Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 6586–6592
Marseille, 11–16 May 2020

Preparation of Bangla Speech Corpus from Publicly Available Audio & Text

Shafayat Ahmed[*†], **Nafis Sadeq**[*†], **Sudipta Saha Shubha**[*†], **Md. Nahidul Islam**[*]
Muhammad Abdullah Adnan[*], **Mohammad Zuberul Islam**[‡]
[*]Bangladesh University of Engineering and Technology (BUET)
shafayat, nafis, sudipta, nahid.rimon@ra.cse.buet.ac.bd, adnan@cse.buet.ac.bd
[‡]Samsung R & D Institute Bangladesh
m.zuberul@samsung.com

Abstract

Automatic speech recognition systems require large annotated speech corpus. Manual annotation of a large corpus is very difficult. In this paper, we focus on automatic preparation of a speech corpus for Bangladeshi Bangla. We have used publicly available Bangla audiobooks and TV news recordings as audio sources. We designed and implemented an iterative algorithm that takes as input a speech corpus and a huge amount of raw audio (without transcription) and outputs a much larger speech corpus with reasonable confidence. We have leveraged speaker diarization, gender detection, etc. to prepare the annotated corpus. We also have prepared a synthetic speech corpus for handling out-of-vocabulary word problem in Bangla language. Our corpus is suitable for training with Kaldi. Experimental results show that use of our corpus in addition to Google Speech corpus (229 hours) significantly improves the performance of the ASR system.

Keywords: Speech recognition, Speech corpus generation

1. Introduction

Automatic Speech Recognition (ASR) systems require an annotated speech corpus for training. Speech corpus refers to a collection of audio files with corresponding text transcriptions. The duration of each speech segment is kept under 35 seconds (Panayotov et al., 2015). The speech corpus also contains some additional information such as the gender of speakers, recording environment, etc. For achieving WER of less than 15% for speech recognition, we need a huge annotated speech corpus, more than 1000 hours according to (Moore, 2003). Very few publicly available speech datasets are available for Bangladeshi Bangla. The largest publicly available corpus is the 229 hours speech corpus released by Google (Kjartansson et al., 2018).

Different approaches for speech corpus development have been explored by the researchers. A basic approach is to manually transcribe existing audio files. This is a very time consuming, monotonous, and error-prone task. Transcription of one-hour recording can take 3 to 5 hours or more (Hazen, 2006). A comparatively faster approach is to develop an interactive mobile application that prompts the user to read a particular text (Hughes et al., 2010). The user will have to start recording manually, read the prompted text, and end recording manually. In our experience, preparing one hour's worth of transcribed speech takes around 2 hours in this approach. But this approach still requires some manual labor. A large number of speakers need to be motivated to use this application. It is not cost-effective to offer an attractive incentive for every speaker participating in this process. Some works such as (Panayotov et al., 2015) used forced alignment techniques to produce speech corpus from public domain audiobooks and corresponding text. This approach does not apply to Bangla because for most Bangla audiobooks, corresponding digital text cannot be found. Hence, these existing approaches are not practical and scalable to develop a large speech corpus in the context of Bangla language.

To provide a scalable solution towards developing a large speech corpus, we provide a novel approach considering the limited resources of Bangla language. We provide an iterative algorithm to automatically generate transcription from existing audio sources. Many famous Bangla books are available as audiobooks. Recently, a lot of Bangladeshi TV channels are providing their TV news videos in the public domain such as YouTube. In this work, we have used these audio resources and prepared 510 hours of annotated speech corpus from around 1500 hours of raw audio files. Our algorithm considers output from two different not-so-well-performing speech recognition systems for Bangla language: one is Google Speech API (Google, 2019) and another is developed by us. By considering the difference in output from these two systems, we systematically improve the performance of the speech recognition system developed by us. In each iteration, we calculate the similarity in output from both the systems and add only those audio and text segments in the train set that both the systems agree in transcription. Thus we gradually develop our speech corpus with high confidence audio and text data. Moreover, we have prepared a Bangla word list from publicly available text sources. We have prepared 450 hours of synthetic audio for around out-of-vocabulary Bangla words. Overall, the total size of the speech corpus is around 960 hours.

Our major contributions are as follows:

- We have developed a 960 hours annotated speech corpus automatically from publicly available audio and text data.
- We present an approach that requires minimal manual verification after the transcription process is completed.
- We have prepared synthetic utterances for around 1.6

[†] authors contributed equally

million unique Bangla words, which reduces ASR error induced by the out-of-vocabulary problem.

- Experimental results show that the use of our speech corpus in addition to the Google speech corpus significantly improves the accuracy of the ASR system.

The organization of this paper is as follows. Related works are discussed in section 2, system and system components in section 3, corpus description in section 4, experimental results in section 5, and conclusion and future works in section 6.

2. Related Work

In various languages, researchers have explored different methods for developing speech corpus. Jang and Hauptmann (Jang and Hauptmann, 1999) develop a speech corpus from captioned multimedia speech. Lakomkin et al. (Lakomkin et al., 2019) develop a tool to automatically construct data set for speech recognition from YouTube videos containing transcriptions. Panayotov et al. (Panayotov et al., 2015) present 1000 hours of speech corpus for English by aligning texts and audio files of audiobooks. Mansikkaniemi et al. (Mansikkaniemi et al., 2017) and Helgadóttir et al. (Helgadóttir et al., 2017) use similar alignment techniques to develop speech corpus from recordings and transcriptions from parliamentary speech. Patel et al. (Patel et al., 2018) build a data collection tool and collect around 100 hours of reading speech data in Manipuri Language.

Compared to other languages, research works on developing speech corpus for Bangla language are quite limited. Nahid et al. (Nahid et al., 2018) discuss the development of Bangla real number audio corpus. The recordings were completed in a supervised environment and volunteers were given scripts to read. Khan and Sobhan (Khan and Sobhan, 2018a) develop a speech corpus containing only isolated words for Bangla. All recordings were done in a laboratory. In another work of them, Khan and Sobhan (Khan and Sobhan, 2018b) develop a speech corpus of connected words for Bangla. Researchers from Google (Kjartansson et al., 2018) prepare speech corpora for Bangla and four other languages using interactive mobile application (Hughes et al., 2010). They develop 229 hours of speech corpus for Bangla. Our work differs from these works as none of these works deals with automatic preparation of speech corpus in Bangla language using existing audio and text data.

3. Our System

3.1. Overview of Corpus Preparation

Figure 1 shows an overview of our system. We use publicly available audiobooks and TV news recordings collected from YouTube as an audio source in our system. All our audio files are converted to a 16 kHz mono channel WAV file. All the audio files are equal to or less than 30 minutes of length. We then remove the background noise of the audio files. After that, we perform speaker diarization on the audio files to group the audio portions of the same speaker together. We perform automatic gender detection on the audio files to identify the gender of the speaker. We

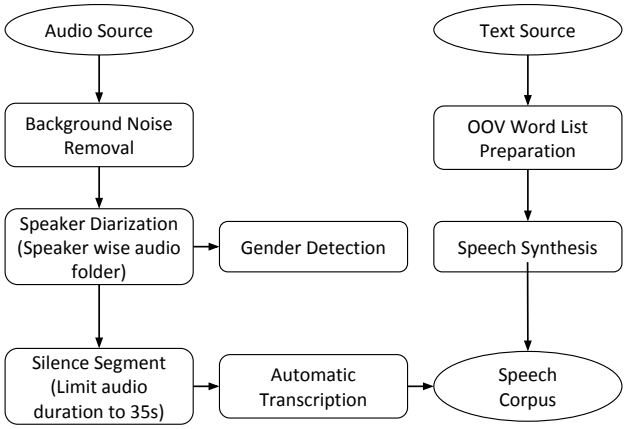

Figure 1: Overview of Corpus Preparation

segment each of the audio files on silence intervals and ensure that all the audio segments are less than or equal to 35 seconds. Finally, we automatically generate transcriptions for the audio files as we do not have corresponding text for the audio files. For generating the transcriptions with reasonable confidence, we have designed and implemented an iterative algorithm (Algorithm 1). Details of each of the system components are described in section 3.2.

3.2. Corpus Preparation from Raw Audio

3.2.1. Background Noise Removal

It is important that we remove the background noise from the audio files for proper speech transcription. We follow (Dev and Bansal, 2010) and (Ravindran et al., 2006). We study the MFCC features of the audio files for noise identification. In the beginning, the auto-correlation coefficients of relatively higher order are extracted. Then we use FFT on the magnitude spectrum of the resultant speech signal and it is differentiated with respect to frequency. Finally, the differentiated magnitude spectrum is transformed into MFCC-like coefficients.

3.2.2. Speaker Diarization

Speaker diarization refers to the task of grouping speech segments in an audio stream containing multiple speakers in a way that speech segments from the same speaker form a cluster. We follow (Patino et al., 2018a) for speaker diarization. This speaker diarization system (Patino et al., 2016),(Patino et al., 2018b), (SAIVT-BNEWS, 2019) is based on the binary key speaker modelling (Ng et al., 2002). Binary key speaker modelling provides a compact and efficient representation of speech segments or clusters in the form of a vector. The vector captures speaker-specific features. The classification task is carried out by computing the similarity measures between binary keys. The proposed system obtained a Diarization Error Rate (DER) of 11.93%. In this system, ICMC (Q transform Mel-frequency cepstral coefficients) were used in place of baseline MFCC acoustic features. For clustering, an affinity matrix is calculated. The eigenvectors corresponding to the top eigenvalues estimated from that derives the similarities between data points being clustered. So after refinement, which smooths and

denoises the data, we perform Eigenvalue decomposition and sort the eigenvalues in descending order. Then we select the number of clusters according to the value which maximizes the eigengap. The spectral clustering algorithm often results in the estimation of a single speaker, therefore the system is configured to force the return of two or more clusters. Then the system performs pre-clustered thresholding of the eigengap between the two largest eigenvalues which completes the processing.

3.2.3. Gender Detection

We use (Blog, 2017) as a guideline for gender detection. We extract Mel Frequency Cepstrum Coefficients (MFCCs) features from the audio files. A lot of acoustic features like peak frequency (the frequency with the highest energy), meanfun (average of fundamental frequency measured across acoustic signal), minfun (minimum fundamental frequency measured across acoustic signal), etc. are included in MFCC features. We use a Gaussian Mixture Model to build the gender detection system from these extracted features. The training dataset consists of Mozilla common voice data set (Voice, 2019). We had almost 58,000 male voice clips and 17,000 female voice clips in the train set. After training, a test data set consisting of manually tagged Bangla audio clips are used for evaluation. The test set had 826 male and 590 female audio clips. Even though the training set and test set had completely different languages, we achieved a recognition rate of 85% and 98% for male and female clips respectively.

3.2.4. Silence Segment

We segment each of the audio files on silence intervals and ensure that all the audio segments are less than or equal to 35 seconds. We use PyAudioAnalysis (Giannakopoulos, 2015) for this task. We take 0.4 seconds as minimum silence length (the minimum length of the silence at which a split may occur) and 0.0001 as silence threshold (the energy level (between 0.0 and 1.0) below which the signal is regarded as silent).

3.2.5. Automatic Transcription Generation

As we did not have any reference text of any of the audio files, we had to automatically generate the transcriptions. We designed and implemented an iterative algorithm for this task (Algorithm 1).

This algorithm uses two speech recognition systems: one is Google Speech API (Google, 2019) and another is our speech recognition system that has been trained on publicly available 226 hours of speech data from Google (out of the remaining 3 hours, 2 hours for the test set, 1 hour for validation set). We use a hybrid CTC-Attention based end to end system for training (Watanabe et al., 2017). The details of our hyper-parameter choices are given in section 5.2 and the flowchart of automatic transcription process is given on 2

We observe that none of the two systems provide good enough performance and Google API provides better performance than our system. To generate transcription with reasonable confidence, we decide to generate transcriptions using the outputs from both the models. Our intuition is that, for each of the audio files, if we take the longest common sequence of consecutive words between the outputs of both the systems and take only the audio and transcription for that matched portion, we can be confident enough about the accuracy of the transcription. However, we cannot do only 1 iteration as our speech recognition system was performing quite poorly initially. So, we follow an iterative strategy. At each iteration, we increase the number of training data to get a better model in the next iteration. Note that, we only try to increase the performance of our speech recognition system. At each iteration, we generate the transcriptions of the audio files from both of the systems. We consider the longest common sequence of consecutive words from both of the transcriptions. We take the percentage of the length of this matched portion with respect to the length of transcription from Google API. If this percentage value is greater than a threshold (50%), we take only the audio and transcription for that matched portion. After doing these for all the audio files, we add the segmented audio portions and their corresponding texts in the train set for the next iteration. From each initial audio file, we take only one such segmented audio portion and transcription. We stop iterating when the number of newly added training samples does not increase much compared to the previous iteration. At the start of each iteration, we delete the training samples added at the last iteration. Without the deletion of those samples, we observed repetitive data in the training set. Finally, for each of the audio files, we take the longest common sequence of consecutive words between the outputs of the final version of our speech recognition system and Google Speech API. We take only the audio and transcription for that matched portion to be included in our final speech corpus. The performance of the algorithm is discussed in section 5.1. We only consider exact matching within our threshold. The matching threshold of 50% is intuitive. It may be possible to improve the algorithm by tuning this threshold. But we avoid doing that due to the computational complexity of the iterative corpus generation algorithm.

We refer to the corpus generated from Automatic transcription as 'Transcribed corpus'. The size of the Transcribed corpus is around 510 hours.

3.3. Corpus Preparation for OOV words

In this section, we describe our approach for synthetic corpus generation for out-of-vocabulary Bangla words.

3.3.1. Out-of-Vocabulary Word List

We first prepare a large Bangla text corpus. It is described in section 4. Our text corpus has 10 million Bangla sentences containing 1.7 million unique words. Among these words, 56000 words occur at least once in the speech corpus. The rest of the words are considered out-of-vocabulary.

3.3.2. TTS Model

We prepare our text-to-speech system using ESPnet-TTS (Hayashi et al., 2019). Specifically, we use the Tacotron 2 (Shen et al., 2018) implementation of ESPnet-TTS. Tacotron 2 is a Recurrent Neural Network (RNN) based sequence-to-sequence network. It has a bi-directional LSTM based (BLSTM) encoder and a unidirectional

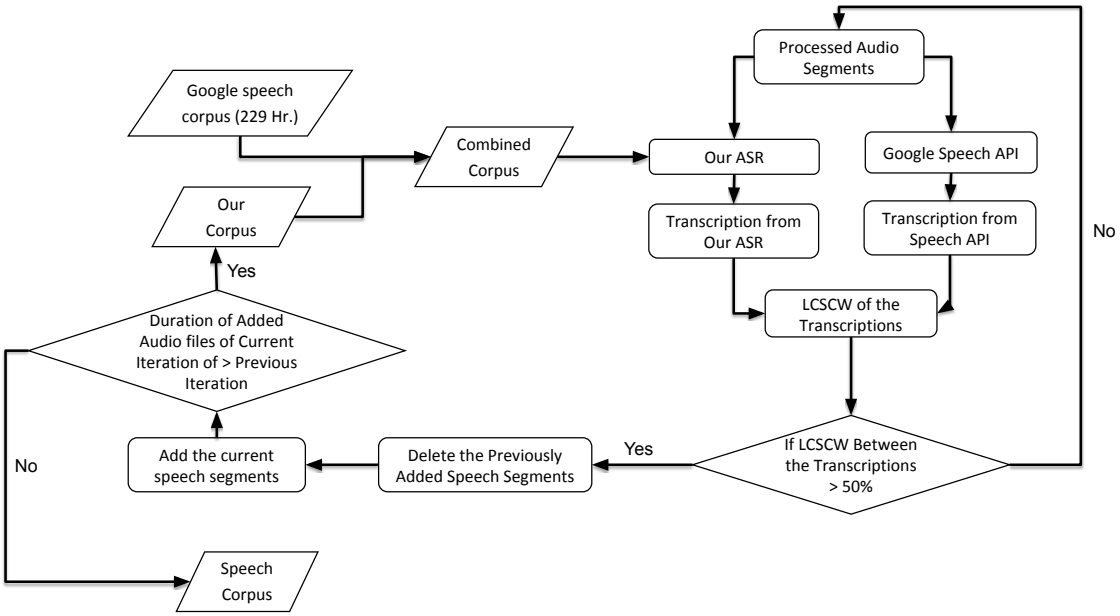

Figure 2: Overview of Automatic Transcription

Algorithm 1 Iterative Algorithm for Transcription

$ot \leftarrow$ Our ASR transcription
$gt \leftarrow$ Google API transcription
$lcscw \leftarrow$ Longest common sequence of consecutive words
$gc \leftarrow$ Google speech corpus
$oc \leftarrow$ Our speech corpus
$dc \leftarrow$ Duration of train data at current step
$dp \leftarrow$ Duration of train data at previous step
$d\delta \leftarrow$ Change in duration of speech corpus
$al \leftarrow$ List of audio files
while $d\delta \neq 0$ **do**
    Train ASR on $(gc + oc)$
    $oc \leftarrow \{\}$
    $dc \leftarrow 0$
    for each $audio$ in al **do**
        generate ot
        generate gt
        $lcscw \leftarrow lcscw(ot, gt)$
        $percentage \leftarrow \frac{len(lcscw)*100}{len(gt)}$
        if $lcscw\_percentage > 50\%$ **then**
            $dc \leftarrow dc + audio\,duration$
            $oc \leftarrow oc + matched\,audio\,segments$
        $d\delta \leftarrow dc - dp$
        $dp \leftarrow dc$
return oc

LSTM-based decoder. Additionally, it uses a location-sensitive attention mechanism. In our implementation, the encoder network has 1 layer with 512 BLSTM units. The decoder network has 2 layers with 1024 unidirectional LSTM units in each layer.

3.3.3. Speech Synthesis

Speech synthesis is done in the following manner. First, our TTS model takes an input text sequence and generates log Mel filter bank feature sequence. Then log Mel filter bank feature sequence is converted to a linear spectrogram. Finally, the Griffin-Lim algorithm (Perraudin et al., 2013) is applied to the spectrogram to generate audio.

We use out-of-vocabulary words as input for our text-to-speech system. We refer to the corpus generated from Speech synthesis as 'Synthesized Corpus'. The size of the Synthesized corpus is around 450 hours.

4. Corpus Description

4.1. Speech Corpus

Our final corpus has about 297065 transcribed audio files. The overall duration of the corpus is 960 hours, the average duration of audio files is about 7.14 seconds. There are 519 speakers in the corpus. Among them, 268 speakers as male and 251 speakers as female. We split the corpus into three portions: train, validation, and test. 1000 audio files were selected as the validation set and 13000 audio files were selected as a test set. The test set was manually corrected and verified. The duration of validation and test set were 1 hour and 13 hours respectively.

4.2. Text Corpus

We crawl around 42 Bangla websites and apply text cleaning to remove non-Bangla text, punctuation, alphanumeric characters, duplicate text from the collected raw text. Then we apply text normalization. We convert numbers to text, expand abbreviations, normalize percentage symbol and decimal point, consider the date, contact numbers, etc. After preprocessing, we have 10 million Bangla sentences containing 1.7 million unique words.

5. Evaluation

5.1. Evaluation of Iterative Algorithm

Figure 3 shows the histogram of the percentage of the longest common sequence of consecutive words (LCSCW). We calculate it in the following way. We calculate the LCSCW between the transcription provided by our ASR and Google Speech API. We calculate the percentage of LCSCW with respect to the transcription length provided by the Google Speech API. We plot the histogram of these percentages within 10 ranges: 0-10%, 10-20%, etc. Each color in the graph represents a particular iteration. We can see from figure 3 that in the earlier iterations, most of the LCSCW percentages are in shorter regions. Iteration 1 has the highest frequency in the lower percentage area. As we add more data to the training corpus, the performance of our ASR increases. It starts recognizing more words accurately, resulting in a longer common consecutive word sequence length. We can see that in iteration 6 and 7, there are more sentences with higher LCSCW percentages. Figure 3 shows the rightward shift of the histogram during different iterations of algorithm 1. While analyzing Figure 3, we need to keep in mind that Google Speech API does not provide perfect transcription. It may be the case that the LCSCW between two transcriptions for audio can decrease at a later iteration - this does not necessarily imply that our ASR is getting worse at that iteration.

Table 1 shows the evaluation of the corpus generated at each iteration of our algorithm. The second column shows how many transcribed speech data were generated at that particular iteration. We train a hybrid CTC-Attention based end to end system using each corpus and evaluate the performance of that system. We use our generated corpus at each step in addition to the Google speech dataset for evaluation. We can also see in table 1 that the amount of transcribed corpus generated at each iteration and corresponding WER reaches saturation after only 6-7 iterations. One possible reason is the limited ability of the system to add variance to the existing training corpus. The size of the corpus after the system reaches saturation is around 510 hours. The drawback of this system is that it fails to utilize all collected audio files. But there are two key benefits. It allowed us to transcribe 510 hours of speech data very quickly. Also, this system is very useful in cases where forced alignment technique cannot be used (i.e., no reference text available).

Table 1: Evaluation of Corpus by Iteration

Iteration	Size of Corpus (Hours)	WER (%)
1	207	25.98
2	379	24.25
3	426	23.64
4	464	23.46
5	492	23.22
6	509	23.08
7	512	23.00

Table 2: Evaluation of ASR performance

Train Set	Model	WER (%)
Google	HMM-GMM	30.95
	CTC-Attention	26.0
Google +Transcribed	HMM-GMM	27.20
	CTC-Attention	23.0
Google +Synthesized	HMM-GMM	31.40
	CTC-Attention	26.6
Google+ Transcribed+Synthesized	HMM-GMM	24.38
	CTC-Attention	20.2

5.2. Evaluation of ASR Performance

In this section, we evaluate the ASR systems trained on our speech corpus in addition to the Google speech corpus. We prepare a standard test set. We include sentences from the various domain in our test set and add out-of-vocabulary words such as named entities. The test set contains 13000 transcribed audio files. We manually verified all the transcriptions in the test set. Our test set has 35 speakers, 26 males, and 9 females.

To maintain fairness, all the experiments mentioned below use the same text corpus for language model training. The text corpus is described in section 4. We prefer using this additional text corpus instead of the audio transcriptions for the training language model. As our system uses automatically transcribed data from corresponding clipped audio segments, the transcriptions may contain a lot of incomplete sentences and may not be the best text source for language model training.

We use two different models for evaluating each speech corpus. As our first model, we use traditional HMM-GMM based recipe from Kaldi (Povey et al., 2011). HMM-GMM based system requires a lexicon that contains the phonetic transcriptions for all words in the vocabulary. The same lexicon was used for evaluating both speech corpora. We use our previously developed lexicon (Shubha et al., 2019). The lexicon contains 95000 transcribed Bangla words. We use most frequently used 65000 words in all of our current experiments. An N-gram based language model is used with HMM-GMM based model.

As our second model, we use a deep learning-based end to end speech recognition system that uses a hybrid of CTC and Attention mechanism. We follow the approach of (Watanabe et al., 2017). We use four BLSTM layers in the encoder network. The number of BLSTM cells in each layer is 320. Each BLSTM layer is connected to a linear projection layer with 320 units. The decoder network has 1 layer with 300 unidirectional LSTM units. We use a Recurrent Neural Network-based language model in shallow fusion with the CTC-Attention network. We use character level RNN. The RNN has 2 layers with 650 LSTM units in each layer. We get the best performance when using CTC weight 0.3 with the language model weight 0.5.

All training is done on a desktop with a Core i7 processor, 16 GB RAM, and Nvidia RTX 2070 GPU. When training with our 1189 hours speech corpus (229 hours from Google, 510 hours Transcribed, 450 hours Synthesized),

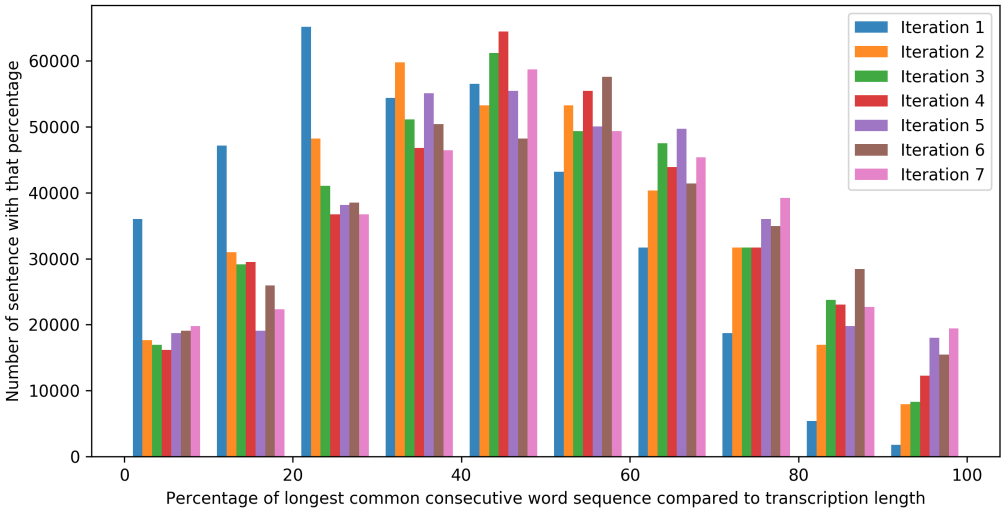

Figure 3: Histogram for percentage of longest common consecutive word sequence length between two transcriptions with respect to transcription from Google Speech API

the HMM-GMM based recipe took about 40 hours and a deep learning-based system took around 100 hours in the above configuration.

Table 2 shows the performance of the ASR system for different combination of training datasets and models. Best performance is achieved when we use both transcribed and synthesized corpus along with Google's dataset. For example, when using CTC-Attention network, the combined corpus achieves WER of 20.2%. This system outperforms the same model trained on Google's dataset only, which shows WER of 26.0%. When we use Google+Transcribed corpus the WER is 23.0%. This shows the effectiveness of our iterative corpus generation approach. When use Google+Synthesized corpus, the system actually performs even worse than the system trained on Google corpus alone. Possibly, it happens because the size of synthesized corpus is larger than Google corpus. The overwhelming size of the synthesized corpus leads to over-fitting. But presence of synthesized corpus improves the ASR performance when we use all three corpus combined(see table 2).

6. Conclusion and Future Works

In this paper, we present a novel approach for preparing an annotated speech corpus automatically from public domain audio and text resources. Following this approach, we were able to develop a speech corpus relatively quickly compared to other conventional approaches. We use speaker diarization, gender detection, and existing low-performance ASR to annotate the public domain audio automatically. We also use a text-to-speech system for synthesizing audio for all public domain Bangla words. This method of corpus development can be used in addition to more supervised methods of speech corpus development to build large speech corpus efficiently. In our current approach, we only consider the audio portions that lead to good quality transcriptions. We have set the threshold to 50% for the percentage of the longest common sequence of consecutive words (between the 2 speech recognition outputs) intuitively as we felt taking below than this threshold will negatively impact the accuracy. In future, we will extend this work and see how this threshold percentage change behaves with overall accuracy. Moreover, we want to incorporate the approach like (Karita et al., 2018) that would allow us to utilize the unpaired audio and text dataset more efficiently.

7. Acknowledgements

This research work is funded by Samsung R & D Institute Bangladesh.

8. Bibliographical References

Dev, A. and Bansal, P. (2010). Robust features for noisy speech recognition using mfcc computation from magnitude spectrum of higher order autocorrelation coefficients. *International Journal of Computer Applications*, 10(8):36–38.

Giannakopoulos, T. (2015). pyaudioanalysis: An open-source python library for audio signal analysis. *PloS one*, 10(12).

Hayashi, T., Yamamoto, R., Inoue, K., Yoshimura, T., Watanabe, S., Toda, T., Takeda, K., Zhang, Y., and Tan, X. (2019). Espnet-tts: Unified, reproducible, and integratable open source end-to-end text-to-speech toolkit. *arXiv preprint arXiv:1910.10909*.

Hazen, T. J. (2006). Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Ninth International Conference on Spoken Language Processing*.

Helgadóttir, I. R., Kjaran, R., Nikulásdóttir, A. B., and Guðhnason, J. (2017). Building an asr corpus using althingi's parliamentary speeches. In *INTERSPEECH*, pages 2163–2167.

Hughes, T., Nakajima, K., Ha, L., Vasu, A., Moreno, P. J., and LeBeau, M. (2010). Building transcribed speech corpora quickly and cheaply for many languages. In *Eleventh Annual Conference of the International Speech Communication Association*.

Jang, P. J. and Hauptmann, A. G. (1999). Improving acoustic models with captioned multimedia speech. In *Proceedings IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 767–771. IEEE.

Karita, S., Watanabe, S., Iwata, T., Ogawa, A., and Delcroix, M. (2018). Semi-supervised end-to-end speech recognition. In *Proc. Interspeech*, pages 2–6.

Khan, M. F. and Sobhan, M. A. (2018a). Construction of large scale isolated word speech corpus in bangla. *Global Journal of Computer Science and Technology*.

Khan, M. F. and Sobhan, M. A. (2018b). Creation of connected word speech corpus for bangla speech recognition systems. *Asian Journal of Research in Computer Science*, pages 1–6.

Kjartansson, O., Sarin, S., Pipatsrisawat, K., Jansche, M., and Ha, L. (2018). Crowd-sourced speech corpora for javanese, sundanese, sinhala, nepali, and bangladeshi bengali.

Lakomkin, E., Magg, S., Weber, C., and Wermter, S. (2019). Kt-speech-crawler: Automatic dataset construction for speech recognition from youtube videos. *arXiv preprint arXiv:1903.00216*.

Mansikkaniemi, A., Smit, P., Kurimo, M., et al. (2017). Automatic construction of the finnish parliament speech corpus. In *INTERSPEECH*, pages 3762–3766.

Moore, R. K. (2003). A comparison of the data requirements of automatic speech recognition systems and human listeners. In *Eighth European Conference on Speech Communication and Technology*.

Nahid, M. M. H., Islam, M., Purkaystha, B., Islam, M. S., et al. (2018). Comprehending real numbers: Development of bengali real number speech corpus. *arXiv preprint arXiv:1803.10136*.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856.

Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. (2015). Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210. IEEE.

Patel, T., Krishna, D., Fathima, N., Shah, N., Mahima, C., Kumar, D., and Iyengar, A. (2018). An automatic speech transcription system for manipuri language. *Show and Tell Session in INTERSPEECH, Hyderabad*.

Patino, J., Delgado, H., Evans, N., and Anguera, X. (2016). Eurecom submission to the albayzin 2016 speaker diarization evaluation. *Proc. IberSPEECH*.

Patino, J., Delgado, H., and Evans, N. (2018a). The eurecom submission to the first dihard challenge. In *Proc. INTERSPEECH*, volume 2018, pages 2813–2817.

Patino, J., Delgado, H., Yin, R., Bredin, H., Barras, C., and Evans, N. W. (2018b). Odessa at albayzin speaker diarization challenge 2018. In *IberSPEECH*, pages 211–215.

Perraudin, N., Balazs, P., and Søndergaard, P. L. (2013). A fast griffin-lim algorithm. In *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 1–4. IEEE.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al. (2011). The kaldi speech recognition toolkit. Technical report, IEEE Signal Processing Society.

Ravindran, S., Anderson, D. V., and Slaney, M. (2006). Improving the noise-robustness of mel-frequency cepstral coefficients for speech processing. *Reconstruction*, 12:14.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R., et al. (2018). Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4779–4783. IEEE.

Shubha, S. S., Sadeq, N., Ahmed, S., Islam, M. N., Adnan, M. A., Khan, M. Y. A., and Islam, M. Z. (2019). Customizing grapheme-to-phoneme system for non-trivial transcription problems in bangla language. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3191–3200.

Watanabe, S., Hori, T., Kim, S., Hershey, J. R., and Hayashi, T. (2017). Hybrid ctc/attention architecture for end-to-end speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 11(8):1240–1253.

9. Language Resource References

Applied Machine Learning Blog. (2017). *Voice Gender Detection using GMMs : A Python Primer*.

Google. (2019). *Cloud Speech API*.

SAIVT-BNEWS. (2019). *SAIVT News Dataset*.

Mozilla Common Voice. (2019). *Common Voice by Mozilla*.