Project 10c
Implementation of Hindi Vocalizer on Cell Broadband Engine

January 27, 2013

GOAL DOCUMENT

Name	**Roll No**	**Contact No**	**Email**
Gaurangi Anand	MT2012046	9483001973	Anand.Gaurangi@iiitb.org
Jayati Deshmukh	MT2012056	9916747257	Jayati.Deshmukh@iiitb.org
N Puneeth	MT2012083	8867909775	Puneeth.N@iiitb.org
Pushpendra Sinha	MT2012107	8095316684	Pushpendra.Sinha@iiitb.org
Sindhu Priyadarshini	MT2012134	9916660667	Sindhu.Priyadarshini@iiitb.org

Team Leader - N Puneeth (MT2012083)

Guide : Prof. Shrisha Rao

Operating Systems (CS110)

M. Tech 2012 Batch

Contents

List of Figures

1 Introduction

Objective

The project aims at developing a Text-to-Speech (TTS) Vocalizer application in an Indian Language, which we have chosen as Hindi. Further this system is to be deployed on the IBM CBE architecture.

Background

Hindi, the official language of India, is spoken as a first language by 33 percent of the Indian population, and by many more as a *lingua franca*. In contrast, only a very small percentage of Indians use English as a means of communication. Coupled with the prevalent low literacy rates makes the use of conventional user interfaces difficult in India. Spoken language interfaces enabled with TTS synthesis have the potential to make information and other Information and Communication Technology (ICT) based services accessible to a large proportion of the population. Text-to-speech system is the most widely used system in speech technology. We have various TTS synthesizer systems available like Festival, Multilingual and Flite etc. A Text-To-Speech system is a computer-based system that should be able to read any text aloud, whether it was directly introduced in the computer by an operator or scanned and submitted to an Optical Character Recognition (OCR) system. Speech synthesis is a process where verbal communication is replicated through an artificial device.

The Hindi language is based on the Devanagari script. It is a combination of around 13 vowels and 40 consonants. The whole of the text is composed of various tokens that are processed and corresponding pseudo-human voice is generated.

Cell is an architecture for the microprocessor that is the product of the joint venture of an alliance called “STI” i.e. Sony, Toshiba and IBM. These cells have a cellular architecture with 9 cores, that are new in the market and have the power of parallel computing at the thread level. This brings in a great deal of speedup and marks an edge over the traditional desktop processors, mostly the Core 2 and Athlon 64 families in terms of performance.

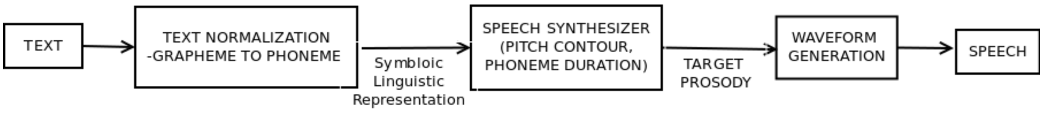
TEXT
TEXT NORMALIZATION -GRAPHEME TO PHONEME
Symbloic Linguistic Representation
SPEECH SYNTHESIZER (PITCH CONTOUR, PHONEME DURATION)
TARGET PROSODY
WAVEFORM GENERATION
SPEECH

Figure 1: A Text-to-Speech System

Vocalizer

A vocalizer is a TTS system that has to be sophisticated enough to recognize the piece of text as accurate as possible based on the Unicode standard. A TTS is a system that analysis the text, processes it, generates speech for corresponding words processed and produces voice as output as shown in Figure 1. The Natural Language Processing (NLP) Engine is robust to generate the sound for the words interpreted as they are meant to be pronounced. The database used for recognition is vast and includes all possible characters used in the Hindi language.

The goal of TTS is the automatic conversion of unrestricted natural language sentences in text form to a spoken form that closely resembles the spoken form of the same text by a native speaker of the language. This field of speech research has witnessed significant advances over the past decade with many systems being able to generate a close to natural sounding synthetic speech. Research in the area of speech synthesis has been fuelled by the growing importance of many new applications. These include information retrieval services over telephone such as banking, public announcements at places like train stations, bus stands and reading out of manuscripts [1] Speech synthesis can also be applied in tools for reading emails, faxes and web pages over telephone and voice output in automatic translation systems or to improve the accessibility.

Cell Broadband Engine

Cell is a heterogeneous multi-core processor comprised of control-intensive processor and compute-intensive SIMD processor cores, each with its own distinguishing features. The Cell consists of one control-intensive Power Processing Element (PPE) and eight compute-intensive Synergistic Processing Element (SPEs) [2, 3] as shown in Figure 2. A high-speed bus called the Element Interconnect Bus (EIB) is used for connecting these processor cores within the Cell. The EIB also provides connections to main memory and external I/O devices, making it possible for processor cores to access data.

Basically cell has 2 types of cores :

- **PowerPC Processor Element (PPE):** The PPE implemented in the Cell is a general-purpose processor with functions equivalent to those offered by the 64-bit PowerPC architecture. PPE allows execution of the operating system and applications. It also performs input/output control when the operating system accesses the main memory and external devices, and provides control over SPEs. Accordingly, the PPE can be designed as a control-intensive processor dedicated mainly to processing control.

- **Synergistic Processor Element (SPE):** The Cell incorporates eight processor cores called the SPEs. The SPEs are less complex processing units than the PPE as they are not designed to perform control-intensive tasks. The SPEs iterates simple operations necessary for processing multimedia data. The Cell delivers an exceptional computational capability by selectively using these computationally intensive processor cores. The eight SPEs are the primary computing engines on the Cell processor. Each SPE contains a Synergistic Processing Unit (SPU), a memory flow controller, a memory management unit, a bus interface and an atomic unit for synchronization mechanisms. To make full use of all the computational power available on the Cell BE processor, data must be distributed and communicated between PPE and SPEs. The PPE interacts with the SPEs through Memory-Mapped Input/Output (MMIO) registers supported by the Memory Flow Controller (MFC) of each SPE. PPE is often used as an application manager, assigning and directing work to the SPEs. A large part of this task is loading main storage with the data to be processed, and then notifying the SPE.

2 Gap Analysis

Good quality Hindi or any local language vocalizer system that can be used for practical application are presently not available. Though a number of prototypes of Indian language TTS systems have been developed [4, 5], none of these can be compared to systems in languages like English, German and French which has attracted a lot of research and development. The main reason for this is that developing a TTS system in any new language requires close collaboration between linguists and technologists for solving many language specific issues. Significant amount of annotated data is required for

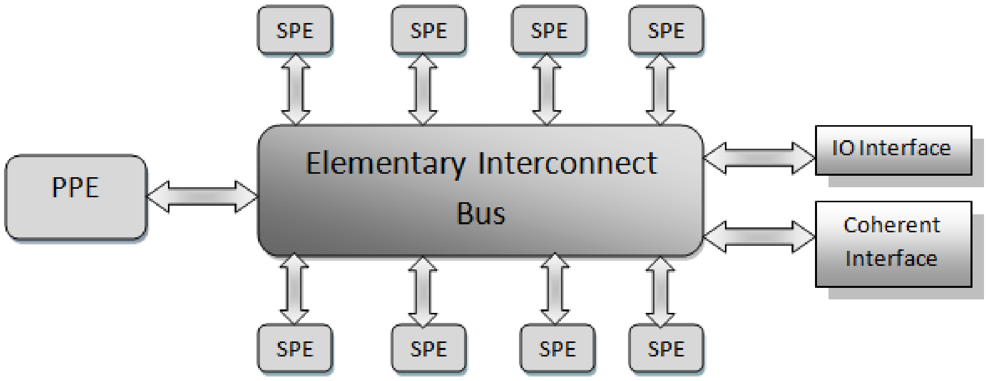

Figure 2: Basic CBE Architecture

developing language-processing applications which involves many parts such as speech (POS) taggers, syntactic parsers, intonation and duration models. This is a resource intensive task, which requires significant institutional support which has been traditionally lacking in developing countries like India. We propose to implement a TTS system in Hindi with the help of the festival framework with a few alterations specific to an indic language. Further we plan to implement it on a parallel architecture provided by IBM's Cell Broadband Engine so that we can parallelize some of the tasks.

3 System Architecture

The architecture of the vocalizer to be developed is depicted in Figure 3. The vocalizer will be developed using the Festival framework which provides an open architecture for research in speech synthesis, which will then be implemented on CBE [6].

- First step is the capture of the input text from the user, it can be done via a text box or through a text file.
- Next step is the standardization of the input text, which needs to be in Unicode. Appropriate conversion is done at this stage. This is a one time step and is the first process to be done on the CBE. This standardized text is then fed into the text processing module.
- **Sentence Segmentation** : In this module the input is partitioned into sentences using the sentence end markers of Hindi language. This

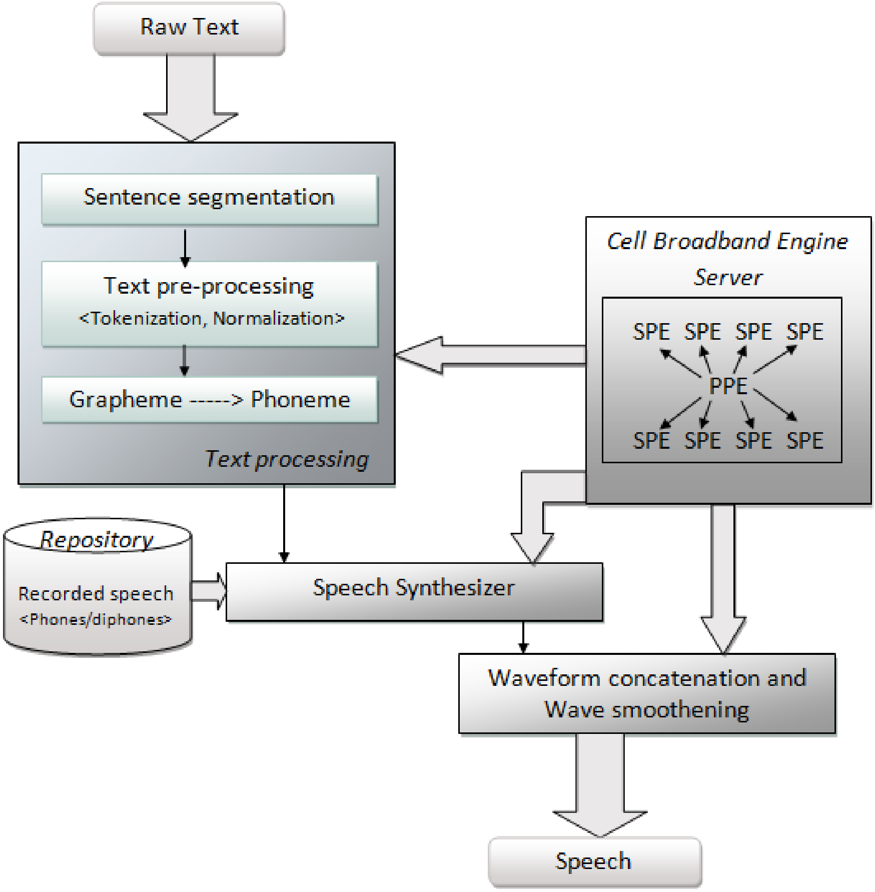

Figure 3: System Architecture

process is known as sentence segmentation. The PPE can direct segments to one or more of the eight SPEs simultaneously if the user input is large. In this way we can handle large data instances without significant time delay.

- **Text Processing** :
  - This module performs tokenization, normalization and grapheme to phoneme conversion.
  - Tokenization and normalization can be termed as text preprocessing. In the Tokenization step the PPE instructs the SPEs to split the input text at white spaces and at punctuation marks (except end markers) by making use of regular grammar.
  - In the Normalization module the PPE instructs the SPEs to further break down the words generated after Tokenization step into its constituent grapheme.
  - Grapheme to phoneme conversion is a data driven approach. In this module the tokens, now called graphemes, are mapped to their corresponding phonemes. This process uses either dictionary-based or rule-based strategies. In the end we get the whole input text in the form of phonemes.
- **Speech Synthesizer :** In this module the PPE instructs the SPEs to map the phonemes obtained from the text processing module onto their corresponding stored speech units contained in the speech repository in the form of phones or diphones.
- **Waveform generation and Concatenation :** In this module the output of speech synthesizer is modulated using waveform generation and waveform smoothening which gives a voice output. In waveform concatenation the segments of recorded speech are concatenated together by the SPEs. In waveform smoothening, the noise incurred in the data present as short term variations are removed.
- The different words are generated at various SPEs are then combined by the PPE and speech in the pseudo-human voice is generated.

System Requirements

- **Hardware Requirement :** Laptop/PC with 1 GB RAM and 25 GB Hard disk space for optimum performance and A Sony PS3 with CBE architecture.

- **Software Requirement :** Operating System(s) - Ubuntu 12.04 (or higher versions) with gcc/g++ compiler, IBM CBE SDK and Festival Framework.

4 Development Plan

Phase 1: Initialization

1. Installation of Cell Broadband Engine SDK and Festival.
2. Begin to create the Database.
3. Implementation of the Text capture module.

Phase 2: Text Processing

1. Implementation of a sentence segmentation in PPE unit of CBE.
2. Implementation of the Tokenizer module on CBE. It is a part of the text preprocessing step which splits the input text at white spaces and punctuation marks.
3. Implementation of a Normalization module on CBE, which makes the text more consistent for further processing.
4. Conversion of Graphemes to Phonemes on CBE. This maps the processed text to its corresponding prosodic units.

Phase 3: Speech synthesis

1. Completion and Consolidation of database containing pieces of recorded speech, including annotation, which was initiated in Phase 1.2.
2. Training and Implementation of Speech synthesizer using the speech database. This includes the generation of waveforms from the symbolic linguistic representation of the text.

Phase 4: Testing

1. Unit Testing of various modules.
2. Integration of modules and its testing.
3. Test the System with diverse set of input.

5 Milestones

- January 18: First draft of the goal statement.
- January 28: Final draft of goal statement submitted and setting up of CVS/SVN/Git version control.
- January 30: Completion of Phase 1.1, Initiate Phase 1.2 and 1.3.
- February 4: Completion of Phase 1.3.
- February 5: Brief presentation of project architecture. Completion of Phase 2.1
- February 15: Completion of Phase 2.2.
- February 20: Completion of Phase 2.3.
- February 25: Completion of Phase 2.4
- March 4: Completion of Phases 1.2 and 3.1 and Detailed mid-term progress review.
- March 10: Completion of Phase 3.2.
- March 15: Begin of Phase 4.1.
- March 20: Completion of Phase 4.1. Begin phase 4.2.
- March 27: Completion of Phase 4.2. Begin phase 4.3.
- April 4: Completion of Phase 4.3. and release of Beta version.
- April 10: Project Enhancement.
- April 18: Final submission of the project and the project report.

6 Glossary

1. **Unicode:** Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language.

2. **Festival Framework:** Festival offers a general framework for building speech synthesis systems as well as including examples of various modules.

3. **Phoneme:** A phoneme is the smallest contrastive unit in the sound system of a language.

4. **Grapheme:** The smallest meaningful contrastive unit in a writing system.

5. **Prosody:** The rhythm, stress, and intonation of speech.

7 References

[1] Samuel Thomas, “Natural sounding speech synthesis based on syllable-like units”, M.S. thesis, Department of Computer Science and Engineering, IIT Madras, May, 2007.

[2] *Introduction to Cell Broadband Engine Architecture*, version 1.02. Sony Computer Entertainment, October, 2007.

[3] Yang Song, Gregory M. Striemer, Ali Akoglu, “Performance Analysis of IBM Cell Broadband Engine on Sequence Alignment,” ahs, pp.439-446, *2009 NASA/ESA Conference on Adaptive Hardware and Systems*, 2009

[4] S.P. Kishore, Rohit Kumar, and Rajeev Sangal,“A Data-Driven Synthesis approach for Indian Languages using Syllable as Basic Unit”, *International Conference on Natural Language Processing (ICON)*, 2002, pp. 311 to 316.

[5] J. Rama, A.G. Ramakrishnan, and R. Muralishankar, “A Complete TTS system in Tamil”, *IEEE Workshop on Speech Synthesis*, 2002.

[6] Hartmut R. Pfitzinger Uwe D. Reichel. “Text preprocessing for speech synthesis”, *TC-STAR Workshop on Speech-to-Speech Translation*, 2006