MULTI-LINGUAL SPEAKER-INDEPENDENT VOICE USER INTERFACE FOR MOBILE DEVICES

Juha Iso-Sipilä, Marko Moberg, Olli Viikki
Nokia Technology Platforms, Tampere, Finland
Email: {juha.iso-sipila, marko.moberg, olli.viikki}@nokia.com

ABSTRACT

This paper presents a multi-lingual speaker-independent voice User Interface (UI) that has been implemented for Nokia S60 mobile phones. The paper concentrates on discussing the specific approach used for achieving a multi-lingual and configurable speech recognition and speech synthesis system. The main applications are speaker-independent name dialing and voice commands. The novelty of the applications is that the user does not need to train the voice dialing system but the application reads the user's phonebook and generates the required voice tags automatically. The speaker-independent voice dialing has already been introduced in regions where the language diversity is not so great. The system presented in this paper is the first of its kind to support both speech recognition and speech synthesis in more than 40 languages in embedded devices with strict memory and performance requirements.

1. INTRODUCTION

Speaker-dependent voice dialing, including name dialing and voice commands, has been the de-facto feature in mobile devices today. The argument is based on the fact that almost all major mobile phone manufacturers have speaker-dependent voice dialing in their handsets. The benefits of such systems are multiple, ranging from language-independency and low complexity to high recognition accuracy and readily available voice feedback. These benefits do not, however, diminish the fact that the user needs to train each voice tag separately. This limits the number of voice tags to only a few and user has hard time to remember the exact utterance that was spoken during training procedure.

Speaker-independent speech recognition is very intriguing for dialing the contacts in the mobile device. The user does not need to train the voice tags, but the system does this for the user transparently, automatically and without any interaction with the user. The obvious challenges due to speaker-independent speech recognition are language-dependency and increased complexity. The language of the utterance should be known and only then, language-specific methods can be used to find out the pronunciation of the word. Also, pronunciation modeling for every supported language requires a significant amount of development work. Furthermore, the system should be capable of detecting and processing multiple languages at once, which creates several new requirements for the system. These requirements include automatic language identification, non-native word pronunciation modeling, multiple pronunciation support, and multi-lingual acoustic models, etc.

The paper has been constructed as follows: Chapter 2 introduces the multi-lingual framework that has been developed in order to cope with the problems described in the previous paragraph. Chapters 3, 4 and 5 continue on the separate sub topics of multi-lingual text-processing, speech recognition and speech synthesis, respectively. Chapters 6 and 7 discuss the specific user interface and usability related issues found during the development and first deployments. The paper is finished with conclusions, acknowledgements and references to previous work.

2. MULTI-LINGUAL FRAMEWORK

The multilingual speaker-independent speech recognition framework has been introduced in [1]. The basic principles are illustrated in Fig. 1. Multilinguality, i.e. the capability of being able to support a multiple set of languages at the same time, has been the starting point of the whole system design.

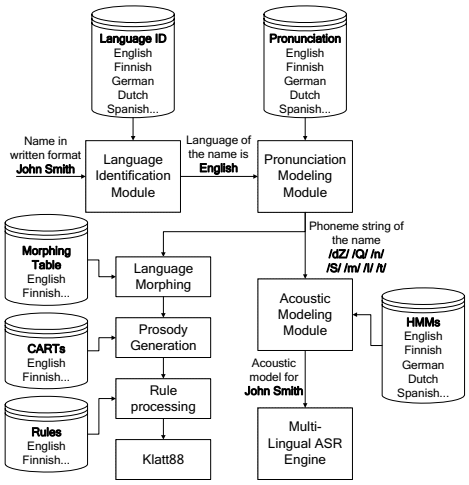

Figure 1. Block diagram of a multi-lingual voice UI system.

In voice dialing, no assumptions can be made on the language identity of the vocabulary items, but the phonebook can include names from multiple different languages. For this purpose, the voice UI system includes an automatic language identification module. Once the language is known (guessed) a language specific pronunciation modeling method is applied to get the pronunciation of the word. Multiple pronunciation variants are typically created in order to cope with the uncertainty related to language identification. Once the phonetic sequence is known, the word can be added to the active vocabulary of the multi-lingual speech recognizer. Similarly, when the speech synthesizer is used, the pronunciation module finds the pronunciation for the word and

then the speech synthesizer uses the phonetic sequence to generate the proper speech output.

It should be noted that the pronunciation module is shared by the speech recognizer and the speech synthesizer. This enables lower memory requirement for the Text-to-Phoneme (TTP) module and also makes sure that the speech recognition and speech synthesis phonetics have been developed in co-operation for each language. The TTP module generates additional information, such as tone and stress, for the speech synthesizer.

The Speech Recognition (ASR) framework uses mono-phone HMMs with beam-search. A multi-state multi-mixture background model is used for noise modeling.

The Text-to-Speech (TTS) engine is based on the Klatt's formant synthesizer and is controlled by a dedicated multi-lingual control language that enables processing of the phoneme sequence for each supported language, including tone control for tonal languages.

3. MULTI-LINGUAL PHONETICS, TEXT-PROCESSING AND PRONUNCIATION MODELING

In order to build a truly multi-lingual system, it is desired that several phonemes are shared between languages to minimize the memory use. The multi-lingual phonetic alphabet, called Nokia IPA (NIPA), has been developed to support all voice dialing languages for both speech recognition and speech synthesis. Since ASR and TTS share the same pronunciation model, special attention has been put to the accuracy of the language-specific phonetics to maintain the high phonetic accuracy of synthesized speech.

The text processing steps before speech recognition or speech synthesis can be divided into the following steps that are presented in the following sub-sections:

- Text pre-processing, character conversions, acronym detection
- Language identification from text
- Pronunciation modeling

3.1. Text pre-processing

The task of text processing is to prepare the input text string to be suitable for language identification and pronunciation modeling. The steps taken in this part consist of acronym detection, marking of digits, deletion of non-pronounceable characters (such as !, ¤, /), Romanization (e.g. transliteration of Greek to get English pronunciation) and language specific character conversions.

The acronym detector processes the text in such a way that words that are written in capital letters can be considered to be acronyms if they fulfill some other requirements. The detection itself is quite complex procedure and the detailed overview is left out of this presentation.

From the end-user point of view, it is vital that the system can cope with various character sets and languages. Consider Swedish name 'Håkan' that is found in a device having English UI. The system should be able to cope with this the best possible way. Text pre-processing for English notices that there is an unknown letter 'å'. Luckily, the letter is known by the overall system and is translated to letter 'o', which is the closest estimate. Similarly, other script mismatches are handled by way of letter transformations or Romanization.

3.2. Language identification from text

The task of language identification from text is considered really difficult in case of short sentences and words. For proper names, such as given and family names, this is especially difficult. It is characteristics to language identification that there are cases for which there are simply no correct answers, e.g. language identity for a name "Peter". Furthermore, the users of mobile phones can have a wide variety of nationalities in their contact data, which leads to enormous problems in detecting the language of the contact name. Due to these problems, language identity can be considered one of the worst performing parts of the overall system and all available language information, e.g. UI language, should be utilized if just possible. N-Grams, neural networks and decision trees have been proposed for language identification [2].

3.3. Pronunciation modeling

This part of the text processing is responsible of finding a pronunciation to the given text string. The language of the text string should be known at this point. In general, the pronunciation can be found by applying several types of methods such as rules, dictionaries, look-up tables, neural networks or decision trees. The mobile platform has limitations on the amount of memory that can be used to store the pronunciation methods and hence a general dictionary is out of question for every language. The output of the pronunciation model is a sequence of phonemes describing the pronunciation of the word(s).

The Nokia voice UI utilizes three types of pronunciation modeling methods, namely lookup-tables, rules and decision trees (DT) [3]. The lookup tables are used for known voice commands and some special cases that cannot be easily handled with rules. The rules are used for majority of languages while decision trees are only used for some irregular languages, such as English and Thai. The Table 1 shows the pronunciation model types and their sizes for a few languages.

	English	Finnish	Thai	Mandarin	French
Type	DT	Rule	DT	Rule	Rule
Memory	43 kB	4 kB	85 kB	76 kB	15 kB

Table 1. Pronunciation methods and sizes for some languages

4. MULTI-LINGUAL ASR FOR VOICE DIALING

Once the phonetics for multi-lingual speech recognition and synthesis have been specified, the development of the multi-lingual speech recognizer can be started. The most important pre-condition in the development is sufficient amount of training data to cover the phonemes of the multi-lingual phonetic definition. Second most important requirement is that there is enough testing material to verify the performance of the system in all the desired languages.

4.1. Acoustic model training

The basis for multi-lingual HMM estimation is the phoneme inventory designed for multiple languages. The training data annotation is constructed accordingly. The main input to the training process is the set of languages that should be included in the resulting HMM. According to the set of languages, a proper set

of training data is chosen for the training process. The important part here is that the languages used for the actual training can be different from the target languages. To have a proficient phonetic coverage for certain phonemes it is necessary to take some extra data from other languages. Also, some of the target languages may not have any training data and this must be compensated by adding some more languages from the training databases. The training procedure itself is Maximum Likelihood estimation using Baum-Welch algorithm.

4.2. Multi-lingual speech recognition engine

The speech recognition engine can be divided into two parts: feature extraction (front-end) and Viterbi decoder (back-end).

4.2.1. Front-end

The front-end is a standard MFCC (12 static coefficients, C0 and their 1st and 2nd order derivatives) front-end with some refinements. The basic difference to the normal MFCC front-end is the mean and variance normalization that is applied to the feature vector components [4].

The normalization has two major effects to each FV component stream:

- The statistics of the signal is limited to the boundaries given by the statistics estimation buffer.
- The normalization enables easier combination of speech databases from various sources and languages.

4.2.2. Back-end

The speech recognition engine has been tailored for the isolated word recognition task and uses a trained background model. The four main features of the back-end are given next:

- Optimized tree-based grammar for the isolated word recognizer
- Beam search with path pruning
- Scalar quantization of FV and model parameters to speed-up computation and reduce memory consumption
- Speaker-adaptation to improve recognition accuracy over time

All the features given above are described more thoroughly in Vasilache et al [5].

5. MULTI-LINGUAL SPEECH SYNTHESIS

Speech synthesis is required to confirm the recognition result to the user. Development of speech synthesis is significantly more time consuming than development of speech recognition due to much more strict requirements for the correctness of phonetics and pronunciation modeling. Furthermore, more language knowledge is needed to be able to objectively measure the goodness of the speech synthesizer for a given language.

The speech synthesizer used in Nokia voice UI is a formant synthesizer based on the work done by Klatt [6][7]. The strict limitation on memory and complexity does not allow use of more sophisticated TTS techniques, such as concatenative waveform synthesizers. A block diagram of the synthesis module was already presented as part of Figure 1.

However, some shortcuts have been taken to have full language support for over 40 languages. For certain languages, so called language morphing is utilized [8]. The morphing is such that for an unsupported language *A*, native TTS engine of language *B* is used. The basic requirement is that the languages should be phonetically and prosodically close enough to be able to generate adequate quality speech for language *A*. Table 2 below lists some language morphing that takes place in the TTS system. The source language does not have native TTS support. The target language has native TTS support and is used to generate the speech for the source language.

Source language	Norwegian	Estonian	Bahasa Malaysian	Ukrainian
Target language	Swedish	Finnish	Bahasa Indonesian	Russian

Table 2. List of some language mappings in the TTS system.

The language information and the phoneme string, either unmodified or modified by morphing, are passed to the prosody generation (See Fig. 1). The language specific CARTs (Classification and Regression Trees) trained with annotated utterances predict syllable boundaries, syllable stress, segment durations and intonation [9].

A set of language specific rules is applied to provide correct Klatt-parameters to the synthesizer. The co-articulation effects are taken into account by rules, which define the parameter transitions. The rules are based on context information, phoneme labels and other linguistic features such as manner and place of articulation. All the language specific information is stored as data to make the language development easier and to separate it from the engine software development. The overall size of a typical language specific data is 15kB.

Parameter frames are then constructed and fed into the Klatt88 synthesizer to create the speech waveform. The frame contents are updated every 5 ms.

6. VOICE USER INTERFACE

For the end-user, there is no automatic built-in value in a voice controlled UI. A perfectly working speech recognizer and text-to-speech systems can be spoilt if the end-user interface is badly designed. The main design objective must be that the user can complete the task in voice UI, i.e. voice dialing or voice command, more efficiently than using conventional input/output modalities. Ease-of-use and the speed of interaction are the two most important requirements for the voice user interface. From these core requirements, the main characteristics of the voice UI were specified as follows:

1. Tight integration between voice and visual UI – voice UI may not be a standalone application.
2. Voice interaction should not be overused in input and output
3. Easy cancellation in case of recognition errors
4. User customization

Voice input/output must be an integral part of the whole UI of the device. If the user needs to start the voice UI functionality by first searching for a dedicated speech application, it is obvious that the speed objectives set for the interaction are very difficult to meet. A dedicated “voice key” is a very desired so that can be used to start voice interaction from any state of the UI by a simple button press.

It is also important not to overuse speech in UI. In many situations, e.g. when scrolling up/down a list view, it is much

faster to use keypad than a voice command. It should also be remembered that every time when the recognizer is started, there is a possibility for a recognition error. In speech output, one also needs to be careful not to use too much spoken prompts. While the first-time user may appreciate lengthy instructions spoken by the machine, e.g. "Please say the name or command", these prompts will become very irritating if the user is continuously using the system.

The fact that speech recognizers make mistakes must also be considered in UI design. There must always be an easy way to cancel the UI action in case a recognition error has occurred. One should also avoid too complicated error correction schemes, e.g. confirmation queries, as those make the dialogues longer and reduce the task completion speed. In many cases, a much faster response is achieved by asking the user to repeat the command rather than starting a long error correction dialogue.

UI personalization is an important element in mobile devices. End-users have almost endless possibilities to update the UI of their phones. It is therefore essential that this is also feasible for voice UI. In Nokia voice UI, rather than having a fixed list of pre-defined voice commands, the users are able to manage their voice command set through the voice commands application. They can create new voice shortcuts and modify the existing commands.

6.1. User Guidance

For consumers, voice control is a very novel concept. There are no well established usage paradigms in voice UI as in more mature input/output modalities. It is very important to guide the users so that they are always confident in *What*, *When* and *How* to command the device. The first use impression is crucial when the users are judging the usefulness of new technology, and therefore, it is necessary that the users understand the limitations of the technology. When the users reported low performance in early trials, the following user-related problems were often the reasons behind the low performance:

- The user used commands and names that were not included in the vocabulary
- The user had "unspeakable" entries (strange acronyms, characters) in their phonebook.
- The users spoke "too early" and the recognizer was not listening the input
- The user spoke in a sloppy manner.
- The users spoke the first and last name in a different order than they had written in the phonebook.

Many of the above reasons are very characteristics to speech recognition and they can be avoided if the users are aware of the limitations. When introducing the new technology, it is thus vital to guide the users to use the technology in the appropriate way.

7. FIRST END-USER LEARNINGS AND CONCLUSIONS

When writing this paper, the first two Nokia products, Nokia N70 and N90 devices, have been in sales for a couple of months and only a little end-user feedback is available. There are, however, some conclusions that can be drawn from the feedback received up to now.

In general, the users have been impressed with the capabilities of the state-of-the-art voice recognition technology. The fact that the system works without any training in all languages is seen as a major improvement over the speaker-trained recognizers that have been integrated in mobile devices already for years. However, at the same time, it has been observed that voice UI is still a technology-driven feature – the use of the technology is not always obvious for the end-users and many technical shortcomings reduce the user experience. The performance variation across the speakers and languages is something that should be improved.

Regarding text-to-speech, the feedback has been positive even though the formant synthesis technology dates back from the "Stone Age". Obviously, when listening to short phrases/names, the main requirement is that the speech output is intelligible and naturalness has only a secondary priority.

The customization aspects of voice UI have received positive comments and the users have already learned that they can customize commands according to their own taste. Several comments requesting more voice command functionality have been received. This is obviously a very positive feedback indicating that the users find the voice UI useful and would like to have even more voice features in the future versions.

8. ACKNOWLEDGEMENTS

The effort to put together the Nokia voice UI system was started sometime at the end of the previous millennium. Since then, several people have contributed the development work. The authors would like to give their sincere thanks to all those people at Nokia Technology Platforms and Research Center who have over the years been involved in the development of the system described in this paper.

9. REFERENCES

[1] O. Viikki, "ASR in Portable Wireless Devices", *Proc. Of IEEE Workshop on Automatic Speech Recognition and Understanding*, Madonna di Campligione, Italy, 2001.

[2] J. Tian, and J. Suontausta, "Scalable Neural Network Based Language Identification from Written Text", *ICASSP 2003*, IEEE, Hong Kong, pp. 48-51, 6-10 April 2003.

[3] J. Suontausta, and J. Tian, "Low memory decision tree method for text-to-phoneme mapping", *Proc. of IEEE Workshop on Automatic Speech Recognition and Understanding*, USA, 2003

[4] O. Viikki, D. Bye, and K. Laurila, "A recursive feature vector normalization approach for robust speech recognition in noise", *Proc. of International Conference on Acoustics, Speech, and Signal Processing*, pp 733-736, Seattle, WA, USA, 1998.

[5] M. Vasilache, J. Iso-Sipilä, and O. Viikki, "On a practical design of a low complexity speech recognition engine", *ICASSP 2004*, IEEE, Montreal, Canada, pp. 113-16, 17-21 May 2004

[6] D. H. Klatt, "Software for a cascade/parallel formant synthesizer", *Journal of the Acoustical Society of America*, vol. 67, pp. 971-995, 1980

[7] D.H. Klatt, and L.C. Klatt. "Analysis, synthesis, and perception of voice quality variations among female and male talkers", *Journal of the Acoustical Society of America*, 87(2), 1990, p 820.

[8] M. Moberg, K. Pärssinen, and J. Iso-Sipilä, "Cross-Lingual Phoneme mapping for Multilingual Synthesis Systems", In *Proceedings of ICSLP 2004, Jeju, Korea*, 2004

[9] A. Black, P. Taylor, and R. Caley, 1999. *The festival speech synthesis system, system documentation*, Centre for Speech Technology Research, University of Edinburgh.