures and adjust its demeanor accordingly. People have the upper hand in recognizing different emotions, but AI is catching up with its ability to analyze large volumes of data. The best example I could think of is when someone asks questions about suicide. It automatically gives a helpline number and forwards you to immediate help.

It is so enormous that a little AI chip could update the network in split seconds. Like Tesla, Siri sends back every piece of information to the database, which is updated and stored. Every new utterance transforms from text to speech then it is labeled and stored in the database.

The key part of explaining my project is demonstrating how Siri takes the waveform and makes a spectrogram of our voice. Building a high-quality text-to-speech (TTS) system for a personal assistant is not an easy task. The first phase is to find a professional voice talent whose voice is both pleasant and intelligible and fits the personality of Siri. To cover some of the wide variety of human speech, we first need to record 10—20 hours of speech in a professional studio. The recording scripts vary from audiobooks to navigation instructions, from prompted answers to witty jokes. Typically, we cannot use this natural speech as recorded because it is impossible to record all possible utterances the assistant may speak. Thus, unit selection TTS is based on slicing the recorded speech into its elementary components, such as half-phones, and then recombining them according to the input text to create an entirely new speech. In practice, selecting appropriate phone segments and joining them together is not easy because the acoustic characteristics of each phone depend on its neighboring phones and the prosody of speech, which often makes the speech units incompatible with each other.

This algorithm behind Siri, Natural Language Processing, is driven by ML techniques and takes away. Siri was made to pick up keywords and important phrases. During the text–speech process can take the waveform, turn it into a spectrogram, create phonetic labels which recognize the vowel, stress labels, and pitch labels, and further decide which part gets selected during the interface.

The DNN-power (Deep Natural Network) voice trigger keeps Siri in the iCloud, which can hear the user's command of "Hey Siri" at any moment. It computes the confidence score to identify if you want to wake Siri up. Of the two layers used in Siri, one is for detection, and the other is for checking.

Imagine there are 6000 languages to translate. And to record every word from these is impossible. So instead, human speech is recorded and transcribed by other humans. This forms a canonical representation of words and how they sound aloud, dictated by real people to ensure accuracy. This raw training data is then fed into an algorithmic machine training model. The computer language model attempts to predict the transcription of arbitrary strings of words. The algorithm can improve automatically over time as it is trained with more data. Apple will tune the data internally and then move on to the next step.

# Domain & Application

I picked up a dataset where it has 65,000 one-second-long utterances of 30 short words by thousands of different people. I wanted to test the accuracy of detecting the trigger word and then checking for the user's voice in a stream of instantaneous waveforms. And from what I learned, Siri transforms our command at a rate of 16000 waveform samples per second.

A spectrum analysis stage converts the waveform sample stream to a sequence of frames, each describing the sound spectrum of approximately 0.01 sec. About twenty of these frames at a time (0.2 sec of audio) are fed to the acoustic model. Finally, a Deep Neural Network (DNN) converts each of these acoustic patterns into a probability distribution over a set of speech sound classes used in the "Hey Siri" phrase, plus silence and other speech, for a total of about 20 sound classes.

This is then forwarded to generating labeled training data and a self-supervised model. It keeps updating the errors from the text-to-speech outputs. I used the audio data to test to connect the sigmoid layers to perform temporal integration with the soundwaves into an acoustic pattern. Almost all the computation in the "Hey Siri" detector is in the acoustic model. The temporal integration computation is relatively cheap, so we disregard it when assessing size or computational resources.

The final application was with Keras. Keras worked with data recognition, converting the waveform patterns and uniquely labeling every data point. I tried to test the performance of the present speech-to-text model that relies upon the hyperparameters used in the dataset. It shows that DNN can model raw and tonal speech signals through Siri with existing recognition systems. I wanted to see how it will convert each sound wave of the acoustic model into spectrograms to visualize the transformation of processing the input trigger words.

# Technique

**Speech recognition**

I used it to recognize multiple words. Siri basically works with two main technologies - Voice Recognition and NLP(Natural Language Processing) integrated with Machine Learning. Speech recognition software can translate spoken words into text using closed captions to enable a person with hearing loss to understand what others are saying. I used speech recognition to enable trigger words for those with limited use of their hands to work with computers, using voice commands instead of typing. The Speech recognition package helped translate spoken words in French into text using closed captions to enable a person to understand what others are saying. Speech recognition helped me visualize and test audio files and translate them through gtts.

**Trigger word detection**

Similar to our first assignment on movie reviews, I tried to find the accuracy in the answers to users' questions. Here we say positive and negative examples of failed attempts. First, I labeled training data to train our model. In the data set, there was a record of 10 seconds of audio clips of people saying positive "Hey Siri" in this case and negative (words that are not "Hey Siri") examples and labels manually when the people spoke the trigger words. Labeling the data manually is complex and time-consuming. Instead, the training data is generated artificially. We would then need three audio clips: 1. Positive examples of people saying the word "Hey Siri," 1 or 2 seconds each, 2. Negative examples of people saying random words, 1 or 2 seconds each 3. Background noise, for example, coffee shop or office, 10 seconds each. The training data we have generated need to be preprocessed before sending it to a machine-learning model. Due to the variation of air pressure, sound can be produced. The input data to the model is the spectrogram for each generated audio, due to which the target will be the labels created earlier. In recent years, Deep Learning (DL) has occupied increasing attention within the industry and academic world for its high performance in various domains.

**TensorFlow**

The heart of this project. I used TensorFlow to catch sound waves. This project aims to load a wav file with TensorFlow and generate a spectrogram to look at the audio accuracy. Then, I used it to convert audio files(.wav) to spectrograms. The spectrogram is a concise 'snapshot' of an audio wave, and since it is an image, it is well suited to being input to DNN-based architectures developed for images, but I used for audio classification. Spectrogram chops up the duration of the sound signal into smaller time segments and then applies a transformation to each segment to determine the frequencies contained in that segment. It then combines the Transforms for all those segments into a single plot.

**Classifier inception**

An inception module is an image model block that aims to approximate an optimal audio sparse structure in a DNN. It allowed me to use multiple types of filter sizes in a single voice record instead of being restricted to a single filter size, which we then concatenate and pass onto the next layer during convolution. In addition, I found it to give me the highest accuracy in classification.

**Tested convolution layers**

I used a stacked 1D convolutional neural network for end-to-end small-footprint voice trigger detection in a streaming scenario. Voice trigger detection is an important speech application for users to activate their devices by simply saying a keyword or phrase. This model consists of a 1D convolution layer followed by a depth-wise 1D convolution layer. We can see that it can be expressed as a special case of the 1D convolution layer. From what I read, the state-of-the-art algorithm published uses Convnet, so I think Siri uses a similar approach. Although Apple began to publish recently, I did not see any publications about Siri. The possibility that they have a very different approach is pretty low because the original team left Apple and started a company. I did not see any big difference in their approach.

**Assigned weights and normalization**

I used batch normalization to adapt speech data in the dataset's amplitude, frequency, and time domains. I faced a problem with the split in train data and the need for normalization for the time domain. To work on this, I labeled the data to make a tractable exact computation of the sequence-level normalization. I assigned weights to balance and bridge the missing words and the noise. From there, I went on to find the model accuracy, loss, precision, and recall.

**Translation** – a basic transformation of sound waves from get text to speech(gtts)

I used a simple model to translate language with the trigger and wake words. Basically, this idea was to show how Siri operated and converts the input language with a simple "translate" command. For example, I translated French to English using gtts.

# Development

Change the balance between two kinds of error by changing the activation threshold.

During development we try to estimate the system's accuracy by using a large test set, which is quite expensive to collect and prepare but essential. There is "positive" data and "negative" data. The "positive" data does contain the target phrase. You might think that we could use utterances picked up by the "Hey Siri" system, but the system doesn't capture the attempts that failed to trigger, and we want to improve the system to include as many of such failed attempts as possible. These are early development models. We then estimate the accuracy and observe the positive or negative result. If we get a negative result, we get back to including the failed attempts. These failed attempts are sent back to the database and tested with the train data.

False detection and false wakes

We compare the score with a threshold to decide whether to activate Siri. In fact, the threshold is not a fixed value. We built some flexibility to make it easier to activate Siri in difficult conditions while not significantly increasing the number of false activations. There is a primary, or normal threshold and a lower threshold that does not normally trigger Siri. If the score exceeds the lower threshold but not the upper threshold, we may have missed a genuine "Hey Siri" event. When the score is in this range, the system enters a more sensitive state for a few seconds, so that if the user repeats the phrase, even without making more effort, then Siri triggers. This second-chance mechanism improves the system's usability significantly, without increasing the false alarm rate too much because it is only in this extra-sensitive state for a short time. Even if there are any occurrences of the target phase - they are labeled and stored in the negative data. The model does not count them as errors.

**Conclusion**

Data synthesis is an effective way to create a large training set for speech problems

In practice, selecting appropriate phone segments and joining them together is not easy, because the acoustic characteristics of each phone depend on its neighboring phones and the prosody of speech, which often makes the speech units incompatible with each other. Using the symbolic linguistic representation created by the text analysis module, the prosody generation module predicts values for acoustic features, such as intonation and duration. These values are used to select appropriate units. The task of unit selection has high complexity, so modern synthesizers use machine learning methods that can learn the correspondence between text and speech and then predict the values of speech features from the feature values of unseen text.

This model must be learned at the training stage of a synthesizer using a large amount of text and speech data. The input to the prosody model are the numerical linguistic features, such as, phone identity, phone context, and syllable, word, and phrase-level positional features converted into convenient numerical form. The model's output is composed of the numerical acoustic features of speech, such as spectrum, fundamental frequency, and phone duration. At synthesis time, the trained statistical model is used to map from the input text features into speech features, which are then used to guide the unit selection backend process where appropriate intonation and duration are crucial.

An end-to-end deep learning approach

The DNN consists mostly of matrix multiplications and logistic nonlinearities. Each "hidden" layer is an intermediate representation discovered by the DNN during its training to convert the filter bank inputs to sound classes. On iPhone, we use two networks one for initial detection and another as a secondary checker. The initial detector uses fewer units than the secondary checker.

The output of the acoustic model provides a distribution of scores over phonetic classes for every frame. A phonetic class is typically like "the first part of an /s/ preceded by a high front vowel and a front vowel. We then process the audio data asynchronously from the input audio streaming to avoid breaking audio streaming.

A sliding/moving input window is an effective way to reduce delay. The acoustic framework shows the pain points to decide how to move the inputs into the model to help reduce the time spent processing the text-to-speech data. Also, the other thing I learned was about NLP.

This natural language processing allows computers to be able to answer questions, extract information, analyze sentiment, and make predictions based on the context of the phrase we input. Machine learning and natural language processing technologies are at the core of voice assistants and through these revolutionary advancements, we need to improve the integration of deep neural networks can significantly enhance the speaker vector generation process.

# Contribution

Speech Vs. Sound

I wanted to develop my own kind of model that explains what is happening inside our devices. From where is the data coming, how is it processed, and how accurate this is? Starting from this point, the trigger word detection, I understood the concept of acoustic voice patterns and how sound waves are transformed into spectrograms. Trigger word detection is important as the entry point of user communication. This training process produces estimates of the probabilities of the phones. It states given the local acoustic observations, but those estimates include the frequencies of the phones in the training set. This may be very uneven and have little to do with the circumstances in which I will use the detector, so we compensate for the priors before the acoustic model outputs are used.

Like in the data set where we had 65000 different words, one word can sound similar to another, which could have an entirely different meaning. And this is prone to a bigger error when transforming the speech to text. For this, I wanted to understand the background of acoustic models. It is created by taking audio recordings of speech and their transcriptions and then compiling them into statistical representations of the sounds for words. The other component is called a language model, which gives the probabilities of sequences of words. The acoustic model typically deals with the raw audio waveforms of human speech, predicting what phoneme each waveform corresponds to, typically at the character or subword level.

I also tried to use a model generalization of different words and noise. We typically think of machine learning models as modeling two parts of the training data, the underlying audio signal and the randomness specific to that dataset, the noise. Fitting both of those parts increases training set accuracy, but fitting the signal also increases test set accuracy, while fitting the noise decreases both. So I used batch normalization, dropout, and similar techniques to make it harder to fit the noise and more likely to fit the signal. Increasing the amount of noise in the training data is one such approach, but it seems unlikely to be as useful. Finally, I tried to compare random noise to the model and reset the model multiple times to reprocess the dataset. I tried to use this technique to compare and contrast my results with and without noise.

In this part, we identify supervised and unsupervised audio classes. I used supervised audio classification to develop a common audio-based platform for highlight extraction that works across 65000 commonly used words. Then I used a heuristic to identify noise and adjust the trigger word accuracy and the output. I tried a combination of unsupervised and supervised learning approaches to replace the heuristic model. The proposed unsupervised framework mines the audio wav labels to detect the "Hey Siri" wake word.

Importance of product effectiveness and accessibility - people use Apple's accessibility features to personalize how they interact with their devices in ways that work for them. An accessible voice assistant supports accessibility personalization’s by design and gives everyone a great user experience, regardless of their capabilities or how they use their devices. I think getting the updated features that perform their best will change the product's effectiveness and reputation.

# Next steps

**TWD + Gender + Genre + Location + Translate**

I want to test all these factors I could add to the voice assistant model. As I thought about voice detection, gender classification came into my mind. The first step for that was the accuracy of TWD. Once I find a significant use of DNN in defining and explaining this model, I would move forward with gender. Gender detection is a sensitive topic and I am skeptical because it can falsely detect the inputs. Some of the samples may have a deep voice and could be misclassified as the opposite gender. Having said that, I refrain from trying gender detection.

My next step in building this project was to find the location of this device based on the Find My feature through Siri. If I am able to come up with a DNN model, I would then expand to find the type of questions the user would ask from a specific place. I am highly interested in making a hypothesis and linking the location and the genre of questions. So let's see where it takes me!

Question asked during the presentation

Q - Out of all the speech recognitions out there, why did you choose only Siri?

I chose Siri for my Trigger Word Detection Project because it is the most used voice assistant in the world. I wanted to see the accuracy of this voice recognition system for one more reason - Google assistant claims it is better than Siri. I just gave a shot on this to see where it lagged behind. Out of all the trigger words, if the false wakes are minimized, I think Siri will be in a better shape.

# Appendix

*Deep learning for siri's voice: On-device deep mixture density networks for hybrid unit selection synthesis*. Apple Machine Learning Research. (n.d.). Retrieved December 13, 2022, from https://machinelearning.apple.com/research/siri-voices

Doshi, K. (2021, May 21). *Audio deep learning made simple (part 2): Why Mel Spectrograms perform better*. Medium. Retrieved December 13, 2022, from https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505

Hall, Z. (2017, October 18). *Apple explains how 'hey Siri' works using a deep neural network and machine learning.* 9to5Mac. Retrieved December 7, 2022, from https://9to5mac.com/2017/10/18/how-hey-siri-works/

*Hey Siri: An on-device DNN-powered voice trigger for Apple's personal assistant*. Apple Machine Learning Research. (n.d.). Retrieved December 13, 2022, from https://machinelearning.apple.com/research/hey-siri

Inc., A. (n.d.). *Human interface guidelines*. Human Interface Guidelines - Human Interface Guidelines - Design - Apple Developer. Retrieved December 13, 2022, from https://developer.apple.com/design/human-interface-guidelines/guidelines/overview

Levy, S. (2016, August 24). *An exclusive look at how AI and machine learning work at Apple | Backchannel.* Wired. Retrieved December 7, 2022, from https://www.wired.com/2016/08/an-exclusive-look-at-how-ai-and-machine-learning-work-at-apple/

Lutkevich, B., & Kiwak, K. (2021, September 13). *What is speech recognition?* Customer Experience. Retrieved December 13, 2022, from https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition

Manankshastri. (n.d.). *Build software better, together*. GitHub. Retrieved December 13, 2022, from https://github.com/manankshastri/Trigger-Word-Detection/projects

*Personalized hey Siri. Apple Machine Learning Research*. (n.d.). Retrieved December 7, 2022, from https://machinelearning.apple.com/research/personalized-hey-siri

*Quora insincere questions classification.* Kaggle. (n.d.). Retrieved December 7, 2022, from https://www.kaggle.com/competitions/quora-insincere-questions-classification/rules

Saini, K. (2019, September 16). *Natural language processing ft. Siri*. Medium. Retrieved December 13, 2022, from https://medium.com/mytake/natural-language-processing-ft-siri-2bc7b854a2a3

Sanger. (n.d.). *DPL-425 - as an R&D scientist I would like add hi-C quants to quanthub to automate the process to reduce the time and effort taken to perform quants, persist the quants, support the pooling and reduce the chances for errors · issue #169 · Sanger/quanthub*. GitHub. Retrieved December 13, 2022, from https://github.com/sanger/quanthub/issues/169

Sharma, A. (2020, June 2). *Behind hey Siri: How Apple's AI-powered personal assistant uses DNN.* Analytics India Magazine. Retrieved December 7, 2022, from https://analyticsindiamag.com/behind-hello-siri-how-apples-ai-powered-personal-assistant-uses-dnn/

Speech recognition - human car interaction. (n.d.). Retrieved December 13, 2022, from http://humancarinteraction.com/speech-recognition.html

*Speech recognition using Deep Learning on Intel® Architecture*. Intel. (n.d.). Retrieved December 13, 2022, from https://www.intel.com/content/www/us/en/developer/articles/technical/speech-recognition-using-deep-learning-on-intel-architecture.html

*TensorFlow speech recognition challenge.* Kaggle. (n.d.). Retrieved December 7, 2022, from https://www.kaggle.com/competitions/tensorflow-speech-recognition-challenge

Writer, G. (2022, May 4). *Emotional AI: Are algorithms smart enough to decipher human emotions?* IoT For All. Retrieved December 13, 2022, from https://www.iotforall.com/emotional-ai-are-algorithms-smart-enough-to-decipher-human-emotions