The three sections (Section A, B, C) in this document explain the pre-requisites of an automatic speech recognition system (ASR), data collection process, and the training algorithms that were used to train the ASR.

**Section A: Automatic Speech Recognizer (ASR) Pre-requisites:**

The basic design principle of a speech recognition system is that it starts learning individual sounds first. Then by concatenating a valid sequence of individual sounds, it learns words. Then it learns sentences from a valid sequence of words. This is similar to how a child learns a language.

The main components required to build an ASR system are:

1. Audio files

2. Transcription files with time information (what was spoken when)

3. Dictionary where each entry in the dictionary is a word followed by a set of phones telling the speech recognizer a valid sequence of individual sounds.

(E.g. UNIVERSITY Y UW N AH V ER S AH T IY)

4. A language model built over a large collection of sentences collected over the internet.

A-1, A-2, A-3 are required for the ASR to learn individual sounds and words. This is good enough if we want to deploy the ASR to output isolated words. However, if we want the ASR for to work for spontaneous speech (more realistic scenario), then A-4 is required to learn the sentence structure.

We obtained A-1 through A-4 from the following sources:

A-1, A-2: Audio files and text from IL news broadcasts

A-3: An existing CMU dictionary (133,000 words)

A-4: The text from IL news broadcasts were used to build the language model

**Section B: Data collection and processing:**

This most expensive and time-consuming part of the data collection process is manual annotation. This is because while training an ASR system, we need to let the system know what was spoken when (as mentioned in Section A-2) as accurately as possible. This can be achieved by marking the time boundaries at the phone/word/sentence/paragraph level. Keeping in mind the time and costs involved, we decided to mark the time boundaries at the paragraph level.

1. A collection of audio files and their corresponding text were collected from IL News agency. The text collected from IL News agency had no time boundaries initially.

2. Undergraduates were hired to manually mark the time boundaries - i.e. start and end times in terms of seconds and milliseconds. Although marking at the word boundaries can lead to better speech recognizer accuracies, this is an expensive process. To expedite the process, boundaries were marked at the start and end points of small paragraphs instead of a word. A "paragraph" here means at least one (but possibly more) English sentences. On an average, the paragraphs are about 12.42 seconds long. This constitutes the first pass of the manual transcription.

3. Natalie did a second pass of the transcriptions by fixing errors from the first pass.

4. Amit did a third pass of the transcriptions by doing additional post-processing to make it ready for ASR experiments.

Below is a typical conversion of a transcript from raw to processed form after third pass. The transcript below is artificial but it highlights many instances of conversion.

Raw:

21m51.320s 22m03.853s "SIL this is a test of “hello” Mary’s a/b a&b a%b w-w-w SEC for-you a.m. Feb. concluded B. Obama. SIL"

Processed:

21m51.320s 22m03.853s "!SIL this is a test of hello Mary's a slash b a and b a percent b w. w. w. S. E. C. for you a. m. Feb. concluded B. Obama. !SIL"

**Section C: Speech Recognition Algorithm Steps:**

HMMs (Hidden Markov models) were used to build the ASR system. The following steps outline the training algorithm:

1. Feature extraction: Each audio clip is split into smaller frames of 25 ms long. At 16 kHz sampling rate, there are 400 speech samples in every frame. Those speech samples were converted to a set of 40 Mel frequency cepstral coefficients (MFCCs). MFCCs are a compact representation of the most relevant spectral features associated with a frame.

2. Monophones training: Monophones are the individual sounds of speech. For example, in the word "university", there are 10 monophones viz. /Y/ /UW/ /N/ /AH/ /V/ /ER/ /S/ /AH/ /T/ /IY/. Monophone HMMs (hidden Markov models for individual sounds) were trained using the Expectation-Maximization (EM) algorithm. Forced alignments at the monophone level were generated using Viterbi decoding. These alignments provide the timing information for each phone for subsequent training stages.

3. Triphone training: Triphones are sounds associated with a temporal context. For e.g., in the word "university", there are 10 triphone contexts viz. # - /Y/+/UW/, /Y/-/UW/+/N/, /UW/-/N/+/AH/, /N/-/AH/+/V/, /AH/-/V/+/ER/, /V/-/ER/+/S/, /ER/-/S/+/AH/, /S/-/AH/+/T/, /AH/-/T/+/IY/, /T/-/IY/+#. It is clear that each center phone is associated with a left (indicated by - ) and right phone (indicated by +) which defines the temporal context of the center phone. Using the time information from the monophone alignments, triphone training was performed again using the EM algorithm. Following this, forced alignments were generated for triphones using Viterbi decoding.

4. Feature Discrimination using LDA: Linear discriminant analysis (LDA) was performed on the MFCCs to obtain better features which improve discrimination capabilities (between class separation/within class separation). Furthermore, a maximum likelihood linear transform (MLLT) was also computed to transform the means of the existing model so that the transformed model increases the likelihood of the data. The final model is the LDA+MLLT model.

5. Speaker Adaptive Training: The forced alignments obtained from the LDA+MLLT model were further used for speaker adaptive training (SAT). This was done by computing feature-space maximum likelihood linear regression (fMLLR) transforms per subset of speakers. This is the LDA+MLLT+SAT model. This is the final trained HMM. This concludes the training stage.

In the test stage, the LDA+MLLT+SAT triphone HMM model was used to convert speech to text for the test audio clips. Using, this we obtained a word error rate (WER) of 43%.

Word Error Rate (WER) = 100 x (TE)/(TW)

TE = Total number of errors caused by the speech recognizer

TW = Total number of words in the test transcripts

TE is calculated as the minimum number of word insertions + deletions + substitutions required to exactly match the output transcripts of the speech recognizer with the true (or reference) transcripts.