**Introduction**

Usage of native language to access digital technology has always been a great challenge, especially for third world countries. People had tried to overcome this challenge by localizing the User Interface to the local language. This has greatly assisted non-native English speakers to use Computer Softwares, but the Software scenario has greatly changed and people have been shifting towards newer ways to interact with computers. One of the ways that people feel easy to interact with the system is via voice. With services like Siri, Cortana and Google Now on the pop culture, it will still be a challenge for non-native English speakers to adapt to those systems.

Above mentioned system use Voice Recognition at its core. Internally, user's voice is translated into English text. The text then can be parsed utilizing existing programs and applied to various uses. Therefore, in order to nourish the development of Nepali Voiced based system a minimum library to translate Nepali Speech to Text is required. Thus, we plan to create a Speech to Text System for Nepali language.

**Objectives**

The objectives of this project can be summarized below:

1. To enhance the development of Nepali Voice based apps by providing a Nepali Voice Recognition library
2. To create a platform for Nepali Voice to Text Transcription (Example: Nepali Dictation)

**Scope**

After the completion of this project, we will create a library that can be consumed to create different applications. Creating this library will open many windows of opportunities to create applications based on the library. We will also extend the library to create a Nepali Speech to Text Converter. But besides that, the end project can be used to create applications like:

1. Nepali Voice Command Control for PCs
2. Mix it with Nepali Voice Synthesizer to Create Personal Assistants
3. The Speech to Text Converter can be applied to wide areas of interests like in Chat Applications, System Control Applications.
4. The Speech to Text Converter system can also be useful for disabled to interact with the system. Screen Reader is not enough.

**Literature Review**

In the past, lots have been done in the field of speech recognition, but the most popular ones are done for English language. A thorough knowledge of the Linguistics is required for speech recognition and less have been done for Nepali. We can find some programs like Nepali TTS closely related to ASR, but there have been no any known work in developing a Nepali speech recognition system. Since Hindi and Nepali share common ancestor, i.e., Sanskrit, the closest match for similar work would be Hindi ASR. But since the engine utilizes common technologies, Nepali ASR is adaptable from English ASR replacing their linguistics with Nepali.

There are plenty of papers one can review and grasp the concept of the underlying technology inside ASR. Out of which, we have managed to skim some of these writings.

The widely accepted way to design an ASR is by using Spectral Analysis and HMMs. We will also follow the same principle since finding a new way would be a research on its own. We can classify ASR into two variants: Directed Dialogs and Natural Language Conversation. Directed Dialogs only has to look for specific speech so it's simpler. However, it does not give much flexibility that we require. The other variant will allow us to create a library that can recognize almost any sentences from Nepali language, so in order to widen our scope of the project; we have decided to go to Natural Language Conversation as our ASR variant to work on.

There are also multiple methods to train or tune our ASR system. The simple way, also referred to as "Human Tuning" is a manual process where developers take care of tuning the system. Another variant to this is Active Learning – commonly used for applications where the system has to personalize to the user who's using it. We will choose Human Tuning for the sake of simplicity.

Having pointed out the similarities, we also need to think about the distinctions between what has already been done and widely accepted to what we propose. There are a couple of differences, majority of which lie in the linguistics part of the system.

Every ASR needs to have a proper linguistic background. It is backed by the atomic unit of speech – called Phonemes. Phonemes differ according to language. English language has 44 Phonemes in its speech, which is different in Nepali. We are looking at Nepali Linguistic Papers in order to grasp some basic ideas and integrate them with our project.

We can name quite a few already available speech recognition engines that are currently being used for lots of other languages. If we were to use such tools, then all we have to do is to feed audio clips and corresponding texts so as to train those engines. Some examples of widely used engines/toolkits are **cmusphinx, htk, kaldi,** etc. We have widely learnt from cmusphinx and their developers, so the project might have some similarities with that engine.

**Methodology**

The task of speech recognition can be broken down into various processes. An overview can be seen in the following chart.

Figure 5.1: Overview of Automated Speech Recognition Engine

Record Speech

(Sample at 8KHz)

Preprocess and Frame

Extract Features

Run Through Acoustic Model

Run Through Language Model

Output the Recognized Text

Each text in the above figure is described below:

**Record Speech**

The first and foremost part of any automated speech recognition task is to actually record the voice data from users. Most of the recorders today have a sampling frequency of 44.1 KHz, a widely accepted frequency since it is enough to capture the music composed by sound creators. However, human speech has relatively low bandwidth ranging from 100 Hz to 8 KHz. Therefore, it is safe to use a sampling frequency of 8 KHz for a speech recognition task and save on the processing power for later.

**Preprocessing and Framing**

Another major task after recording the speech is to process the given discrete signal, and segment it into frames of N samples each. At first, the signal will be passed through a digital filter that will emphasize the higher frequency signals. After that, in the segmentation process, overlapping the frames will allow us to have resulting feature vector to be smoother across adjacent frames, but will demand higher processing power. So, the frames will be overlapped for a reasonable area to balance the tradeoff between processing power and smoother change in feature vector. A Hamming window with α = 0.54 and β = 1 – α = 0.46 is considered suitable window function after framing the signal. The signal is then de-noised, or noise reduction algorithms are run through the signal. The noise reduction algorithms can be widely classified into three major categories *viz.* **Filtering, Spectral Restoration, and Speech-Model Based** algorithms, each of which has different applications. Speech-Model Based algorithm is an interesting but a bit complex algorithm to work on, while Filtering is not that interesting algorithm to use in speech recognition scenario. A Balance would be Spectral Restoration which induces the missing spectral components of non verbal sounds by adding noise to increase intelligibility.

After pre-processing, framing and de-noising, each frame's feature vector is calculated. This process can be summarized as:

Figure 5.2.1: Signal Preprocessing and Framing

Emphasize Signal

Segment and Window Signal

De-noise Signal

To feature Extraction

**Feature Extraction**

**Project Breakdown**

Since the method we chose is scrum, it is not possible to create a Gantt chart before we start the project itself. However, since scrum also emphasizes on Product Backlogs, we are still able to break down the project (which will change over time, as we learn further) and document what part took how much time. Thus, a Gantt chart can be published after some work has been done and the initial Project Breakdown looks like this:

1. Learn and Implement Signal Modeling
2. Learn and Perform Spectral Analysis
3. Learn about Nepali Linguistics, Phoneme categorization
4. Learn about Phoneme Recognition Process
5. Learn about usage of Hidden Markov Models in predicting Phoneme Sequence
6. Start developing the system from base.

**References**

J. P. Hosom *et al.,* "Speech Recognition Using Neural Networks," Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology., Beaverton, OR, Jul. 06, 1999.

J. Picone, "FUNDAMETALS OF SPEECH RECOGNITION," Institute for Signal and Information Processing, Mississippi State University, May 1996.

S.P. Srivastava, "Nepali in West Bengal," India. Govt. of India. 2011, ch. 5 pp. 157-237

M. Zajechowski. (2014, Dec. 29). *Automatic Speech Recognition (ASR) Software – An Introduction* [Online]. Available: http://usabilitygeek.com/automatic-speech-recognition-asr-software-an-introduction/

D. József, T. Gavril, "Continuous Speech Phoneme Recognition Using Dynamic Artificial Neural Networks" in *Automation Computers Applied Mathematics*, Communication Department, Technical University of Cluj-Napoca, 2008, vol. 17 pp. 5-11

A. Waibel *et al., Phoneme Recognition: Neural Networks vs. Hidden Markov Models.* [Online]. Available: http://isl.anthropomatik.kit.edu/cmu-kit/downloads/ Phoneme\_Recognition\_Neural\_Networks.pdf