Festival-si: A Sinhala Text-to-Speech System

Ruvan Weerasinghe, Asanka Wasala, Viraj Welgama and Kumudu Gamage
Language Technology Research Laboratory, University of Colombo School of Computing,
Colombo, Sri Lanka
arw@ucsc.cmb.ac.lk, awasala@webmail.cmb.ac.lk, vwelgama@webmail.cmb.ac.lk,
kgamage@webmail.cmb.ac.lk

Abstract

This paper brings together the development of the first Text-to-Speech (TTS) system for Sinhala using the Festival framework and practical applications of it. Construction of a diphone database and implementation of the natural language processing modules are described. The paper also presents the development methodology of direct Sinhala Unicode text input by rewriting Letter-to-Sound rules in Festival's context sensitive rule format and the implementation of Sinhala syllabification algorithm. A Modified Rhyme Test (MRT) was conducted to evaluate the intelligibility of the synthesized speech and yielded a score of 71.5% for the TTS system described.

1. Introduction

A Text-To-Speech synthesizer is a computer-based system capable of converting computer readable text into speech. The conversion of text to speech involves many important processes. These processes can be divided mainly in to three stages; text analysis, linguistic analysis and wave-form generation [1]. The text analysis stage is responsible for converting the non-textual content into text. This stage also involves tokenization and normalization of the text; identification of words or chunks of text. Text normalization establishes the correct interpretation of the input text by expanding the abbreviations and acronyms. This is done by replacing the non-alphabetic characters, numbers, and punctuation with appropriate text-strings depending on the context. The linguistic analysis stage involves finding the correct pronunciation of words, and assigning prosodic features (e.g. phrasing, intonation, stress) to the phonemic string to be spoken. The final stage of a TTS system is waveform generation which involves the production of an acoustic digital signal using a particular synthesis approach such as formant synthesis, articulatory synthesis or waveform concatenation [6]. The text analysis and linguistic analysis stages together are known as the Natural Language Processing (NLP) component, while the waveform generation stage is known as the Digital Signal Processing (DSP) component of a TTS System. In this paper, we describe the implementation and evaluation of a Sinhala text-to-speech system based on the diphone concatenation approach. The Festival framework [1] was chosen for implementing the Sinhala TTS system. The Festival Speech Synthesis System is an open-source, stable and portable multilingual speech synthesis framework developed at the Center for Speech Technology Research (CSTR), of the University of Edinburgh.

TTS systems have been developed using the Festival framework for different languages including English, Japanese [1], Welsh [12], [2], Turkish [9], and Hindi [5], [8], Telugu [3], [5], among others. However, no serious Sinhala speech synthesizer has been developed this far. This is the first known documented work on a Sinhala text-to-speech synthesizer. The system is named “Festival-si” in accordance with common practice.

The rest of this paper is organized as follows: Section 2 gives an overview of the Sinhala phonemic inventory and writing system; Section 3 explains the diphone database construction process; the implementation of natural language processing modules is explained in section 4. Section 5 discusses the potential applications while Section 6 presents an evaluation of the current system. The work is summarized and future research directions and improvements are discussed in the last section.

2. Sinhala Phonemic Inventory and Writing System

2.1. The Sinhala phonemic inventory

Sinhala is one of the official languages of Sri Lanka and the mother tongue of the majority - 74% of its population. Spoken Sinhala contains 40 segmental phonemes; 14 vowels and 26 consonants as classified below in Table 1 and Table 2 [4].

Table 1: Spoken Sinhala vowel classification

	Front		Central		Back	
	Short	Long	Short	Long	Short	Long
High	i	i:			u	u:
Mid	e	e:	ə	ə:	o	o:
Low	æ	æ:			a	a:

Table 2: Spoken Sinhala consonant classification[1]

		Lab.	Den.	Alv.	Ret.	Pal.	Vel.	Glo.
Stops	Unvoiced	p	t̪		ʈ		k	
	Voiced	b	d̪		ɖ		g	
Affricates	Unvoiced					ʧ		
	Voiced					ʤ		
Pre-nasalized voiced stops		^{m}b	^{n}d̪		^{n}ɖ		^{ŋ}g	
Nasals		m		n		ɲ	ŋ	
Trill				r				
Lateral				l				
Fricatives	Unvoiced	f	s			ʃ		h
	Voiced	v						
Approximants						j		

There are four nasalized vowels occurring in two or three words in Sinhala. They are /ã/, /ã:/, /æ̃/ and /æ̃:/ [4]. Spoken Sinhala also has following Diphthongs; /æi/, /iu/, /eu/, /æu/, /ou/, /au/, /ui/, /ei/, /oi/ and /ai/.

A separate sign for vowel /ə/ is not provided by the Sinhala writing system. In terms of distribution, the vowel /ə/ does not occur at the beginning of a syllable except in the conjugational variants of verbs formed from the verbal stem “කර” /kərə/ *(to do)*. In contrast to this, though the letter “ඦ”, which symbolizes the pre-nasalized consonant /^{n}ʤ/ exists, it is not considered a phoneme in Sinhala.

2.2. The Sinhala writing system

The Sinhala character set has 18 vowels, and 42 consonants as shown in Table 3.

Table 3: Sinhala character set

Vowels and corresponding vowel modifiers (within brackets): අ ආ(◌ා) ඇ(◌ැ) ඈ(◌ෑ) ඉ(◌ි) ඊ(◌ී) උ(◌ු) ඌ(◌ූ) ඍ(◌ෘ) ඎ(◌ෲ) ඏ(◌ෟ) ඐ(◌ෳ) එ(◌ෙ) ඒ(◌ේ) ඓ(◌ෛ) ඔ(◌ො) ඕ(◌ෝ) ඖ(◌ෞ)
Consonants: ක ඛ ග ඝ ඞ ඟ ච ඡ ජ ඣ ඤ ඦ ට ඨ ඩ ඪ ණ ඬ ත ථ ද ධ න ඳ ප ඵ බ භ ම ඹ ය ර ල ව ශ ෂ ස හ ළ ෆ ◌ං ◌ඃ
Special symbols: ◌්‍ර ◌්‍ය ර්‍ ඥ Inherent vowel remover (Hal marker): ◌්

Sinhala characters are written left to right in horizontal lines. Words are delimited by a space in general. Vowels have corresponding full-character forms when they appear in an absolute initial position of a word. In other positions, they appear as ‘strokes’ and, are used with consonants to denote vowel modifiers. All vowels except “ඎ” /iru:/, are able to occur in word initial positions. The vowel /ə/ and /ə:/ occur only in loan words of English origin. Since there are no special symbols to represent them, frequently the character “ඇ” is used to symbolize them [4].

All consonants occur in word initial position except “◌ං” /ŋ/ *(Anusvaraya)* and the nasals. The symbols “ණ”, and “ළ” represent the retroflex nasal /ɳ/ and the retroflex lateral /ɭ/ respectively. But they are pronounced as their respective alveolar counterparts “න”-/n/ and “ල”-/l/. Similarly, the symbol “ෂ” representing the retroflex sibilant /ʂ/, is pronounced as the palatal sibilant “ශ”-/ʃ/. The corresponding aspirated symbols of letters ක, ග, ච, ජ, ට, ඩ, ත, ද, ප, බ namely ඛ, ඝ, ඡ, ඣ, ඪ, ථ, ධ, ඵ, භ respectively are pronounced like the corresponding un-aspirates [4]. When consonants are combined with /r/ or /j/, special conjunct symbols are used. “ර්”-/r/ immediately following a consonant can be marked by the symbol “◌්‍ර” added to the bottom of the consonant preceding it. Similarly, “ය්”-/j/, immediately following consonant can be marked by the symbol “◌්‍ය” added to the right-hand side of the consonant preceding it [4]. “ඏ” /ilu/ and “ඐ” /ilu:/ do not occur in contemporary Sinhala. Thus, only 40 phonemes are necessary to represent the 60 symbols in the language.

3. Diphone Database Construction

Designing, recording, and labeling a complete diphone database is a laborious and a time consuming

[1] Labial, Dental, Alveolar, Retroflex, Palatal, Velar, Glottal

task. The overall quality of the synthesized speech is entirely dependent on the quality of the diphone database. This section describes the methodology adopted in the construction of Sinhala diphone database.

Prior to constructing the diphone database, the answers to the following two questions were investigated [9]: What diphone-pairs exist in the language? What carrier words should be used? Generally, the number of diphones in a language is roughly the square of the number of phones. Therefore, 40 phonemes for Sinhala identified in section 2.1 suggest roughly 1600 diphones should exist. The first phase involved the preparation of matrices mapping all possible combinations of consonants and vowels; i.e. CV, VC, VV, CC, _V, _C, C_ and, V_. Here '_' denotes a short period of silence. Silence is also considered a phoneme, usually taken at the beginning and ending of the phonemes to match the silences occurring before, between and after words; they are therefore an important unit within the diphone inventory. In the second phase, redundant diphones were marked to be omitted from the recording. Due to various phonotactic constraints, not all phone-phone pairs occur physically (for instance, diphone "^{m}b-^{ŋ}g" never occurs in Sinhala). All such non-existent diphones were identified after consulting a linguist. Finally, 1413 diphones were determined.

The third phase involved in finding the answer to the second question; what carrier words should be used? In other words, to compile set of words each containing an encoded diphone. Following the guidelines given in the Festvox manual [1] it was decided to record nonsense words containing the targeted diphone. These nonsense words were embedded in carrier sentences including four other nonsensical context words. A care was taken when coining these nonsensical words, so that these words act in accordance with phonotactics of the Sinhala language. The diphone is extracted from the middle syllable of the middle word, minimizing the articulatory effects at the start and end of the word. Also, the use of nonsensical words helped the speaker to maintain a neutral prosodic context. The output of the third phase was 1413 sentences.

The fourth phase involved recording the sentences. A native professional male speaker chosen for recording practiced the intelligibility and pronunciation of the sentences. He was advised to maintain a constant pitch, volume, and fairly constant speech rate during the recording. In order to maintain all of the above stated aspects, recordings were limited to two 30-minute sessions per day. At each session, 100 sentences were recorded on average.

Recording was done in a professional studio with an optimum noise free environment. Initially the sentences were directly recorded to Digital Audio Tapes, and later transferred into wave files, redigitising at 44.1 kHz/16 bit quantization.

The most tedious and painstaking tasks were carried out in the fifth phase where the recordings were split into individual files, and diphone boundaries hand-labeled using the speech analysis software tool 'Praat'[2]. Afterwards, a script was written to transform Praat text-grid collection file into diphone index file (EST) as required by Festival [1].

The method for synthesis used in this project is Residual Excited Linear Predictive Coding (RELP Coding). As required by this method, pitch marks, Linear Predictive Coding (LPC) parameters and LPC residual values had to be extracted for each diphone in the diphone database. The script `make_pm_wave` provided by speech tools [1] was used to extract pitch marks from the wave files. Then, the `make_lpc` command was invoked in order to compute LPC coefficients and residuals from the wave files [1]. Having tested synthesizing different diphones, several diphones were identified problematic. An analysis of the errors revealed that most were due to incorrect pitch marking caused by the use of default parameters when extracting the pitch marks. The accurate parameters obtained by analyzing samples of speech with Praat were set in the scripts as per the guidelines given in [7]. Moreover, it was realized that lowering the pitch of the original wave files resulted in a more lucid speech. A proprietary software tool was used to lower the recorded pitch, and normalize it in terms of power so that all diphones had an approximately equivalent power. Subsequently, modified `make_pm_wave` and `make_lpc` scripts were used to extract the necessary parameters from the wave files. These overall post-processing steps significantly improved the voice quality.

The final step was to group the diphone database in order to make it accessible for Festival's UniSyn; the synthesizer module, and to make it ready for distribution. Preparing complete inventory of the diphones took virtually three months. A full listing of the scripts used for recording and creating the diphone database is available for download from http://www.ucsc.cmb.ac.lk/ltrl/si.

[2] Available from: http://www.praat.org

4. Natural Language Processing Modules

When building a new voice using Festvox [1], templates of the natural language processing modules required by Festival are automatically generated as Scheme files. The NLP modules should be customized according to the language requirements. Hence, the language specific scripts (phone, lexicon, tokenization) and speaker specific scripts (duration and intonation) can be externally configured and implemented without recompiling the system [1], [9]. The NLP related tasks involved when building a new voice are [1], [5]:

- Defining the phone-set of the language
- Tokenization and text normalization
- Incorporation of letter-to-sound rules
- Incorporation of syllabification rules
- Assignment of stress patterns to the syllables in the word
- Phrase breaking
- Assignment of duration to phones
- Generation of f0 contour.

4.1. The phone set definition

The identified phone-set for Sinhala in section 2.1 is implemented in the file festvox/ucsc_sin_sdn_phoneset.scm. This module defines the phones and describes their features. The proposed set of symbol scheme is found to be a versatile representation scheme for Sinhala phone-set. Along with the phone symbols, features such as vowel height, place of articulation and voicing are defined. Apart from the default set of features, new features that are useful in describing Sinhala phones are also defined. E.g. whether a consonant is pre-nasalized or not. These features will prove extremely useful when implementing prosody.

4.2. Tokenization and text normalization

The default text tokenization methodology implemented in Festival (which is based on white-space, and punctuation characters) is used to tokenize Sinhala text. Once the text has been tokenized, text normalization is carried out. This step converts digits, numerals, abbreviations, and non-alphabetic characters into word sequence depending on the context. Text normalization is a non trivial task. Therefore, prior to implementation, it was decided to analyze running text obtained from a corpus. Text obtained from the category "*News Paper > Feature Articles > Other*" of the UCSC Sinhala corpus BETA was chosen due to the heterogeneous nature of these texts and hence better representation of the language in this section of the corpus[3]. A script was written to extract sentences containing digits from the text corpus. The issues were identified by thoroughly analyzing the sentence. Strategies to address these issues were devised. A function is implemented to convert any number (decimal or integer up to 1 billion) into spoken words.

In Sinhala, the conversion of common numbers is probably more complicated when compared to English. In certain numerical expressions, the number may be concluded from a word suffix. e.g. 5න් pahen *(out of five)*, 1 වැනි paləvæni *(1st)*. Such expressions are needed to be identified by taking into consideration the added suffix in a post-processing module. A function is implemented to expand abbreviations into full words. Common abbreviations found by the corpus analysis are listed, but our architecture allows easy incorporation of new abbreviations and corresponding words. In some situations, the word order had to be changed. For example, 50% must be expanded as "සියයට පනහ" - si:jəjaʈə panəha *(percent hundred)*, 50m should be expanded as මීටර් පනහ - mi:ʈər panəha *(meters fifty)*. All above mentioned functions are called effectively by analyzing the context, and then accurate expansions are obtained.

The tokenization and text normalization modules are implemented in festvox/ucsc_sin_sdn_tokenizer.scm and capable of normalizing elements such as numbers, currency symbols, ratios, percentages, abbreviations, Roman numerals, time expressions, number ranges, telephone numbers, email addresses, English letters and various other symbols.

4.3. Letter-to-sound conversion

The letter-to-sound module is used to convert an orthographic text into its corresponding phonetic representation. Sinhala being a phonetic language has an almost one-to-one mapping between letters and phonemes.

We implemented the grapheme to phoneme (G2P) conversion architecture proposed by Wasala et

[3] This accounts for almost two-thirds of the size of this version of the corpus

al. in [10]. In this architecture, the UTF-8 textual input is converted to ASCII based phonetic representation defined in the Festival. This process takes place at the user-interface level. Owing to the considerable delay experienced when synthesizing the text, it was decided to re-write the above G2P rules in the Festival's context sensitive format [1]. The rules were re-written in UTF-8 multi-byte format following the work done for Telugu language [3]. The method was proven to work well causing no delay at all. The 8 rules proposed in [10] expanded up to 817 rules when re-written in context sensitive format. However, some frequently encountered important words were found incorrectly phonetized by these rules. Hence, such words along with their correct pronunciation forms were included in Festival's addenda, a part lexicon. The letter-to-sound rules and lexicon are implemented in festvox/ucsc_sin_lexi.scm.

Festival's UTF-8 support is still incomplete; however, we believe the above architecture as the best to deal with Unicode text input in Festival over other proposed methods [12], [10].

4.4. Syllabification & stress assignment

Instead of Festival's default syllabification function `lex.syllabify.phstress` based on sonority sequencing profile [1], a new function `(syllabify 'phones)` is implemented to syllabify Sinhala words. In this work, syllabification algorithm proposed by Weerasinghe et al. [11] is implemented. This algorithm is reported to have 99.95% accuracy [11]. The syllabification module is implemented in festvox/ucsc_sin_sdn_syl.scm.

4.5. Phrase breaking algorithm

The assignment of intonational phrase breaks to the utterances to be spoken is an important task in a text-to-speech system. The presence of phrase breaks in the proper positions of an utterance affects the meaning, naturalness and intelligibility of speech. There are two methods for predicting phrase breaks in Festival. The first is to define a Classification and Regression Tree (CART). The second and more elaborate method of phrase break prediction is to implement a probabilistic model using probabilities of a break after a word based on the part of speech of the neighboring words and the previous word [1]. However, due to the unavailability of a Part-of-Speech (POS), and a POS tagger for Sinhala, probabilistic model cannot be constructed yet. Thus, we opted for the simple CART based phrase breaking algorithm described in [1]. The algorithm is based on the assumption that phrase boundaries are more likely between content words and function words. A rule is defined to predict a break if the current word is a content word and the next is seemingly a function word and the current word is more than 5 words from a punctuation symbol.

This algorithm, initially developed for English, has proved to produce reasonable results for Sinhala as well. The phrasing algorithm is defined in festvox/ucsc_sin_sdn_phrase.scm.

4.6. Prosodic analysis

Prosodic analysis is minimal in the current system and will be implemented in the future. The major challenge for building prosody for Sinhala is the lack of a POS tag-set, POS tagger and tagged text corpus. An experiment was carried out to adapt CART trees generated for an English voice prosody (f0 & duration) modules into Sinhala. The CART trees were carefully modified to represent the Sinhala Phone-set. The phone duration values were also hand modified to incorporate natural phone durations. The above steps resulted in more natural speech when compared to the monotonous speech produced before incorporating them. These adapted modules (cmu_us_kal_dur.scm, cmu_us_kal_int.scm) are incorporated to the Festival-si system.

5. Integration with Different Platforms

Festival offers a powerful platform for the development and deployment of speech synthesis systems. Since most Linux distributions now come with Festival pre-installed, the integration of Sinhala voice in such platforms is very convenient. Furthermore, following the work done for Festival-te, the Festival Telugu voice [3], the Sinhala voice developed here was made accessible to GNOME-Orca[4] and Gnopernicus[5] - powerful assistive screen reader software for people with visual impairments.

The next task involved the building of Festival support natively on Windows. It is noteworthy to mention the modification to Festival's text reading module as part of this task. We experienced that Festival is incapable of reading UTF-8 text files with byte-order marker (BOM). UTF-8 text files saved by Windows Notepad or OpenOffice Word will always include the byte-order-marker. Festival fails to read such files and will give an error "Un-pronunciation

[4] Available from: http://live.gnome.org/Orca

[5] Available from: http://www.baum.ro/gnopernicus.html

word". Manual removal of BOM from file each time proved to be a repetitive process. Therefore, a short patch was written to make Festival capable of reading text files with UTF-8 byte-order-marker. The patch is available for download from http://www.ucsc.cmb.ac.lk/ltrl/si.

Motivated by the work carried out in the Welsh & Irish Speech Processing Resources (WISPR) project [12], steps were taken to integrate Festival along with the Sinhala voice into the Microsoft Speech Application Programming Interface (MS-SAPI) which provides the standard speech synthesis and speech recognition interface within Windows applications [13]. As a result of this work, the MS-SAPI compliant Sinhala voice is accessible via any speech enabled Windows application. We believe that the visually impaired community would be benefited at large by this exercise owing to the prevalent use of Windows in the community. The Sinhala voice also proved to work well with Thunder[6] a freely available screen reader for Windows. This will cater to the vast demand for a screen reader capable of speaking Sinhala text. It is noteworthy to mention that for the first time the print disabled community in Sri Lanka will be able to work on computers in their local language by using the current Sinhala text-to-speech system.

6. Evaluation

Text-to-speech systems can be compared and evaluated with respect to intelligibility, naturalness, and suitability for used application [6]. As the Sinhala TTS system is a general-purpose synthesizer, a decision was made to evaluate it under the intelligibility criterion.

A Modified Rhyme Test (MRT) [6], [9] was designed to test the Sinhala TTS system. The test consists of 50 sets of 6 one or two syllable words which makes a total set of 300 words. The words are chosen to evaluate phonetic characteristics such as voicing, nasality, sibilation, and consonant germination. Out of 50 sets, 20 sets were selected for each listener. The set of 6 words is played one at the time and the listener marks the synthesized word. The overall intelligibility of the system from 20 listeners is found to be 71.5%. This is the first known documented work on a Sinhala text-to-speech synthesizer, and also the first Sinhala TTS system, which had been evaluated.

7. Conclusions and Future Work

In this paper we described the development and evaluation of the first TTS system for Sinhala language based on the Festival architecture. The design of a diphone database and the natural language processing modules developed has been described.

Future work will mainly focus on improving the naturalness of the synthesizer. Work is in progress to improve the prosody modules. A speech corpus containing 2 hours of speech has been already recorded. The material is currently being segmented, and labeled. We are also planning to improve the duration model using the data obtained from the annotated speech corpus. A number of other ongoing projects are aimed at developing a POS tag set, POS tagger and a tagged corpus for Sinhala. Further work will focus on expanding the pronunciation lexicon. At present, the G2P rules are incapable of providing accurate pronunciation for most compound words. Thus, we are planning to construct a lexicon consisting of compound words along with common high frequency words found in our Sinhala text corpus, which are currently incorrectly phonetized.

All resources developed under this project are made available at: http://www.ucsc.cmb.ac.lk/ltrl/si.

8. Acknowledgements

This work was made possible through the PAN Localization Project, (http://www.PANL10n.net) a grant from the International Development Research Center (IDRC), Ottawa, Canada, administered through the Center for Research in Urdu Language Processing, National University of Computer and Emerging Sciences, Pakistan.

9. References

[1] A.W. Black, and K.A. Lenzo, *Building Synthetic Voices*, Language Technologies Institute, Carnegie Mellon University and Cepstral LLC. Retrieved from: http://festvox.org/bsv/, 2003.

[2] R.J. Jones, A. Choy and B. Williams, "Integrating Festival and Windows", *InterSpeech 2006*, *9th International Conference on Spoken Language Processing*, Pittsburgh, USA, 2006.

[3] C. Kamisetty and S.M. Adapa, *Telugu Festival Text-to-Speech System*. Retrieved from: http://festival-te.sourceforge.net/wiki/Main_Page, 2006.

[6] Available from: http://www.screenreader.net/

[4] W.S. Karunatillake, *An Introduction to Spoken Sinhala*, 3rd edn., M.D. Gunasena & Co. ltd., 217, Olcott Mawatha, Colombo 11, 2004.

[5] S.P. Kishore, R. Sangal and M. Srinivas, "Building Hindi and Telugu Voices using Festvox", *Proceedings of the International Conference On Natutal Language Processing 2002 (ICON-2002)*, Mumbai, India, 2002.

[6] S. Lemmetty, *Review of Speech Synthesis Technology*, MSc. thesis, Helsinki University of Technology, 1999.

[7] A. Louw, *A Short Guide to Pitch-Marking in the Festival Speech Synthesis System and Recommendations for improvement*. Local Language Speech Technology Initiative (LLSTI) Reports. Retrieved from: http://www.llsti.org/documents.htm, (n.d.).

[8] A.G. Ramakishnan, K. Bali, P.P. Talukdar and N.S. Krishna, "Tools for the Development of a Hindi Speech Synthesis System", *In 5th ISCA Speech Synthesis Workshop*, Pittsburgh, 2004, pp. 109-114.

[9] Ö. Salor, B. Pellom and M. Demirekler, "Implementation and Evaluation of a Text-to-Speech Synthesis System for Turkish", *Proceedings of Eurospeech-Interspeech 2003*, Geneva, Switzerland, 2003, pp. 1573-1576.

[10] A. Wasala, R. Weerasinghe and K. Gamage, "Sinhala Grapheme-to-Phoneme Conversion and Rules for Schwa epenthesis", *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, Sydney, Australia, 2006, pp. 890-897.

[11] R. Weerasinghe, A. Wasala, and K. Gamage, "A Rule Based Syllabification Algorithm for Sinhala", *Proceedings of 2nd International Joint Conference on Natural Language Processing (IJCNLP-05)*, Jeju Island, Korea, 2005, 438-449.

[12] B. Williams, R.J. Jones and I. Uemlianin, "Tools and Resources for Speech Synthesis Arising from a Welsh TTS Project", *Fifth Language Resources and Evaluation Conference (LREC)*, Genoa, Italy, 2006.

[13] Microsoft Corporation.: *Microsoft Speech SDK Version 5.1*, Retrieved from: http://msdn2.microsoft.com/en-s/library/ms990097.aspx, (n.d.).