IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. II (May.-June. 2017), PP 107-112
www.iosrjournals.org

Experimental speech recognition system based on Raspberry Pi 3

Hasan Gyulyustan[1], Svetoslav Enkov[2]
[1](Computer technologies, Plovdiv university "Paisii Hilendarski", Bulgaria)
[2](Computer informatics, Plovdiv university "Paisii Hilendarski", Bulgaria)

Abstract: *The paper represents a speech recognition system developed using Raspberry Pi microcomputer and USB webcam's microphone to detect and record speech. The software includes combination of different programming and scripting languages for Linux Debian based operating systems. Raspbian is used for the current experiment.*

Keywords: *pi, Raspberry, Raspbian, recognition, speech, system*

I. Introduction

Voice or speech recognition is the ability of a machine or program to receive and interpret dictation, or to understand and carry out spoken commands. For use with computers, analog audio must be converted into digital signals. This requires analog-to-digital conversion. For a computer to decipher the signal, it must have a digital database, or vocabulary, of words or syllables, and a speedy means of comparing this data with signals. The speech patterns are stored on the hard drive and loaded into memory when the program is run. A comparator checks these stored patterns against the output of the A/D converter.

Both acoustic modeling and language modeling are important parts of modern statistically-based speech recognition algorithms. Hidden Markov models (HMMs) are widely used in many systems. Language modeling is also used in many other natural language processing applications such as document classification or statistical machine translation.

1.1. Hidden Markov models

Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time-scale (e.g., 10 milliseconds), speech can be approximated as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes.

Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use. In speech recognition, the hidden Markov model would output a sequence of n-dimensional real-valued vectors (with n being a small integer, such as 10), outputting one of these every 10 milliseconds. The vectors would consist of cepstral coefficients, which are obtained by taking a Fourier transform of a short time window of speech and decorrelation of the spectrum using a cosine transform, then taking the first (most significant) coefficients. The hidden Markov model will tend to have in each state a statistical distribution that is a mixture of diagonal covariance Gaussians, which will give a likelihood for each observed vector. Each word, or (for more general speech recognition systems), each phoneme, will have a different output distribution; a hidden Markov model for a sequence of words or phonemes is made by concatenating the individual trained hidden Markov models for the separate words and phonemes.

Described above are the core elements of the most common, HMM-based approach to speech recognition. Modern speech recognition systems use various combinations of a number of standard techniques in order to improve results over the basic approach described above. A typical large-vocabulary system would need context dependency for the phonemes (so phonemes with different left and right context have different realizations as HMM states); it would use cepstral normalization to normalize for different speaker and recording conditions; for further speaker normalization it might use vocal tract length normalization (VTLN) for male-female normalization and maximum likelihood linear regression (MLLR) for more general speaker adaptation. The features would have so-called delta and delta-delta coefficients to capture speech dynamics and in addition might use heteroscedastic linear discriminant analysis (HLDA); or might skip the delta and delta-delta coefficients and use splicing and an LDA-based projection followed perhaps by heteroscedastic linear discriminant analysis or a global semi-tied co variance transform (also known as maximum likelihood linear transform, or MLLT). Many systems use so-called discriminative training techniques that dispense with a purely statistical approach to HMM parameter estimation and instead optimize some classification-related measure of the training data. Examples are maximum mutual information (MMI), minimum classification error (MCE) and minimum phone error (MPE).

Decoding of the speech (the term for what happens when the system is presented with a new utterance and must compute the most likely source sentence) would probably use the Viterbi algorithm to find the best path, and here there is a choice between dynamically creating a combination hidden Markov model, which includes both the acoustic and language model information, and combining it statically beforehand (the finite state transducer, or FST, approach).

A possible improvement to decoding is to keep a set of good candidates instead of just keeping the best candidate, and to use a better scoring function (re scoring) to rate these good candidates so that we may pick the best one according to this refined score. The set of candidates can be kept either as a list (the N-best list approach) or as a subset of the models (a lattice). Re scoring is usually done by trying to minimize the Bayes risk (or an approximation thereof): Instead of taking the source sentence with maximal probability, we try to take the sentence that minimizes the expectancy of a given loss function with regards to all possible transcriptions (i.e., we take the sentence that minimizes the average distance to other possible sentences weighted by their estimated probability). The loss function is usually the Levenshtein distance, though it can be different distances for specific tasks; the set of possible transcriptions is, of course, pruned to maintain tractability. Efficient algorithms have been devised to re score lattices represented as weighted finite state transducers with edit distances represented themselves as a finite state transducer verifying certain assumptions.

1.2. Dynamic time warping (DTW)-based speech recognition

Dynamic time warping is an approach that was historically used for speech recognition but has now largely been displaced by the more successful HMM-based approach. Dynamic time warping is an algorithm for measuring similarity between two sequences that may vary in time or speed. For instance, similarities in walking patterns would be detected, even if in one video the person was walking slowly and if in another he or she were walking more quickly, or even if there were accelerations and deceleration during the course of one observation. DTW has been applied to video, audio, and graphics – indeed, any data that can be turned into a linear representation can be analyzed with DTW.

A well-known application has been automatic speech recognition, to cope with different speaking speeds. In general, it is a method that allows a computer to find an optimal match between two given sequences (e.g., time series) with certain restrictions. That is, the sequences are "warped" non-linearly to match each other. This sequence alignment method is often used in the context of hidden Markov models.

1.3. Neural networks

Neural networks emerged as an attractive acoustic modeling approach in ASR in the late 1980s. Since then, neural networks have been used in many aspects of speech recognition such as phoneme classification, isolated word recognition, and speaker adaptation. In contrast to HMMs, neural networks make no assumptions about feature statistical properties and have several qualities making them attractive recognition models for speech recognition. When used to estimate the probabilities of a speech feature segment, neural networks allow discriminative training in a natural and efficient manner. Few assumptions on the statistics of input features are made with neural networks. However, in spite of their effectiveness in classifying short-time units such as individual phones and isolated words, neural networks are rarely successful for continuous recognition tasks, largely because of their lack of ability to model temporal dependencies.

However, recently LSTM Recurrent Neural Networks (RNNsand Time Delay Neural Networks(TDNN's)[2] have been used which have been shown to be able to identify latent temporal dependencies and use this information to perform the task of speech recognition.

Deep Neural Networks and Denoising Autoencoders[1] were also being experimented with to tackle this problem in an effective manner. Due to the inability of feedforward Neural Networks to model temporal dependencies, an alternative approach is to use neural networks as a pre-processing e.g. feature transformation, dimensionality reduction, for the HMM based recognition.

1.4. Deep Feedforward and Recurrent Neural Networks

A deep feedforward neural network (DNN) is an artificial neural network with multiple hidden layers of units between the input and output layers. Similar to shallow neural networks, DNNs can model complex non-linear relationships. DNN architectures generate compositional models, where extra layers enable composition of features from lower layers, giving a huge learning capacity and thus the potential of modeling complex patterns of speech data.[58]

A success of DNNs in large vocabulary speech recognition occurred in 2010 by industrial researchers, in collaboration with academic researchers, where large output layers of the DNN based on context dependent HMM states constructed by decision trees were adopted.[5][6] [7] See comprehensive reviews of this development and of the state of the art as of October 2014 in the recent Springer book from Microsoft

Research.[8] See also the related background of automatic speech recognition and the impact of various machine learning paradigms including notably deep learning in recent overview articles.[9][10]

One fundamental principle of deep learning is to do away with hand-crafted feature engineering and to use raw features. This principle was first explored successfully in the architecture of deep autoencoder on the "raw" spectrogram or linear filter-bank features, showing its superiority over the Mel-Cepstral features which contain a few stages of fixed transformation from spectrograms. The true "raw" features of speech, waveforms, have more recently been shown to produce excellent larger-scale speech recognition results.

1.5. End-to-End Automatic Speech Recognition

Traditional phonetic-based (i.e., all HMM-based model) approaches required separate components and training for the pronunciation, acoustic and language model. End-to-end models jointly learn all the components of the speech recognizer. This is valuable since it simplifies the training process and deployment process. For example, a n-gram language model is required for all HMM-based systems, and a typical n-gram language model often takes several gigabytes in memory making them impractical to deploy on mobile devices. Consequently, modern commercial ASR systems from Google and Apple (as of 2017) are deployed on the cloud and require a network connection as opposed to the device locally.

The first attempt of end-to-end ASR was with Connectionist Temporal Classification (CTC) based systems introduced by Alex Graves of Google DeepMind and Navdeep Jaitly of the University of Toronto in 2014. The model consisted of recurrent neural networks and a CTC layer. Jointly, the RNN-CTC model learns the pronunciation and acoustic model together, however it is incapable of learning the language due to conditional independence assumptions similar to a HMM. Consequently, CTC models can directly learn to map speech acoustics to English characters, but the models make many common spelling mistakes and must rely on a separate language model to clean up the transcripts. Later, Baidu expanded on the work with extremely large datasets and demonstrated some commercial success in Chinese Mandarin and English.

An alternative approach to CTC-based models are attention-based models. Attention-based ASR models were introduced simultaneously by Chan et al. of Carnegie Mellon University and Google Brain and Bahdanaua et al. of the University of Montreal in 2016.[3][4] The model named "Listen, Attend and Spell" (LAS), literally "listens" to the acoustic signal, pays "attention" to different parts of the signal and "spells" out the transcript one character at a time. Unlike CTC-based models, attention-based models do not have conditional-independence assumptions and can learn all the components of a speech recognizer including the pronunciation, acoustic and language model directly. This means, during deployment, there is no need to carry around a language model making it very practical for deployment onto applications with limited memory. By the end of 2016, the attention-based models have seen considerable success including outperforming the CTC models (with or without an external language model). Various extensions have been proposed since the original LAS model. Latent Sequence Decompositions (LSD) was proposed by Carnegie Mellon University, MIT and Google Brain to directly emit sub-word units which are more natural than English characters; University of Oxford and Google DeepMind extended LAS to "Watch, Listen, Attend and Spell" (WLAS) to handle lip reading and surpassing human-level performance for the first time.

The purpose of the experiment is research, selection and realization of the voice recognition system with Raspberry pi 3. Several algorythms and data structures are also surveyed.

II. Design of the speech recognition system

Hardware for the experiment are raspberry pi boards and a working microphone. Because of that Raspberry Pi doesn't include microphone could be external sound card based microphone, USB microphone or a USB web camera with included microphone. The Raspberry Pi is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. The original model became far more popular than anticipated,[6] selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. Some accessories however have been included in several official and unofficial bundles.

The Foundation provides Raspbian, a Debian-based Linux distribution for download, as well as third party Ubuntu, Windows 10 IOT Core, RISC OS, and specialized media center distributions. It promotes Python and Scratch as the main programming language, with support for many other languages. The default firmware is closed source, while an unofficial open source is available.

The Raspberry Pi 3 Model B features a quad-core 64-bit ARM Cortex A53 clocked at 1.2 GHz. This puts the Pi 3 roughly 50% faster than the Pi 2. Compared to the Pi 2, the RAM remains the same – 1GB of LPDDR2-900 SDRAM, and the graphics capabilities, provided by the VideoCore IV GPU, are the same as they ever were. As the leaked FCC docs will tell you, the Pi 3 now includes on-board 802.11n WiFi and Bluetooth 4.0. WiFi, wireless keyboards, and wireless mice now work out of the box.

Table 1. Raspberry Pi 3 Specifications

CPU	1.2 GHz 64-bit quad core ARM Cortex-A53
Memory (SDRAM)	1 GB (shared with GPU)
USB 2.0 Ports	4 (5 with the on-board 5-port USB hub)
Video input	15-pin MIPI camera interface (CSI) connector , used with the Raspberry Pi camera or Raspberry Pi NoIR camera
Video outputs	HDMI (rev 1.3), composite video (3.5 mm TRRS jack)
On-board storage	Micro SDHC slot
On-board network	10/100 Mbit/s Ethernet, 802.11n Wireless, Bluetoth 4.1
Power source	5 V with Micro USB or GPIO header
Weight	45 g
Size	85.60 mm x 56.5 mm, not including protruding connectors

Google Cloud Speech API enables developers to convert audio to text by applying powerful neural network models in an easy to use API. The API recognizes over 80 languages and variants, to support your global user base. You can transcribe the text of users dictating to an application's microphone, enable command-and-control through voice, or transcribe audio files, among many other use cases. Recognize audio uploaded in the request, and integrate with your audio storage on Google Cloud Storage, by using the same technology Google uses to power its own products.

Google Voice Search or Search by Voice is a Google product that allows users to use Google Search by speaking on a mobile phone or computer, i.e. have the device search for data upon entering information on what to search into the device by speaking.

Initially named as Voice Action which allowed one to give speech commands to an Android phone. Once only available for the U.S. English locale – commands were later recognizable and replied to in American, British, and Indian English; French, Italian, German, and Spanish.

In Android 4.1+ (Jelly Bean), it was merged with Google Now.

In August 2014, a new feature was added to Google Voice Search, allowing users to choose up to five languages and the app will automatically understand the spoken language.

All voice-recognition systems or programs make errors. Screaming children, barking dogs, and loud external conversations can produce false input. Much of this can be avoided only by using the system in a quiet room. There is also a problem with words that sound alike but are spelled differently and have different meanings -- for example, "hear" and "here." This problem might someday be largely overcome using stored contextual information. However, this will require more RAM and faster processors than these currently available in personal computers.

III. Development

The software of voice recognition system is developed mainly with C++ programming language.

Figure 1 is a basic scheme which represents the working process of the system.

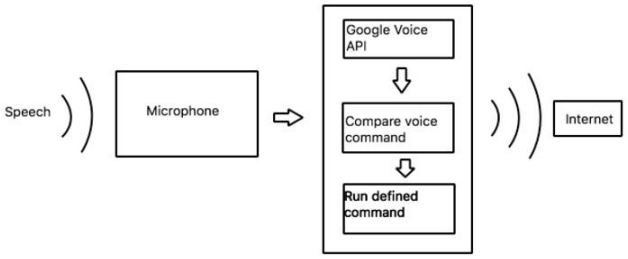

Fig. 1. Working scheme of Google speech api based speech recognition software

There are several bash scripts which install and uninstall the software to the system.

The following code is the main function of SpeechCommand.cpp file which is the main class of thee system which manage communication between google speech api and the user:

```
int main(int argc, char* argv[]) {
   SpeechCommand speechCommand;

   speechCommand.CheckParameters(argc,argv);

   FILE *cmd = NULL;
   char message[200];
```

```
        message[0] = '\0';

        speechCommand.GetConfiguration();
        speechCommand.CheckCommandLineParameters(argc,argv);

        if(speechCommand.quiet)
           fprintf(stderr,"running in quiet mode\n");
        if(speechCommand.ignoreOthers)
           fprintf(stderr,"Not querying for answers\n");
        if(speechCommand.edit) {
           speechCommand.EditConfig();
        } else if(speechCommand.continuous && speechCommand.forced_input.empty()) {
           fprintf(stderr,"running in continuous mode\n");
           if(speechCommand.verify)
              fprintf(stderr,"verifying command as well\n");
           fprintf(stderr,"keyword duration is %s and duration is %s\n",speechCommand.command_duration.c_str(),speechCommand.duration.c_str());
           float volume = 0.0f;
           changemode(1);
           string cont_com = "curl -X POST --data-binary @/dev/shm/noise.flac --user-agent 'Mozilla/5.0' --header 'Content-Type: audio/x-flac; rate=16000;' 'https://www.google.com/speech-api/v2/recognize?output=json&lang=" + speechCommand.lang + "&key=AIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw&client=Mozilla/5.0' | sed -e 's/[{}]/"/g' | awk -F\":\" '{print $4}' | awk -F\",\" '{print $1}' | tr -d '\\n"';

           while(speechCommand.continuous) {
              volume = GetVolume(speechCommand.recordHW, speechCommand.command_duration, true);
              if(volume > speechCommand.thresh) {
                 //printf("Found volume %f above thresh %f\n",volume,speechCommand.thresh);
                 if(speechCommand.verify) {
                    system("flac /dev/shm/noise.wav -f --best --sample-rate 16000 -o /dev/shm/noise.flac 1>>/dev/shm/voice.log 2>>/dev/shm/voice.log");
                    cmd = popen(cont_com.c_str(),"r");
                    if(cmd == NULL)
                       printf("ERROR\n");
                    fscanf(cmd,"\"%[^\"\n]\"",message);
                    fclose(cmd);
                    system("rm -fr /dev/shm/noise.*");
                    //printf("message: %s, keyword: %s\n", message, speechCommand.keyword.c_str());
                    if(iequals(message,speechCommand.keyword.c_str())) {
                       message[0] = '\0'; //this will clear the first bit
                       ProcessVoice(cmd,speechCommand,message);
                    }
                 } else {
                    ProcessVoice(cmd,speechCommand,message);
                 }
                 message[0] = '\0'; //this will clear the first bit
              }
              if(kbhit()) {
                 if(getchar() == 27) {
                    printf("Escaping\n");
                    speechCommand.continuous = false;
                    changemode(0);
                 } else if(getchar() == 'v') {
                    if(speechCommand.verify) {
                       printf("Turning verify off\n");
                       speechCommand.verify = false;
                    } else {
                       printf("Turning verify on\n");
```

```
                    speechCommand.verify = true;
                }
            }
        }
    }
} else {
    if(speechCommand.forced_input.empty()) {
        string command = "speech-recog.sh";
        if(speechCommand.differentHW) {
            command += " -D ";
            command += speechCommand.recordHW;
        }
        command += " -d ";
        command += speechCommand.duration;
        command += " -l ";
        command += speechCommand.lang;
        cmd = popen(command.c_str(),"r");
        fscanf(cmd,"\"%[^\"\n]\"",message);
        speechCommand.ProcessMessage(message);
        fclose(cmd);
    } else {
        speechCommand.ProcessMessage(speechCommand.forced_input.c_str());
    }
}

return 0;
}
```

The code is a part of software realization of the automated speech recognition system. It records the continuous speech and launches a connection to google speech API then prints the result to the command line. The development of this provides to run an action based on the results. Using another hardware and sensors the following system could be useful to develop more complex systems.

IV. Conclusion

This paper has realized a public API based speech recognition system with Raspberry Pi and USB microphone as hardware and several programming languages like C and bash. It gathers references to base speech recognition algorithms and structures and their important application in existing speech recognition systems. The paper also covers the hardware specification of the speech recognition system which is very important as size and supported platform. It maps an application of public speech APIs and show a realization using one of them and appropriate hardware.

Affiliation: UNIVERSITY OF PLOVDIV PAISII HILENDARSKI,
24 TZAR ASEN, 4OOO PLOVDIV, BULGARIA

References

[1]. Maas, Andrew L.; Le, Quoc V.; O'Neil, Tyler M.; Vinyals, Oriol; Nguyen, Patrick; Ng, Andrew Y. (2012). "Recurrent Neural Networks for Noise Reduction in Robust ASR". *Proceedings of Interspeech 2012*
[2]. Waibel, Alex (1989). "Modular Construction of Time-Delay Neural Networks for Speech Recognition" (PDF). *Neural Computation*. 1 (1): 39–46. doi:10.1162/neco.1989.1.1.39
[3]. Chan, William (2016). "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition". *ICASSP*.
[4]. Bahdanau, Dzmitry (2016). "End-to-End Attention-based Large Vocabulary Speech Recognition.
[5]. Yu, D.; Deng, L.; Dahl, G. (2010). "Roles of Pre-Training and Fine-Tuning in Context-Dependent DBN-HMMs for Real-World Speech Recognition". *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.
[6]. Dahl, George E.; Yu, Dong; Deng, Li; Acero, Alex (2012). "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition". *IEEE Transactions on Audio, Speech, and Signal Processing*. 20 (1): 30–42.
[7]. Deng L., Li, J., Huang, J., Yao, K., Yu, D., Seide, F. et al. Recent Advances in Deep Learning for Speech Research at Microsoft. ICASSP, 2013.
[8]. Yu, D.; Deng, L. (2014). "Automatic Speech Recognition: A Deep Learning Approach (Publisher: Springer)".
[9]. Deng, L.; Li, Xiao (2013). "Machine Learning Paradigms for Speech Recognition: An Overview". *IEEE Transactions on Audio, Speech, and Language Processing*.
[10]. Schmidhuber, Jürgen (2015). "Deep Learning". Scholarpedia. 10 (11).