AUTOMATED AUDIO VISUAL SPEECH CORPUS CREATION FROM PUBLICLY AVAILABLE SOURCES FOR BANGLA LANGUAGE

Md. Mohaiminul Islam *Farhan Tanvir* *Shafayat Ahmed* *Nafis Sadeq*
Muhammad Abdullah Adnan
{mohaiminmahim, farhan.tanvir.utshaw, shafayat309, nafissadeq}@gmail.com,
adnan@cse.buet.ac.bd
Bangladesh University of Engineering and Technology (BUET)

ABSTRACT

The performance of an automatic speech recognition (ASR) system can potentially benefit from other modes of data, such as visual information related to lip movement. For developing an accurate, robust, and efficient audio-visual speech recognition (AVSR) system we need a huge amount of labeled data. But in most cases, developing these resources takes a huge tedious effort for manual annotation. In this paper, we present our efforts to automatically develop an audio-visual speech corpus for Bangla language. We have developed a novel approach for efficient, automated audio-visual corpus development and for organizing the corpus for the speech recognition process. Using our automated approach we developed 25 hours of AVSR corpus for Bangladeshi Bangla. We make the corpus and related source code publicly available. [1] [2]

Index Terms— Audio-Visual Speech Corpus, Multimodal Speech Recognition, Automated Corpus Development

1. INTRODUCTION

Speech recognition has become tremendously popular in recent years to improve Human-Computer interaction. Being the most popular language for the internet, speech recognition for English is far superior to any other language. For Bangla speech recognition, there are limited resources out there. More resources may help significantly in improving the performance of Bangla ASR applications. Automatic speech recognition (ASR) systems have seen an upward trajectory in both their performance and accuracy. This is mostly due to improvement in the field of deep learning, more available data, and more capable hardware. Still, the need to improve computer perception of human speech remains a top research priority and also a desirable commodity in everyday life. This goal is unreachable for computer systems if it relies solely on the uni-modal data source for learning such as only audio-based speech recognition systems. To perceive human speech seamlessly computer systems need to learn to extract speech-related context from multiple modalities. Humans themselves can extract speech context from several physiological cues such as facial expressions, visual posture, voice modulation, and body language, etc. Hence audio-visual speech recognition has received more attention in recent years due to its promising prospects.

For training and evaluating such deep learning systems, a prerequisite is the construction of large audio-visual corpora. Often for research purposes, small corpora are built by manually collecting and labeling data on a small scale. This process is tedious and highly inefficient. Moreover, the collected data in controlled environments differ from spontaneous real-life behavior. However, there are a huge amount of audio and video resources publicly available on the internet. If these resources are utilized and processed correctly, we can construct a large audio-visual dataset. Such vast resources would also reflect spontaneous human speech behavior more effectively than data collected in the studio environment. This is an open-ended problem in audio-visual speech processing. Such a large-scale database can be crucial for determining the most effective AV feature vectors and training accurate classifiers. This paper presents a novel approach to automatically process publicly available resources to extract, organize, and label audio-visual data. We also include an audio-visual corpus for Bangladeshi Bangla which has been created by the proposed processing pipeline. Our main contributions are as follows

- An automated approach to generating audio-visual speech corpus in any language from publicly available resources.
- A new approach for automated corpus generation using unsupervised clustering algorithm for speaker diarization and detection of shot boundaries.
- We generate 25 hour AVSR corpus for Bangla language. Our corpus contains speech samples of over 750 speakers.
- We make the AVSR corpus publicly available along with the codes for automatic AVSR corpus generation.

[1] Source code can be found here - https://github.com/Utshaw/AVSD2

[2] Generated corpus here - https://drive.google.com/file/d/1zVBIcX7N9-iC2HvqmDe-oO46SGh4rlIK/view

We hope that this would allow and encourage other researchers to generate similar corpora for other low resource languages.

The organization of this paper is as follows. Relevant works are discussed in section 2, data selection preliminaries in section 3, the detailed layout of our processing pipeline in section 4, overview of our corpus in section 5, implementation and experimental setup in section 6, and conclusion and future works in section 7.

2. RELATED WORK

Automated AV corpus generation is a relatively new area of exploration. GRID[1] was one of the first AV corpora to contain full sentences and was of decent size, but the data collection was in a controlled environment consisting of a relatively small number of speakers. MODALITY[2] provided high-quality video data but was neither automatically generated nor used publicly available video resources. [3] used a multi-staged pipeline to automatically generate the 'Lip Reading Sentences' (LRS) corpus from a variety of BBC programs. They used HOG-based face detection to detect faces in every frame and a KLT tracker to group faces of individual speakers. [4] used a similar process to construct the LRS2-BBC dataset, the key difference being they used a CNN face detection scheme based on SSD (Single Shot MultiBox Detector) instead of the more traditional HOG-based one. Both studies used color histogram analysis among continuous frames to detect shot boundaries. [5] followed up their previous work by applying the same multi-staged pipeline on 400 hours of TED and TEDx videos to obtain LSR3-TED corpus. [6] built a filter-based pipeline system that heavily uses FlumeJava [7] large parallel processing and follows [8] to run selectively collected YouTube videos through multiple (eg.length, language, clip quality) filters and uses color histogram classifiers for face and shot boundary detection. The uniqueness of our approach lies in the method that we use unsupervised clustering group frames for each speaker(and isolate the primary speaker). Then our process uses the frame annotation information to detect shot boundary and frame continuity. The same labeling information is then used for AV sync and audio extraction. For a low resource language like Bangla, [9] prepares a speech-to-text dataset from publicly available news recordings and audiobooks.

3. PRELIMINARIES

We use publicly available TV news recordings collected from YouTube raw video source in our system. The important thing to notice that these recordings include one primary speaker (I.e. news presenter) and several other secondary speakers (e.g. reporters, featured people in reports, bystanders). That is why shot boundaries are longer and shot changes less occurring. Coincidentally, among publicly available Bangla videos, this genre of videos is the most abundant and also covers a wide array of topics and is spoken by a large number of individual speakers. Such a selection of input videos assure that data from a single topic or speaker would not outweigh others.

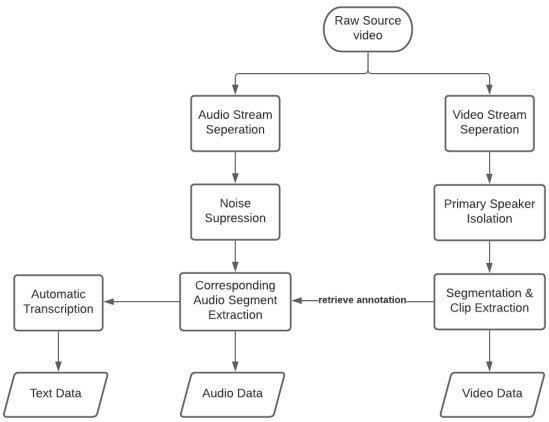

Fig. 1. System Overview

4. OUR SYSTEM

The Figure below describes the overview of our whole process. The goal of our process is to isolate the video segments where the primary speaker is active. It extracts and annotates these segments automatically, constantly taking raw source video as input and appending it to the serialized and meaningfully labeled dataset. We first separate the audio and video streams from our source. From this point onwards both streams are processed in parallel with the processed video stream sends labeling information, which the audio stream receives and utilizes to extract synchronized corresponding audio segments. The video processing thread performs two main actions. Firstly, it isolates the primary speaker segments from the raw footage, which is a nonobvious process on its own consisting of two steps- Facial image separation by a pre-trained face detection model and an unsupervised image clustering algorithm. Secondly, the process meaningfully labels these segments and feeds the labeling information to the audio processing thread. The audio processing thread firstly uses a denoising autoencoder to suppress background noise in the audio stream. After receiving labeling information this thread extracts corresponding audio segments from the audio stream. Both threads store these synchronized segments as structured audio and video data. The detailed process of each thread is explained in the following sections.

4.1. Data Collection

We collected publicly available Bangla news videos from numerous YouTube channels. The goal behind this was

twofold. For once, Tv news recordings for the majority of the duration focuses on a singular active speaker on the screen, which is ideal for our processing method. On the other hand, news recordings contain acoustic data of many different dimensions(e.g. Politics, sports, recent events). These would guarantee the prevention of domain bias from one or few-dimensional data. The process was also automated, using a crawler which utilized the youtube-dl program for collecting entire playlists from YouTube.A total of 1470 sample videos were collected, with a combined duration of approximately 410 hours. hence, Avg. duration of the raw footage was 16.7 minutes.

4.2. Video Data Processing

Each frame from the source input video was extracted. After that, a pre-trained face detection model was used to detect faces in each of the frames. The face portion on each of the frame was cropped. If multiple faces were detected in some frames, the largest face was chosen. We assume that the closest person to the camera is most likely the speaker in this case. Then those face images were stored and labeled with the frame number. In the next step, we used an unsupervised clustering algorithm for the detection of the primary speaker's face. Here we extract the largest cluster among all the stored faces in the previous step. Obviously, this specific face cluster represents the primary speaker in our raw video. In the next step, our goal is to obtain continuous video segments of the primary speaker speaking (without the audio). So, we need continuous frames. For that reason, we ran a linear search in our primary speaker cluster. The image files in the cluster were already labeled with their corresponding frame number. We used this information to separate continuous at least 100 to at most 350 frames. These segments yielded video clips of duration minimum of 4 secs to a maximum of 14 secs respectively. These clips were then labeled with speaker id, utterance id, and corresponding frame number. This labeling info is then sent to the audio processing thread to extract respective audio segments.

4.3. Audio Data Processing

Fig. 3 represents an overview of our audio processing thread. This thread runs in parallel with the video processing thread and is assigned with two main tasks. First of all, it reduces the background noise of the input video stream. This is done by standard techniques that analyze the MFCC feature space of the audio file and tries to identify the noise portion of the signal. In the second step, after retrieving labeling information about frame continuity of the primary speaker from the video stream the process extracts corresponding audio segments from the raw audio. Audio segments obtained in this fashion are converted to 16 kHz mono channel WAV files with a bit rate of 128 kbps. Converted audio files are organized and stored according to the corpus structure.

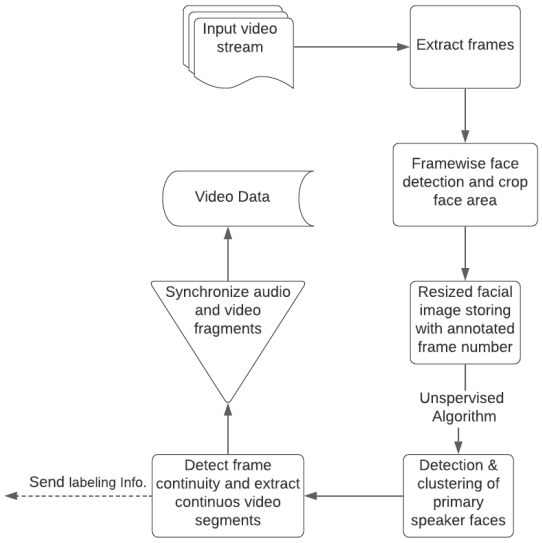

Fig. 2. Video processing thread

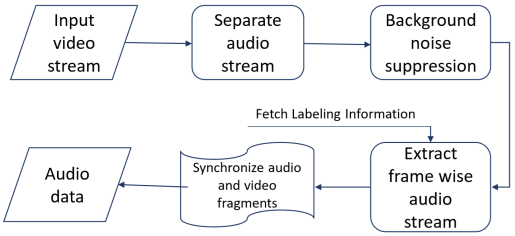

Fig. 3. Audio processing thread

4.4. Automatic Transcription

Subtitles for Bangla news recordings were not publicly available, so we followed the automatic transcription process proposed by [9] for transcription of speech data to text. We paired speech, text by labeling them with the same name.

5. DATASET

Finally, the audio-visual corpus contains roughly 25 hours of audio, video, and transcribed text with avg. duration of 8.52s per file. Hence, the ratio of raw footage to processed video is roughly 16:1. This ratio can be improved many times by processing different types of youtube content such as product reviews, vlogs as raw input. But the challenge, in that case, is, these videos are not as abundant for all languages as news recordings are. Moreover, excess of one specific video type might introduce domain bias into the AV corpus. Collecting raw video would then require a more sophisticated set of evaluation criteria. Table-1 and Table-2 summarizes the corpus properties.

Metric	Property or quantity
# of files	10,563
Total duration	25 hours
Avg. duration	8.52 sec (min = 4 sec) (max = 14 sec)
# of speakers	764
Audio format	.wav, 16.1 kHz mono channel

Table 1. Audio data summary

Metric	Property or quantity
# of files	10,563
Total duration	25 hours
# of speakers	764
Input video format	1280*720 pixels, FPS = 25
Extracted face size	55*55 pixels ($\pm 5\%$)
Extracted video format(resized)	Avg.[55*55($\pm 5\%$)] FPS = 25

Table 2. Video data summary

6. IMPLEMENTATION AND EXPERIMENTS

6.1. System Setup

To evaluate the efficiency of our corpus generation process, we perform the experiment on three machines with different configurations. 410 hours of raw video data were processed in each machine. The processing time along with corresponding hardware configuration is shown in Table 3. All three machines were running on Ubuntu 18.04 LTS operating system. Computationally, the most expensive operation of the entire corpus generation process is related to the face encoding and clustering operation. Since we perform the operation in the CPU, the speed largely depends on the CPU performance. RAM size becomes important if we process video with a large resolution. The speed of face encoding can be further improved by using GPU based implementation, but we do not use that for our experiment.

6.2. Face Detection Scheme

For detecting faces in each video frame we use a Caffe-based face detector model that utilizes the OpenCV DNN module. This DNN model was created with Single Shot Detection(SSD)[10] framework using a similar architecture to ResNet-10 as the backbone. It takes an image as input and creates a blob from it. The blob is passed through the DNN to detect all the faces existing in the image. Each of the detection has a confidence measure, which represents the probability of correct face detection. We set the confidence measure threshold to 60% to filter out weak detection.

Processor	RAM	GPU	Time (hours)
Pentium G4560	8 GB	GTX 970	895
Core i5 9400f	12 GB	RTX 2070	248
Core i7 8700k	16 GB	RTX 2070	130

Table 3. Hardware Configuration

6.3. Face Clustering Algorithm

Whereas other studies used feature-based trackers to identify and group faces of an individual we use a different approach. Our process runs an unsupervised clustering algorithm on the face collected by the previous step. The algorithm is fairly straight forward. Firstly it calculates the face encodings of all the stored faces and generates a serialized encoding for the face set. The encoding of each face are represented by a feature vector of length 128. After all the feature vectors are calculated we need a clustering algorithm to group them. Algorithms such as K-means clustering has to specify the number of clusters beforehand. As there is no way of knowing how many speakers would make an appearance in a video we decided to use Density-based spatial clustering of applications with noise (DBSCAN)[11]. DBSCAN algorithm groups data points based on their distance in an N-dimensional space. We calculate the Euclidean distance between the feature vectors in a 128-d space and use this measurement to identify dense regions (i.e clusters) in the data space. The maximum neighbourhood distance within a cluster is set to 0.5 and minimum number of samples in a cluster is 5. Ideally, each of these clusters represents the faces of an individual present on the source video. From this, we can determine the number of speakers in a given video as well as identify them individually.
After we have obtained the face clusters we detect shot change and shot boundaries. Most of the other works on this topic does this by analyzing the change in color histogram between frames. We go for a simpler yet much more efficient trick. As we have annotated the face images with corresponding frame numbers in section 6.2, we can easily check for frame continuity by label name and detect shot boundary. Frame number information can also be used to extract corresponding audio from the source video.

7. CONCLUSION AND FUTURE WORKS

In this work, we introduce a robust, accurate, automated audio-visual corpus generation process that will help to solve the resource constraint problem for speech recognition. We develop an audio-visual speech corpus for Bangla and make it publicly available. In the future, we will try to increase the size of the AVSR corpus. We would also like to develop a speech recognition architecture that can exploit an independent speech corpus and another independent lip-reading corpus simultaneously. This would allow us to utilize existing speech corpora for Bangla language along with our AVSR corpus.

8. REFERENCES

[1] Martin Cooke, Jon Barker, Stuart Cunningham, and Xu Shao, "An audio-visual corpus for speech perception and automatic speech recognition," *The Journal of the Acoustical Society of America*, vol. 120, no. 5, pp. 2421–2424, 2006.

[2] Andrzej Czyzewski, Bozena Kostek, Piotr Bratoszewski, Jozef Kotus, and Marcin Szykulski, "An audio-visual corpus for multimodal automatic speech recognition," *Journal of Intelligent Information Systems*, vol. 49, no. 2, pp. 167–192, 2017.

[3] Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman, "Lip reading sentences in the wild," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 3444–3453.

[4] Triantafyllos Afouras, Joon Son Chung, Andrew Senior, Oriol Vinyals, and Andrew Zisserman, "Deep audio-visual speech recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[5] Triantafyllos Afouras, Joon Son Chung, and Andrew Zisserman, "Lrs3-ted: a large-scale dataset for visual speech recognition," *arXiv preprint arXiv:1809.00496*, 2018.

[6] Brendan Shillingford, Yannis Assael, Matthew W Hoffman, Thomas Paine, Cían Hughes, Utsav Prabhu, Hank Liao, Hasim Sak, Kanishka Rao, Lorrayne Bennett, et al., "Large-scale visual speech recognition," *arXiv preprint arXiv:1807.05162*, 2018.

[7] Craig Chambers, Ashish Raniwala, Frances Perry, Stephen Adams, Robert R Henry, Robert Bradshaw, and Nathan Weizenbaum, "Flumejava: easy, efficient data-parallel pipelines," *ACM Sigplan Notices*, vol. 45, no. 6, pp. 363–375, 2010.

[8] Hank Liao, Erik McDermott, and Andrew Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for youtube video transcription," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. IEEE, 2013, pp. 368–373.

[9] Shafayat Ahmed, Nafis Sadeq, Sudipta Saha Shubha, Md Nahidul Islam, Muhammad Abdullah Adnan, and Mohammad Zuberul Islam, "Preparation of bangla speech corpus from publicly available audio & text," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 6586–6592.

[10] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al., "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, 1996, vol. 96, pp. 226–231.