Available online at www.sciencedirect.com

ScienceDirect

IFAC-PapersOnLine 49-25 (2016) 013–018

Speech Control of Measurement Devices

Jiří Špale*, Cedric Schweizer*

*Furtwangen University, Faculty of Computer Science, Furtwangen, Germany
(e-mail: {spale, schweize}@hs-furtwangen.de)

Abstract: The technology of speech recognition has undergone a rapid development in recent years. Available are local and server-based, commercial and open source solutions. The detection rate and success vary. The subject of this paper is a comparative study of some selected APIs, libraries and SDKs in combination with Qt with the aim to develop an Android or iOS app that exchanges data with measuring instruments via Bluetooth. The chosen solution was realized.

Keywords: Speech Recognition, Text-to-Speech (TTS), Pocketsphinx, Nuance NDEV, Speech API, OpenEars, JSGF Grammar, Android, iOS, Qt

1. INTRODUCTION

Testo AG is a global manufacturer of measuring instruments for detection of various physical quantities. These devices are used in gas analysis, heating and refrigeration equipment, calibration and pharmaceutical technology, optical and thermal imaging inspection, to name just a few applications. It happens that the technicians who use these devices, have to place a measuring probe or adjust control elements and at the same time need to navigate through the menus of the measuring instruments. Since this often is difficult, testo developed a family of devices, which can exchange data with a smartphone or tablet via Bluetooth. The corresponding app testo Smart Probes is available for Android and iOS and can be downloaded from Google Play or iTunes. A voice control of this app would further facilitate the work or even help to reduce the number of on-site employees. In order to ensure compatibility with other operations performed by testo developments, it was a requirement to use *Qt*. Subject of the project described in this article, which testo entrusted the Faculty of Computer Science of Furtwangen University with, was design, analysis and comparison of possible solutions for voice control of an app that should expand the existing testo Smart Probes app. The selection of the optimum variant should be done under both technical and economic aspects. The app should know at least the following functions:

- Control the app by voice
- Speech to Text - dictate using speech recognition
- Input in multiple languages
- Text to Speech - read out the measured values
- Supported operating systems: Android & iOS
- If possible no license fees for commercial use
- Find the optimal ratio between performance and accuracy

If possible the app should work without an Internet connection. As additional criteria, the power consumption should be taken into account, measured and tested. The user should be informed about the success or failure of the speech recognition by appropriate sounds. The theoretical solution had to be verified with developing of a prototype.

2. SPEECH PROCESSING AND RECOGNITION OVERVIEW

In automatic speech recognition (ASR), there are multiple factors which influence the size of the app, the accuracy and speed of detection. For device control, speaker-independent systems with very limited vocabulary of isolates words or unchanging short word sequences are needed.

Figures 1 and 2 show the general block diagram of speech recognition.

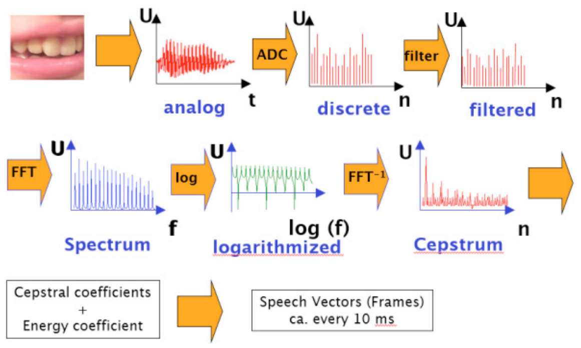

Fig. 1. Pre-processing

By filtering (Fig. 1) disturbing noises should be suppressed. The term "Cepstrum" means spectrum of a function in the frequency domain. The dimension of the independent variable of the Cepstrum - "quefrency" - is equivalent to the size of the variable, from which the original spectrum was formed, in this case the discrete time. In the Cepstrum, the amplitude of the signal is reduced, therefor harmonic components of the signal will become more apparent. This allows conclusions whether in the generation of the signal vocal cords or vocal tract were involved. For even more detailed consideration of processing the square of the spectrum's magnitude is filtered by a filter from the Mel-filter bank (Pfister 2008) usually before the logarithm is applied. Instead of the inverse Fourier transform (IFT, FFT-1), the discrete cosine transform (DCT) is often used. This is possible because for real-valued input, the real part of the

10.1016/j.ifacol.2016.12.003

DFT is a kind of DCT. The reason why DCT is preferred is that the output is approximately decorrelated. Decorrelated features can be modelled efficiently as a Gaussian distribution with a diagonal covariance matrix. For more details, see e.g. (Pfister 2008).

The result of the pre-processing are Speech Vectors in which the Cepstral coefficient and 1 energy coefficients - Mel-Frequency Cepstral Coefficients (MFCC) are stored.

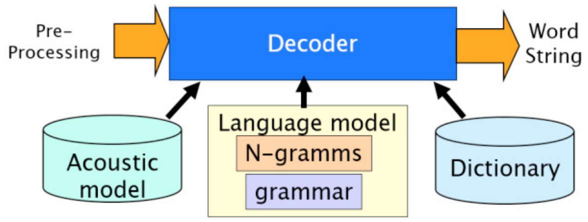

Fig. 2. Phoneme-based decoder

Almost all modern speech recognition systems use the phoneme-based process for recognizing (Fig. 2). The task of the **acoustic models** is determining all possible combinations of matching phonemes to the input signal by means of Hidden Markov Models (HMM). Using the **dictionary**, the matching combinations are compared with the words of the word pool. The selection is limited to this pool. The **grammar** assigns a function to each word using grammatical rules. A part of the **language model** are statistics that define the probability that a combination of e.g. 3 words (**Trigram, N-gram**) may occur. Finally, the combination with the highest probability is selected.

3. PROJECT APPROACH

3.1 Key technologies compared

The following technologies were contemplated:

- **CMU Sphinx**, a very common open source framework written in C, developed at Carnegie Mellon University. The processing occurs offline. Extension modules for a large number of languages are available. The framework stays under a BSD license - there is no restriction against commercial use or redistribution. For quality speech models very good detection rate was found out (Harvey et al. 2010). However, a disadvantage is the implementation effort and maintenance, as well as high effort for creating new language and acoustic models. Some smartphones had problems with recognizing many different words (> 1000) or long sentences (> 3 words).

- **Pocketsphinx**, a variant of CMU Sphinx optimized for mobile devices (Huggins-Daines 2015). It can be configured to use fixed-point or floating-point arithmetic. Floating-point arithmetic brings benefits when the processor architecture supports floating-point operations by hardware. Older or low-end mobile devices use ARM processors which have no hardware support for floating-point operations, and if so, then the floating-point arithmetic is much slower than the fixed-point. For this reason, the fixed-point arithmetic is offered by Pocketsphinx, compare with the Spectral Analysis described in (Spale 2009). However, to calculate with rational numbers, signed 32-bit integers with a radix point at bit 16 (Q15.16 format) are used, rather than the classical integers (Sorensen 1987). On the architecture level it is necessary to pay attention to calculations for time-critical code sections. If possible, these should run using processor registers and the number of memory accesses should be minimized (Huggíns-Daines et al. 2006).

- **Nuance NDEV API**, a cloud-based HTTP API by Nuance Mobile. On the one hand, the required offline functionality would not be complied and the success of speech recognition would depend on the quality of the Internet connection. A further consideration would be the issue of data security. On the other hand, the development, implementation and maintenance would be minimal, no language packs would have to be installed. The app would be platform independent, and they could be kept very small. A major disadvantage is the absence of keyword spotting. Looking for a hotkey in the amount of all recognized words is certainly much more resource hungry.

- **Native implementation** - with the standard tools of considered operating systems. At Android, the feature Okay Google Everywhere was tested. The speech recognition starts with the hotkey "OK Google". Currently it is not possible to define own hotkeys, since these are defined on the CPU level; only the "Original Equipment Manufacturer" have access there. Furthermore, the speech recognition has to be started e.g. by a button click. Anyway, the missing possibility to define own hotkeys to start "permanent listening" is the only restriction. The implementation of speech recognition follows the design pattern Inversion of Control: The button click invokes a function, in which the listening is started, a suitable Language Model is selected, eventually a graphical prompt issued and the code is transferred, by which, later in callback, the initiator of the callback can be detected. The callback function is started after completion of the speech input. If the initiator of the callback really is the request for speech recognition, all recognized words, sorted by their matching probability, will be loaded in a list. Finally, the most likely result stands at index 0.

3.2 Decision-making

For the decision 13 criteria were defined:

1. *Implementation and linking effort*
   [function adaptation to target platform; integration of API or library in the program; setting up the SDK]
2. *Advanced functions* [Voice Activity Detection, Wake-up Recognition, Noise Reduction, Text-to-Speech, …]
3. Ongoing license costs
4. One-time license costs
5. Terms of a license
6. Quality / Performance of recognition by dictation
7. Quality / Performance of recognition of isolated words on voice control