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.

There are two kinds of ASR for human voice – isolated and connected words recognition. While isolated is a bit easier than connected, they share the same basics. Also, there are as many scopes for isolated word recognition when compared to connected words. So, we will be first focusing towards isolated speech recognition and slowly move towards recognizing connected words.

Methodology

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

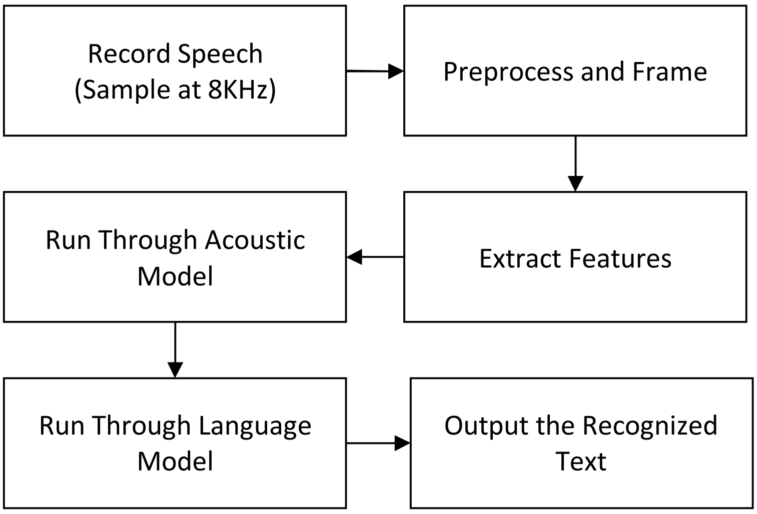
Record Speech (Sample at 8KHz)
Preprocess and Frame
Extract Features
Run Through Acoustic Model
Run Through Language Model
Output the Recognized Text

Figure 5.1: Overview of Automated Speech Recognition Engine

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:

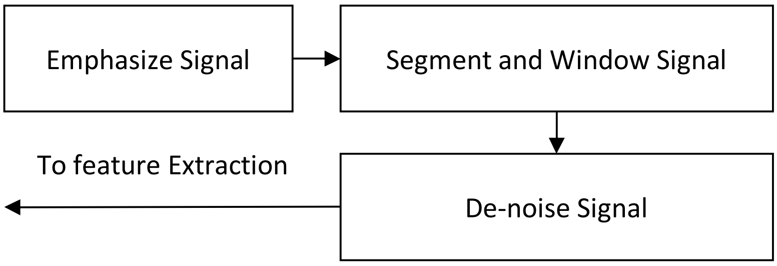

Figure 5.2.1: Signal Preprocessing and Framing

Feature Extraction

The signal, after preprocessing, segmenting, and enhancing contains a lot of information which are not necessarily related to linguistics. These extra information might be speaker dependent characteristics, characteristics of environment, or recording equipment. Since the goal of Automated Speech Recognition engine is to transcribe the linguistic message, feature extraction focuses on suppressing all other irrelevant information from the signal. A Feature Extraction algorithm could be viewed as an algorithm that derives a characteristics feature vector with lower dimensionality than that of the original signal.

There are many ways through which feature extraction is made possible. These can be broadly classified into two classes, *viz.* calculating coefficients based on particular transformations and learning method based hidden features.

In the first class, there are two popular techniques. First of which is calculating the Mel-Frequency Cepstral Coefficients (MFCC), an widely used techniques for feature extraction in automated speech recognition tasks but with considerably higher computational requirement than its counterpart which is wavelet based feature extraction using wavelet transformation (WT). However, it is seen that MFCCs are better suited to speech recognition and WTs are better for classification problems in audio signals.

Learning based algorithms such as non negative matrix factorization (NMF) and artificial neural network (ANN) based classifiers take significant amount of computational time in comparison to MFCC or WT. Although suited for applications like speech separation and robust feature detection, many have combined these algorithms with each other to get better accuracy. For the sake of simplicity, and since MFCC in average tend to give considerable accuracy, we will stick to the use of MFCC in our project.

We will now dive deeper into the implementation details of MFCC in our application. First, let's have a look on algorithmic overview of MFCC.

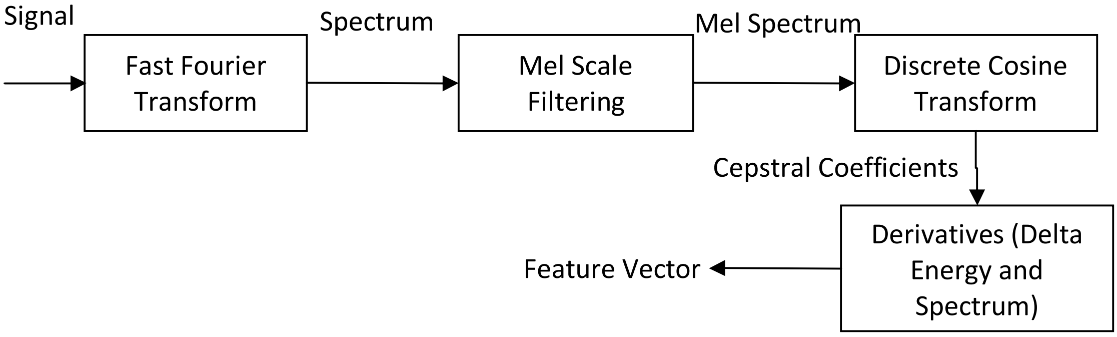

Figure 5.3.1: MFCC Algorithm

After windowing the signal, the first process is to transform the given time-domain signal to frequency spectrum using the application of Discrete Fourier Transform (DFT). In implementations, we can use one of many Fast Fourier Transformation algorithms in order to speed up calculations.

After transforming the signal in time-domain to a frequency spectrum, a series of band pass filters are applied to the spectrum and power of each band is calculated. The powers are mapped into Mel scale using standard formula (See Appendix). In ASR, the normal linear scale cannot be used because it cannot mimic human ear which might use any frequency as center frequency. So a series of equidistant band-pass filter in Mel scale is used in ASR systems. At the end of this process we would have obtained a vector of power of different bands. We will then convert these powers to a logarithmic scale simply by taking element-wise logarithms to the power vector.

The corresponding logarithmic Mel spectrum is now changed back into time domain using Discrete Cosine Transform (DCT), and the vector is now termed as Cepstral Coefficients.

Upto now, we have only been able to extract the needed features for the static characteristic of a segmented voice signal. But human speech is dynamic in nature, so to compensate that, the first and second degree derivatives of the Cepstral Coefficients are taken which extends our feature vector. These derivatives are often termed as delta and double delta coefficients. The collection of Cepstral, Delta, and Double Delta Coefficients creates the complete feature vector for our segmented voice signal – often termed as acoustic vectors.

The Acoustic Model

At this point, we have our feature vector which we will say ***X***. We now seek to find the sequence of words that the feature vector can generate. The Acoustic model will contain Hidden Markov Models for different words which will be fed to Viterbi Algorithm with the phoneme sequence obtained from feature vectors. With the combination of those, we will be able to predict the possible N-words related with the given phoneme sequence. We will pass these arrays of words through our N-gram based language model and come up with a complete sentence. However, if we limit ourselves to the recognition of isolated words or limited phrases, the language model can be an optional component because most of it is used to provide context to the speech recognition process.

Software Development Life Cycle

Talking about software development methodology, we will use srcum – since the problem we've chosen is dynamic in nature and we are not an expert on the subject matter we are tinkering. Scrum has many advantages over traditional, planned way of software development, one of which is that everyone can work as cross-functional team and it focuses on delivering the software by cutting other corners that slows the software development.

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. A Rough estimation of the project can be summarized as follows

1. Frame and Window Signals
2. Perform De-Noising
3. Extract Feature Vectors
4. Learn about Phoneme Recognition Process and Convert Feature Vector to Phoneme Classification
5. Learn more about Hidden Markov Model, Viterbi Search and its application in ASR
6. Differentiate between training and testing phase (Mostly done during/after Phoneme Recognition)
7. After Completing Isolated Words Detection, Research on Word Boundaries and extend the project to recognize connected words.

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, "Fundamentals 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

L. Muda *et al.,* "Voice Recognition using MFCC and DTW Techniques" in *Journal of Computing*, 2010, vol. 2 pp. 138-143

T. Yang, "The Algorithms of Speech Recognition, Programming and Simulating in MATLAB," University of Gaive. 2012.

K. Mark *et al.*, "Noise Removal in Speech Processing Using Spectral Subtraction" in *Journal of Signal and Information Processing*, 2012, vol. 5 pp 32-41.

Appendix

Hidden Markov Model

Hidden Markov models can be considered as an extension of Markov Models. In HMMs, underlying Markov processes with states $z_1, z_2, \dots, z_n$ are present but cannot be observed directly. The emissions however, denoted by $x_1, x_2, \dots, x_n$ which are the probabilistic function of the hidden states can be observed. These are helpful when we have to predict states based on what we can observe and when we do not care about how these states were generated.

HMMs have other advantages too. Since HMM contains a Markov Model as its hidden layer, the underlying layer is also simplified since Markov Model assumes that all the past data can be eradicated in order to predict the future if the current data is present.

Viterbi Algorithm

Viterbi Algorithm solves the problem of finding optimal state sequence

$$q^* = \{q_1{}^*, q_2{}^*, \dots, q_T{}^*\}$$

such that the probability $p(q^*)$ is maximum given an observable sequence $\{o_1, o_2, \dots, o_T\}$ given an HMM.

The basic use of Viterbi algorithm is to predict words after giving a phoneme sequence.

Miscellaneous Formulae

Conversion to Mel Scale: $F(mel) = 2595 \log_{10}\left(1 + \frac{f}{700}\right), f = freq. in\ Hz$

Hamming Window: $w(n) = \alpha - \beta \cos\left(\frac{2\pi(n-1)}{N-1}\right), n = 0, 1, 2, \dots, N-1$