ASR – A real-time speech recognition on portable devices

Ajay Shiv Sharma\*, Rahul Bhalley

Guru Nanak Dev Engineering College, Ludhiana, India

***Abstract*—This paper presents the implementation of real- time automatic speech recognition (ASR) for portable devices. The speech recognition is performed offline using PocketSphinx which is the implementation of Carnegie Mellon University’s Sphinx speech recognition engine for portable devices. In this work, machine Learning approach is used which converts graphemes into phonemes using the TensorFlow’s Sequence-to-Sequence model to produce the pronunciations of words. This paper also explains the implementation of statistical language model for ASR. The novelty of ASR is its offline speech recognition and thus requires no Internet connection compared to other related works. A speech recognition service currently provides the cloud based processing of speech and therefore has access to the speech data of users. However, the speech is processed on the handheld device in offline ASR and therefore enhances the privacy of users.**

1. INTRODUCTION

Speech is a complex phenomenon. Speech recognition has been an interesting subject of study for researchers in the field of Artificial Intelligence (AI) for so long. Today’s state-of-the-art speech recognizers are based on hidden Markov models (HMMs) to deal with the temporal variability of speech [1]. The recognition of speech takes place as the prediction of the most likely sequence-of- words using Viterbi algorithm [1]. Speech recognition is an essential aid for users with physical disabilities that limit their use of the keyboard and mouse. It has helped the people around the globe simplify their lives by performing common tasks like dictation, sending messages, checking mails without having them to look at the screens of their devices.

Speech recognition is a difficult process because the sounds made by speaker are ambiguous and also noisy. Speech recognition can be a problematic task due to mainly three reasons: segmentation [2], coarticulation [3], and homophones [4]. Let us consider that the person says “wreck a nice beach” then it may sound like “recognize speech” when spoken rapidly which is an example of segmentation problem [2]. When “s” sound at the end of “nice” and “b” at the beginning of the word “beach” are spoken swiftly then we hear something close to “sp” sound which is a problem of coarticulation [3]. Thirdly, homophones are the problem to speech recognition because the words like “to”, “two”, and “too” and also the combination of words like “real eyes”, “real lies” sound the same but have different meanings [4].

\* Corresponding author Email: [2ajayshiv@gmail.com](mailto:2ajayshiv@gmail.com)

1. HISTORY OF SPEECH RECOGINITION

Speech recognition has a long history. In 1950s and 1960s, the first speech recognition systems were built. In 1952, Bell Laboratories designed the “Audrey” system, which recognized digits spoken by a single voice. About ten years later, IBM demonstrated at the 1962 World’s Fair its “Shoebox” machine, which could understand 16 words spoken in English. In 1970s, Carnegie Mellon University’s “Harpy” speech recognition system took a big step ahead. Harpy could understand 1011 words, which is approximately equivalent to the vocabulary of an average three years old. In 1980s, speech recognition vocabulary jumped to several thousand words, and could recognize an unlimited number of words. Then in 2007, Apple announced “Siri” which could take commands of users and became their personal assistant. Since then a large number of people started relying on speech recognition for their daily tasks like searching the web or sending messages.

1. PROBLEM FORMULATION

Speech recognition has become a popular way of interacting with the devices for users. Speech recognition is being used for web search, queries, dictations, commanding the portable devices to “take a photo” or “send messages” and other related tasks. Today, the speech recognition is available nearly for anyone to use who has their devices connected to the Internet.

The Internet does play an important role in obtaining the information/connecting the people from around the globe. However, in case of non-availability of Internet, speech recognition becomes less flexible where information asked by user requires user-Internet interaction. Also, in practice, the requests made by users are not limited only to the Internet based tasks. There are many offline tasks that can be performed by the devices. The task might be to do some mathematical calculation, open some application, turn the lights of the room ON/OFF, placing a call, sending messages, dictation, or even changing the device settings. These are just a handful of commands that are possible to be performed offline when the user requires controlling his/her device with just their voice.

Currently, the devices extract the acoustical features from our voice and send them to the servers to recognize the speech to handle the requests. This means that the devices are already powerful enough to extract the acoustic features. Therefore, the recognition of speech is also possible right on the device without requiring the servers perform this task always. As stated by Apple Inc. in June 2016 at World Wide Developer’s Conference (WWDC) that their devices does 11 billion computations per photo to do objects and scene recognition. Therefore, the

978-1-5090-3480-2/16/$31.00 ©2016 IEEE

devices are also capable enough to perform speech recognition right on the device.

This provides the motivation for this work to implement Internet independent speech recognition system on the handheld devices. The offline ASR is implemented using Carnegie Mellon University’s PocketSphinx [5] (see section 4 for structure of PocketSphinx) which is the portable device implementation of SPHINX-4 [6] engine for speech recognition.

1. STRUCTURE OF POCKETSPHINX

PocketSphinx [5, 6] is mainly composed of 3 components: the front end, the decoder, and the linguist. The linguist further contains acoustic model, dictionary, and language model. To perform speech recognition pronunciation model is used along with acoustic model and language model.

The input signals are fed into the front end which is then parameterized into a sequence of features. The linguist translates the pronunciation information present in dictionary along with the language model information and also the structural information from acoustic model, into a search graph. The decoder includes the search manager which inputs the features from front end and search graph from linguist. This is the place where the actual decoding takes place and results are generated. Then these results are fed to the application.

1. BASIC SPEECH RECOGNITION PROCESS

The common steps to recognize speech are the following. First, take the waveform from speech. Then split it on utterances by silences. Afterwards, try to recognize what is being said in each utterance.

The speech can be viewed as a prediction of most-likely sequence of words given the sound from which a vector of features is extracted for speech recognition. And the most likely-sequence of words can be computed with the help of Bayes’ Rule [3]:

argnaxw1:t P(w1:t|s1:t) = argnaxw1:t P(w1:t|s1:t)P(w1:t)(1)

where,

P(w1:t|s1:t) is the acoustic model.

P(w1:t) is the language model. w stands for word.

s stands for sound.

Acoustic model describes the sounds of words that how any grapheme is spoken. Language model specifies the prior probability of each utterance - for example, that “Bayes’ theorem” is about 10,000 times more likely as a word sequence than “theorem Bayes’”. Once the acoustic and language models are defined then the most likely sequence-of-words can be obtained using the Viterbi search algorithm.

The next section describes the implementation of offline ASR on portable devices.

1. METHODOLOGY

The basic framework for offline speech recognition on portable devices is provided by PocketSphinx [5]. PocketSphinx is used for ASR on Android platforms (Android Studio) whereas the wrapper for PocketSphinx called OpenEars [7] is required when the implementation is to be done on iOS platforms (Xcode).

For the recognition of speech the acoustic model, the language model, and the pronunciation model are essential. The acoustic model of English (US) was obtained from online (see reference [8]). For the language model, statistical language modeling approach is used. And the language model of English was built using SRI Language Modeling Toolkit (SRLIM) [9]. The pronunciation model which is responsible for the recognizer to understand the pronunciation of words is constructed using Sequence-to-Sequence G2P Toolkit [10, 11] which is based on TensorFlow’s Sequence-to- Sequence model.

The construction and implementation of requirements: language model, pronunciation model in ASR are described below:

1. *Language Model*

The speech recognizer needs a way to understand that how the words are used with one another in natural language i.e. the informal grammar of that language to comprehend the user’s speech more accurately. Since the natural language is not spoken completely in accordance to the grammar rules of that language, the statistical language model will be a better approach for defining the language model.

The language model defines the different combinations of words that can be formed in a natural language. In statistical language model, given any sequence of n or less than n words (n-gram word models), provides a probability of that sequence being seen in a sufficiently large representative sample of that language. Here, statistical language modeling approach was applied to create language model for offline ASR. The statistical language model depends completely on the content of text corpus. So, to perform task-specific speech recognition the text corpus for statistical n-gram language model should be chosen appropriately.

The probabilities assigned to the words or phrases in statistical n-gram language model are in the log10 form. These probabilities define the likeliness of the occurrence of the words during speech recognition. Here, the n-gram model is actually the Markov chain of order n - 1, where n is the state of a variable. Here, a variable can be a character, a word, or even a phrase. In Markov chain, the probability of character ci depends only on the immediately preceding character and not on any other preceding state of the variable ci . So in a trigram model

[3] (order = 2), we have:

P(ci|c1:i–1) = P(ci|ci–2:i–1) (2)

To create statistical language model, various software packages have been in use for many years such as CMU- Cambridge Language Modeling Toolkit [12], SRI Language Modeling Toolkit [9]. But for this work the SRI Language ModelingToolkit [9] was used. Since the text

corpus affects the language model directly. Therefore, it seems appropriate to use corpus of transcripts of spoken language. This enables task-specific speech recognition as the speech recognizer will have a tighter probability distribution to deal with. After the collection of text corpus the trigram model was created using SRI Language Modeling Toolkit [9].

1. *Pronunciation Model*

A speech recognizer requires the information for the pronunciation of the words. This information is provided to the speech recognizer in Arpabet [13] format which is a phoneme set containing 39 phonemes for representing the pronunciation of the words where each phoneme is represented by one or two capital letters. For speech recognition, the words are to be converted into their corresponding sequence of phonemes for describing their pronunciations. To do this task, Sequence-to-Sequence G2P Toolkit [10, 11, 14] was used. This toolkit is implemented with TensorFlow’s [15] Sequence-to- Sequence model using Python. TensorFlow’s Sequence- to-Sequence model uses recurrent neural network (RNN) with long short-term memory cells (LSTM) for translation tasks. Various tasks have been accomplished successfully by using LSTM Sequence-toSequence models which includes machine translation [16] and grapheme-to- phoneme conversions [14]. The model used for the grapheme-to-phoneme conversion was trained on CMU Pronouncing Dictionary [17]. By using the same model, new words were added to the list of words in pronunciation dictionary for the recognition of new words in speech. The following table shows some examples of grapheme-to-phoneme conversion which were done using the trained model on CMU Pronouncing Dictionary [17].

# TABLE 1

SOME EXAMPLES OF GRAPHEME TO PHONEME CONVERSION

|  |  |
| --- | --- |
| Word | Pronunciation |
| SPEECH | S P IY CH |
| RECOGNITION | R EH K IH G N IH SH AH N |
| BAYES | B EY Z |
| MARKOV | M AA R K OW V |
| VITERBI | V AY T ER B IY |

1. CONCLUSION

This paper explained the implementation and the requirements for implementing offline ASR on portable devices. To create the language model, statistical language modeling approach was used. To construct the pronunciation dictionary of words, the Machine Learning was applied with Sequence-to-Sequence G2P Toolkit and the model used for grapheme-to-phenome conversion was trained on CMU Pronouncing Dictionary which further leads to produce pronunciations of new words.

The main reason behind encouraging offline speech recognition is that it can be helpful in emergency

situations when the Internet connection is unavailable. Another important reason is to protect the privacy of users as many speech recognition services currently provide the cloud based processing of speech. The offline speech recognition can help in keeping the speech data personal to users as the speech recognition task is performed independently on the handheld devices.

# References

1. L. Rabiner and B.-H. Juang, "Fundamentals of speech recognition," 1993.
2. H. Lieberman, A. Faaborg, W. Daher, and J. Espinosa, "How to wreck a nice beach you sing calm incense," in *Proceedings of the 10th international conference on Intelligent user interfaces,* 2005, pp. 278-280.
3. S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and

D. D. Edwards, *Artificial intelligence: a modern approach* vol. 2: Prentice hall Upper Saddle River, 2003.

1. (Accessed 14-July-2016). Available: https://en.wikipedia.org/wiki/Homophone
2. D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, "Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings,* 2006, pp. I-I.
3. P. Lamere, P. Kwok, W. Walker, E. B. Gouvêa, R. Singh, B. Raj, et al., "Design of the CMU sphinx-4 decoder," in *INTERSPEECH,* 2003.
4. (Accessed 14-July-2016). Available:<http://www.politepix.com/openears>
5. (Accessed 14-July-2016). Available: https://sourceforge.net/projects/cmusphinx/files/Acoustic and Language Models/US English
6. A. Stolcke, "SRILM-an extensible language modeling toolkit," in *Interspeech,* 2002, p. 2002.
7. (Accessed 14-July-2016). Available: https://github.com/cmusphinx/g2p-seq2seq
8. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems,* 2014, pp. 3104- 3112.
9. R. Rosenfeld and P. Clarkson, "Statistical language modeling using the CMU-Cambridge toolkit," 1997.
10. (Accessed 14-July-2016). Available: https://en.wikipedia.org/wiki/Arpabet
11. K. Yao and G. Zweig, "Sequence-to-sequence neural net models for grapheme-to-phoneme conversion," *arXiv preprint arXiv:1506.00196,* 2015.
12. (Accessed 14-July-2016). Available: [http://www.tensorflow.org](http://www.tensorflow.org/)
13. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate*," arXiv preprint arXiv:1409.0473,* 2014.
14. (Accessed 14-July-2016). Available:<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>