Online meeting analyser for interviews using image processing and NLP

Dr. S Geeta, Rahul Dakshnamoorthy, S Lakshman Rao, S Vaibhave, Balaji M
School of Computer Science & Engineering., Vellore Institute of Technology Chennai, India
Email: geeta.s@vit.ac.in, balaji.m2019@vitstudent.ac.in, lakshmanrao.s2019@vitstudent.ac.in, vaibhave.s2019@vitstudent.ac.in, rahul.dakshnamoorthy2019@vitstudent.ac.in

***Abstract*—The ongoing pandemic has forced us to adapt to online mode of interviews. Most the companies have opted for online interviews.These online meetings prove to be challenging for both the speaker and the listener. It becomes difficult for the interviewers to analyse the interviewees. This is where our project comes into picture. Our aim is to analyse the emotions of the people attending the interview using image processing and deep learning techniques and use this result along with analysis of the audio. NLP and Machine learning techniques are used to determine if the attendees are involved in the meeting or not. Our system provides a detailed analysis of the meeting. Companies can use the results from the system to judge the candidate.**

***Index Terms*—Image Processing, Filters, binary masking, bit operations, CNN, TF-IDF, Navies Bayesian, Speech-to-text, Google Speech-to-text API, Question predicting, Cosine similarity**

I. Introduction

Focus is a Deep learning based system which can perform emotion detection on the video recording of the online interview and classify the audio files of the online interview based on their interactivity. The system is implemented using various libraries and Packages in Python. We start off by collecting our data i.e. video and audio of the online interview recordings from various sources. The next step is to apply various combinations image processing techniques like thresholding, greyscaling etc. We train the CNN model used for emotion detection with each combination and record the output. The next step involves conversion of these recordings to text format and retrieving the timestamps of the words using the Google Speech Recognition API. Then the timestamps of silence i.e. duration for which there is no interaction is obtained using pydub package in Python. After the timestamps are collected, we classify each sentence based on if it is a question or not using StanfordCoreNLP library. The next stage involves classification of the questions based on trivialness and identifying the involvement of the interviewee.

II. Dataset

The FER-2013 is a widely used emotion dataset. The images are labelled with seven emotions: neutral, happy, surprise, sad, fear, disgust, and anger.

The dataset contains 28,000 of training data, 3,500 of validation data, and 3,500 of test data. The images were collected from Google. The images were collected in a way that they vary in the pose, age, and occlusion.

The dataset was created using the Google image search API to search for images of faces that match a set of 184 emotion-related keywords like blissful, enraged, etc. These keywords were combined with words related to gender, age or ethnicity, to obtain nearly 600 strings which were used as facial image search queries. first 1000 images returned for each query were kept for the next stage of processing.

OpenCV face recognition was used to obtain bounding boxes around each face in the collected images.

Human labelers than rejected incorrectly labeled images, corrected the cropping if necessary, and filtered out some duplicate images. Approved, cropped images were then resized to 48x48 pixels.

III. Literature Review

In [1], the paper proposes a method to monitor students behavior and identify various cues in order to understand the students engagement in classroom/ other learning activities. About 50 students from a German University were observed and analyzed during a seminar. Initially data was collected from the students manually (i.e., self-reported data). Then the facial features of the students were detected using Neural Networks. Then Support Vector Regression was used to regress the facial expressions for each student. The individual Vectors and the self-reported data were used for predicting the engagement of the student using a Regression model.

The authors of [2] focused on classifying the attention states into high/medium/low attention. This system uses 2D and 3D data obtained by the kinect one sensor to build a feature set characterizing both facial and properties such as gaze point and body posture, this sensor was fitted into a classroom during lecture, thus obtaining data. The participants

of the experiment were 22 UG students from a university. Five human observers were employed and were asked to estimate the attention level of the students during lecture on the scale 1...5. Along with this a set of behavior reference signals such as writing, yawning, leaning back, and gaze were recorded and used for analysis. The final set of features used for classifying were the following upper body posture; faze gaze point, and facial features. Several different classifiers were used, decision tree, knn for instance. The accuracy of the proposed system was evaluated by comparing the predicted attention level to the reference attention levels provided by human annotation.

In [3], The main aim of this system is to form an automated behavioral analysis framework to automate the process of observing, recording and tabulating the behavioral observations in the classrooms. Hours of classroom footage would be analyzed by software which would extract and divide the video into frames, finally the analysis would be presented with representative behavior examples. First the in-person behavioral observations were split into eight categories, manual observations were made during a lesson in a third-grade class with multiple cameras recording the lesson finally student observations were paired with student-gaze tracking patterns to train a classifier.

The general steps involved in the algorithm were the following:

1) Estimate student head pose
2) Calculate student gaze targets (whiteboard, teacher, and peer)
3) Observe student behaviors
4) Code student behaviors
5) Automated student engagement classified to classify recorded lesson footage.
6) Present the report to the teacher

IV. Proposed System

Once an interview has been concluded and the interview recording has been fetched, the user uploads the meeting file to our application. Through the anvil cloud servers we fetch the file and pass it to our backend i.e python server where we upload it to Google storage. After processing the audio and video file and generating various attributes we present the final report back to the client in the application which contains the following information -

- Bar chart signifying the count of each emotion
- Average response time of the interviewee
- Number of active speakers (in a normal interview setting it should be two, but may vary)
- Number of questions asked
- Percentage of non-trivial questions
- Percentage of interesting questions

The modules which we employ are:

1. Decomposing the file into audio and video

To extract the audio from the file we use the VideoFileClip function from the MoviePy module in python, if the extraction is successful it returns 1. The file is then sent to a frame extractor function, which uses the VideoCapture function available in OpenCV to get a video capture object. It returns a tuple containing the image and a flag signifying the status of the operation, we run this function till the status becomes false.

2. CNN

The core part of our project lies in the CNN module. it involves data pre-processing as well. First the data is split into training, privateTest, and publicTest based on the 'Usage' column and pre-processing is applied onto them (reshaping and normalization).

The model is designed as a sequential model. The Sequential model API is a way of creating deep learning models where an instance of the Sequential class is created and model layers are created and added to it. It is a linear stack of layers.

The first stack layer is a 2D convolution layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.

$$Conv2D(32,(3,3),activation =' relu'$$

Here 32 is the number of filters that this layer will learn from, it also determines the number of output filters in the convolution. The (3,3) signifies the size of the kernel, i.e height and width.

$$MaxPool2D((2,2)))$$

We also use Max Pooling to reduce the spatial dimensions of the output volume.

The 'relu' is the activation parameter which specifies the name of the activation function which we apply after performing convolution.

We also add dense and flatten layers. Dense Layer is used to classify image based on output from convolutional layers. A flatten operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension.

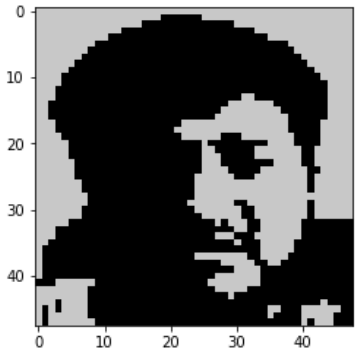
The model is then trained using back propagation for 12 epochs

3. Image processing

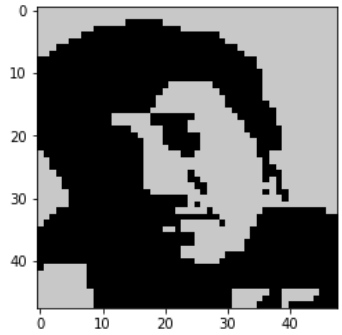
We apply thresholding and grey-scaling to the dataset and train the CNN model using these inputs. Thresholding basically is used for segmenting the background and foreground in an image.

```
.
Epoch 10/12
449/449 [==============================] - 67s 150ms/step - loss: 0.1084 - accuracy: 0.7425 - val_loss: 1.9078 - val_accurac
y: 0.4583
Epoch 11/12
449/449 [==============================] - 68s 150ms/step - loss: 0.0917 - accuracy: 0.7816 - val_loss: 2.1459 - val_accurac
y: 0.4508
Epoch 12/12
449/449 [==============================] - 68s 150ms/step - loss: 0.0776 - accuracy: 0.8108 - val_loss: 2.2424 - val_accurac
y: 0.4486
113/113 [==============================] - 3s 23ms/step - loss: 2.2290 - accuracy: 0.4458
test caccuracy: 0.445806622505188
```

Fig. 1: accuracy of the model



(a) neutral



(b) sad

Fig. 2: Sample input after applying the operations

4. Speech to text

The cloud speech-to-text API from Google enterprise is used in this module. The API converts audio to text by applying powerful neural network models. The API call takes a file path to the resource in Google storage and a set of configurations as its arguments. The corresponding methods to add a file to the Google storage and subsequently remove them are uploadtobucket() and deletefrombucke (). Note We dont store the audio file in any manner to respect the privacy of users. The arguments enablewordtimeoffsets and enableautomaticpunctuation are set to TRUE to get the time stamps of each word and getting the punctuated text respectively.

5. Interesting and difficulty classifier

This module classifies the questions as interesting/not interesting and difficult/not difficult. The data used here was collected through a survey. The data is then transformed as a matrix using the tfidf vectorizer method, the query is also transformed into a vector. Stop words removal, lemmatization and other such techniques are employed for higher performance. Next, we use the cosine similarity to fetch similar sentences and calculate the scores for each query j as follows:

$$Score\ Qj\ = \sum_i cosine\_similairy(Qj, i)\ x\ weight(i)$$

Weight(i) is 1/-1 indicating positive and negative class respectively.

If the computed score is greater than the threshold (0) the query falls into the positive class and negative class otherwise.

6. Question detector

This module detects whether a given sentence is a question or not.

The sentences extracted from the transcript are passed on to the function as input.

Each word in the sentence is then compared with the words present in the NLTK npschat corpus, this corpus consists of ten thousand posts which are PoS tagged and dialogue tagged.

All the words present in the sentence and the class of all the sentences (ie statement/ynquestion/whquestion) obtained from the npschat corpus are used as features for training the Multinomial Nave Bayes Model which is inbuilt in NLTK. The feature set is split in the ration of 80:20 and given as train input and test input to the model. The model classifies the given statement as statement/ynquestion/whquestion based on the feature set. If it classifies as ynquestion or whquestion, the corresponding sentence is flagged as a question.

$$\textbf{TF(i,j)} = \frac{\text{Number of occurrences of token i in document j}}{\text{Total number of words in the document j}}$$

$$\textbf{IDF(i)} = \log \frac{\text{Total number of documents}}{\text{Number of documents consisting of token i}}$$

7. Identifying the number of speakers present in the audio file

The cloud speech-to-text API from Google enterprise is used in this module. The API performs speaker diarization on the audio file using advanced Neural Networks.

We use a function called speech.RecognitionConfig which acc the necessary configuration details corresponding to the

audio file such as encoding, languagecode, diarization configuration, samplehetrzrate etc. The diarization configuration is passed as a speaker diarization config object which contains details like min speaker count, max speaker count etc.

This RecognitionConfig object and the audio file are passed to a function called longrunningrecognize which performs the necessary operations on the file. The return object of this function contains the details of each word like the tag of the speaker etc. from this return object we obtain the number of speakers.

8. Finding the average response time

Once the questions and the word time stamps are ready they are passed to this module where for each question its time stamp and its corresponding response word stamp is fetched.

The word time stamps are a datetime object, subtracting two datetime objects creates a delta datetime object from which the number of seconds can be fetched.

The delta seconds for each question-answer is calculated and added to a global sum and finally the average is calculated.

V. Advantages of the proposed system

1. Our system doesnt disturb the interview in any way.

2. The system provides a comprehensive summary of the interview, giving the interviewer more insights about the candidate

3. The new mode of online interviews are here to stay and this tool would be beneficial for a lot of organizations.

VI. System Architecture

The user first opens the browser and visits our website. The upload option is then used to upload the interview file. Then this file will be sent to the backend i.e python via the Anvil Server.

The recording is then decomposed into separate audio and video files.

The audio is then processed under each module, to extract information such as the No of questions, list of active attendees, the difficulty of question and the average response time. These results are sent back to the Anvil Cloud server which will be displayed in the front end.

The video file is parallelly processed, then thresholding and grey-scaling is applied on the data and the CNN model is trained based on the it, then each frame of the video which differs more than 10 percent from the previous frame is sent for emotion detection, the results are compiled and sent back to the server.

The anvil cloud server are based on full server-side Python environment. The server module is mainly used to execute the code as it is written for full testing such as checking password before changing the file , or secret authentication via third part API.

The file which gets uploaded in the front end is passed to the Anvil server via the @anvil.server.callable function. This data which is obtained from the user is extracted by python via the API request on the server in JSON format.

Then after decoding and performing the necessary processing activities, the results are again passed to the Anvil server via the callable function and an API request on the Anvil server is issued by the front end, which after obtaining the results, displays it.

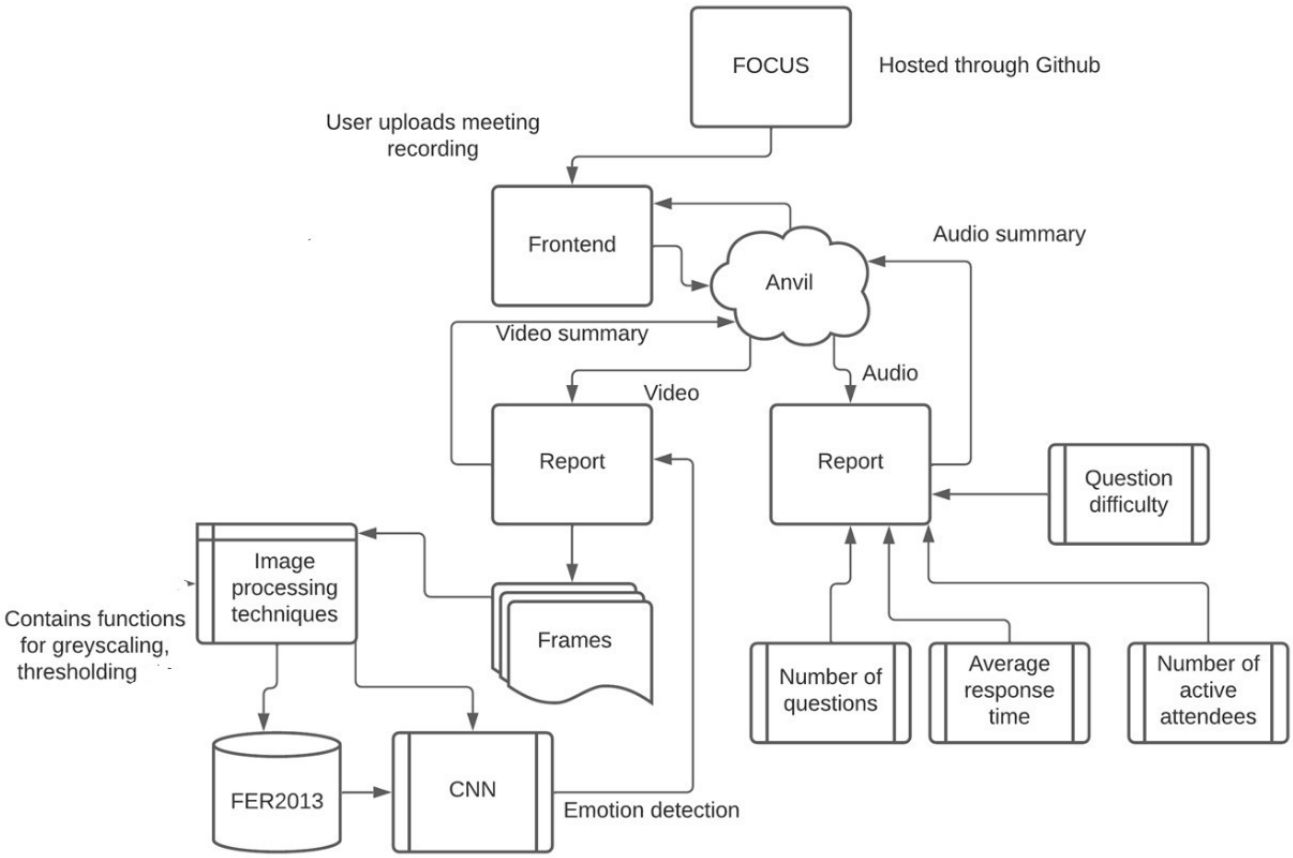


Fig. 3: System architecture

VII. Results

In the Meeting Analyser website, the user has three options : Home, Repository and Audio. In order to upload the meeting file and analyze it, the user must go to the Home directory and to view the source code, the user must click on the Repository option which would direct the page to the GitHub link.

After uploading has finished, the user can perform analysis on the audio component of the meeting file by clicking the Audio option. A bar graph showing the frequency of all emotions shown throughout the interview duration is displayed.

The Neural Network model was trained using the original dataset as well as the datasets obtained after applying all the different combinations of image processing techniques on top of it. Splitting each into train and test sets and validating the models using the testing set gives us the accuracy for each model. Each models accuracy is displayed in tabular format.

After the user clicks on the Analyze option, the number of sentences spoken, the number of questions asked in the interview, the number of difficult and interesting questions, response time and the total number of active speakers are

Fig. 4: Home page, three options

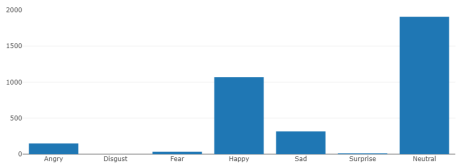


Fig. 5: Frequency of all emotions shown

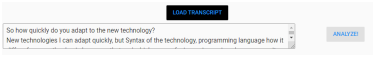


Fig. 6: Audio page, Transcript has been generated for the audio file

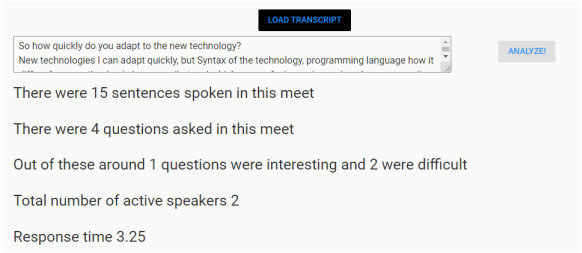


Fig. 7: Results of the question analysis model

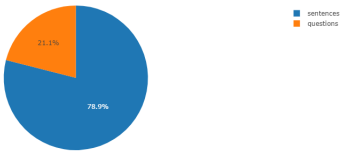


Fig. 8: Distribution of questions in sentences

Fig. 9: Distribution of interesting questions

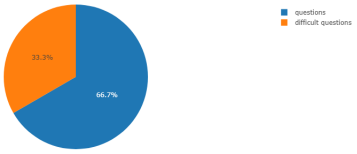


Fig. 10: Distribution of difficult questions

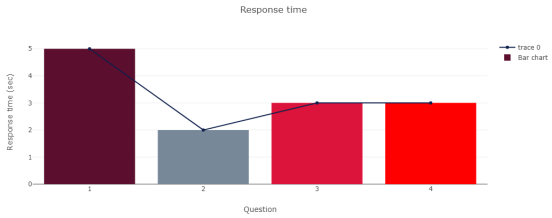


Fig. 11: Response time for each question

displayed[3].

Parallely plots are generated to display the results in a lucid manner.

VIII. Conclusion

Online interviews are here to stay and tools like these would be useful for the future to come. Online meetings are dynamic in nature and cannot be measured with a few attributes however in this paper we used the most feasible attributes with respect to the audio and video of a meeting. The attributes along with supporting pie charts and bar graphs are generated and displayed for the interviewers to asses themselves as well as the candidate more accurately.

References

[1] P. Goldberg, Ö. Sümer, K. Stürmer, W. Wagner, R. Göllner, P. Gerjets, E. Kasneci, and U. Trautwein, "Attentive or not? toward a machine learning approach to assessing students visible engagement in classroom instruction," *Educational Psychology Review*, pp. 1–23, 2019.

[2] J. Zaletelj and A. Košir, "Predicting students attention in the classroom from kinect facial and body features," *EURASIP journal on image and video processing*, vol. 2017, no. 1, pp. 1–12, 2017.

[3] J. Bidwell and H. Fuchs, "Classroom analytics: Measuring student engagement with automated gaze tracking," *Behav Res Methods*, vol. 49, p. 113, 2011.