Multimodal Sentiment Analysis

Abdallah Ahmed, Abdallah Amr, Abdelaziz Hosny, et al.
Computer and Systems Engineering Department
Faculty of Engineering, Ain Shams University
Cairo, Egypt

July 25, 2021

Ain Shams University, Faculty of Engineering

Computers and Systems Engineering Department

Cairo, Egypt

Multimodal Sentiment Analysis

A Report Submitted in Partial Fulfillment of the Requirements of the Degree of Bachelor of Science in Computer and Systems Engineering

Abdallah Ahmed	1600804	Abdallah Amr	1600769
Abdelaziz Hosny	1600763	Abdelrhman Kadry	1600747
Abdelrhman Mohamed	1600750	Ali Mohamed	1600841
Amr Mohamed	1600941	Andrew Raafat	1600331
Mahmoud Swilam	1601306	Mohamed Elsayed	1601129
Mohamed Essam	1601236	Reem Nasser	1600594

Supervised By

Prof. Dr. Ahmed Hassan

Eng. Sarah Abdu

Cairo 2021

Acknowledgment

First of all, we would like to thank our supervisors Prof. Dr. Ahmed Hassan, and Eng. Sarah Abdu, for their guidance and assistance throughout the project. They dedicated their time to help us in the project and to achieve the best results we could afford. So, the outcome and success of this project would never be the same without them and without their technical advice and supervision. We would like to show our gratitude and appreciation for our parents as they have faced a lot of stress with us through the whole year in the pandemic. They offered us a suitable environment to be able to get the best of us. Last but not the least, we should acknowledge each team member as we could not finish this dissertation without our support and dedicating time to each other's whenever anyone faced technical problems throughout the project.

Abstract

Sentiment analysis is a good way to describe the data and analyze it to make use of this data by creating applications that can make our life easier, sentiment analysis is the analysis of the words meaning or the facial expression and returns if this data is positive or negative or neutral, most of the companies use sentiment analysis with one data type like text only or video only or audio only and this method don't give the exact sentiment as the human can get it because the human analyzes the three data types in the same time like if you are talking to someone he can say some bad words but he is Laugh so u get a positive sentiment from him so we used the multi modal sentiment analysis algorithm. We created a website called Sentimeter which is a web application that provides the user to get his own multi modal sentiment analysis model by uploading his data set to the website and chose one of four multi modal sentiment architectures (RAVEN, MULT, MARNN, MMMUBA) the user choses the architecture he or she wants, then the user can create his model and wait for it to finish training and get his model file, then the user can also predict over a new video with his model and know whether the video is positive or negative. To make it easy for the user to split, and transcript the data automatically and labeling the data manually on an excel sheet, we provide the user with a small desktop application the manipulates the data the way we need. All the user needs to do is to install this application and run it then the application will create the data then the user can upload this new data folder to the website to create his model.

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
GloVe	Global vectors for word representation
LSTM	Long Short-Term Memory Network
MSE	Mean Square Error
NLP	Natural language processing
ReLU	Rectified Linear Unit
RNN	Recurrent neural networks

Chapter 1

Introduction

1.1 Motivation

The widespread digital transformation among companies and organizations provided us with a lot of data, but the problem is these data are meaningless and not useful, so we need a way to transform these data into useful information. Here comes the role of sentiment analysis as a vital tool that produces useful insights about data to help the companies better understand the customers and make better decisions.

Not everyone can access a sentiment analysis model specialized for his domain. Typically, you need to be a specialized engineer or hire a company to build the model for you. Here comes our solution: we provide the users a set of tools that uses the users' data to build a sentiment analysis model that can help them make predictions to improve their business and become more competitive.

Most of the applications and companies that deal with sentiment analysis analyze just one type of data, they analyze text data like comments from Facebook or tweets from tweeter or analyze video/ image data or audio data, and sometimes they combine two of these data types but they don't use all three types together in the same time, we provide the user an accurate analysis for their video data that we get from them, we split these video data into the three types of data (text, video, and audio) and combine these types together in one model that takes in consideration the three types and outputs one model that can analyze any video and get the sentiment right just like a human because it analyses the facial expression from the video and the frequency of the voice from the audio and the meaning of the words from the text just like we human do when we talk to each other.

1.2 Current Solutions

Sentiment analysis is one of the most effective ways to measure marketing and branding campaigns. That is why many companies provide sentiment analysis services that allow customers to improve their businesses and make better decisions. We will discuss some of these companies that offer sentiment analysis solutions.

1.2.1 Affectiva

Affectiva is a company that provides solutions for Automotive AI, Media Analytics, and In-Lab Biometric solutions.

We are interested in Media Analytics as it is closer to our solution, The Affectiva Media Analytics solution analyzes facial movements to understand the complexity and nuances of emotions and cognitive states.

Different Companies and organizations hire affectiva to analyze their ads and evaluate if their advertising evoked the intended emotion of their customers and if their emotional response to the ad could predict sales.

Affectiva provides Emotion AI facial tracking tool to identify the emotions and feelings of the watchers while watching the ads tested, they also provide an SDK that helps companies to build a solution to automatically tag different media with emotions and reactions.

Affectiva uses different deep learning architectures designed for facial and vocal analysis tasks. [12]

1.2.2 Azure video Analyzer for media

Azure video Analyzer for media is a cloud application, part of Azure Applied AI Services. It enables users to extract the insights from their videos using Video Analyzer for Media video and audio models.

This service can Identify positive, negative, and neutral sentiments from speech and visual text. It also can identify emotions based on speech.

They provide different ways to access their service, a user can upload his videos into their website and start to analyze these videos, they also provide REST API which lets users integrate their solution into their apps, finally, they provide Embeddable widget that lets users embed the Video Analyzer for Media insights, player, and editor experiences into their apps. [13]

1.2.3 Repustate

Repustat's AI-driven video content analytics model can extract sentiment and brand mentions from videos on YouTube, Facebook, Instagram, TikTok, and any other major channel.

They use Aspect-based sentiment analysis which can easily help distinguish which features of a product or service are liked and which ones can be improved.

A user can access his video analysis over an API or have results displayed to him in an easy-to-use and intuitive dashboard. [14]

1.2.4 Gengo

Gengo is a company that provides a service for sentiment analysis performed by humans. They have a crowd of expert workers who provides a fast and high-quality evaluation of customers' data. Whether text, video, or audio files, they are carefully examined, evaluated, and categorized according to the customer's specific needs. [15]

1.3 Why these solution do not solve the needs of stakeholders?

The previous solutions discussed are suitable for many stakeholders but not all of them, we will discuses how these solutions does not fulfill all the needs of stakeholders.

1.3.1 The missing piece

All previously discussed solution provides almost the same service where the customer's data get analyzed by the service then useful insights are extracted and sent to the customer. But let us think of this scenario where organizations, companies, and even individuals have a data set specialized for a specific domain and want to own a model that makes accurate predictions to future data of the same domain, none of the previously discussed solutions fulfill such requirements and that is the missing piece of their solutions.

1.3.2 Lack of cross-modality

Most solutions deal with one or two types of data when they make a prediction. For example, some solutions extract insights based only on facial movements or the features extracted of an audio. In other words, their models do not support the analysis of all three types of data: visual (face expressions), acoustic (characteristics of talker sound), and textual data (the content extracted from the speech) at the same time to make accurate predictions.

1.3.3 Sentiment analysis performed by humans

Some companies provide a service for sentiment analysis performed by humans. This type of solution has some problems:

- It is not automated and cannot be integrated into a software.
- Not easy to scale.
- Expensive

1.4 Key success indicators

Our solution is unique as we provide what other solutions missed. We will discuss features of our solution that we think it can make our solution successful.

1.4.1 Variety of models

We provide four models with different architectures. The user can choose from them to train with his data. This allows the user to use his data to train various models and pick one with the best accuracy.

1.4.2 Cross modality

Speaker intentions depend on not only verbal behavior but also the non-verbal context. Because of that, our models put the non-verbal context into consideration by splitting the video into three types of data visual, acoustic, and textual. Then extract features from them then fuse these extracted features to get an accurate predictions.

1.4.3 Easy-to-use and intuitive website

We provide a website with a great user interface and user experience. That allows the user to easily create models, upload his data, and get predictions.

1.4.4 A tool for preparing user's data

We also provide an easy-to-use tool that helps the user prepare his data and process it, so the data become usable for model training.

Chapter 2

Theortical Background

In this chapter we will discuss some concepts to better understand our proposed solution.

2.1 Deep learning algoithms

2.1.1 GloVe

It is a pre-trained model for word representation mainly for obtaining vector representations for words which is the embedding matrix. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. Therefore , words having similar meaning are close to each other in the word space representation. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. GloVe does not rely just on local statistics which is position of words in context but incorporates global statistics to obtain word vectors.

2.1.2 Recurrent neural networks

An artificial neural network which models the sequential and time series data. It can handle variable length input sequence. It performs well for sequence data but has short-term memory problem for long sequences. It allows previous outputs to be used as inputs while having hidden states.

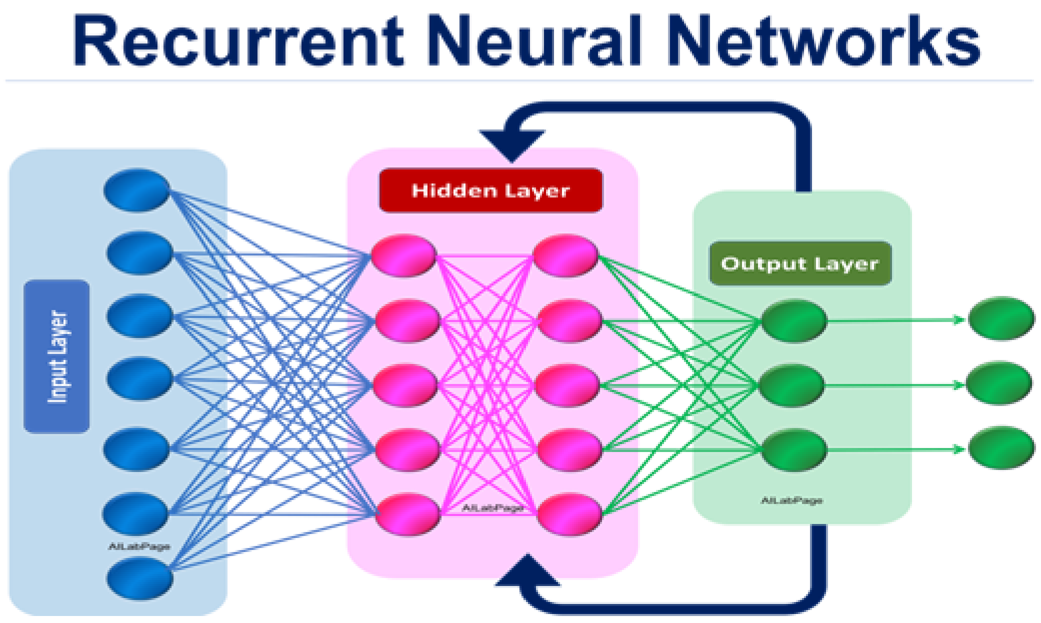


Figure 2.1: RNN architecture [1]

2.1.3 Long Short-Term Memory Networks

An artificial recurrent neural network which has feedback connections. Suitable for long dependent sequence data. LSTM is mainly composed of input gates, output gates, cells and forget gates. The cell remembers values through timesteps, and the three gates regulate the flow of information into and out of the cell. It is suitable for classification and prediction problems.

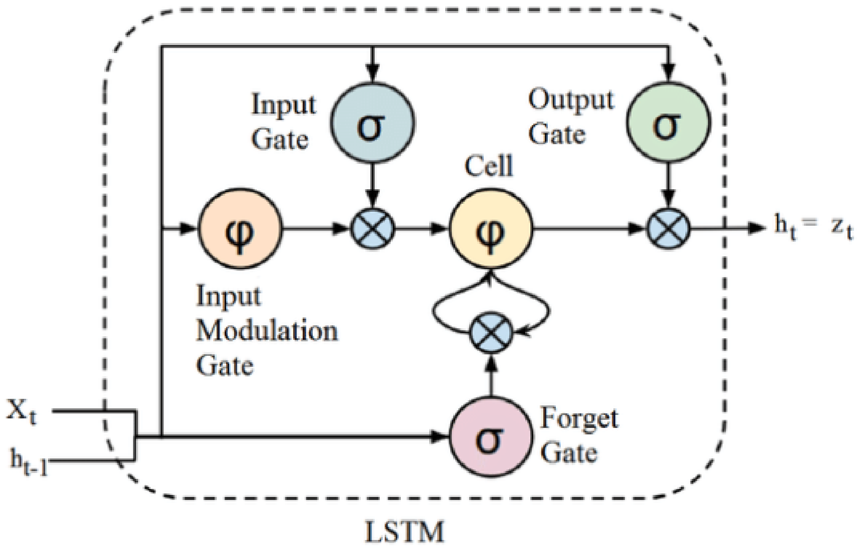


Figure 2.2: LSTM architecture [2]

2.1.4 GRU

GRU'S are a gating mechanism in recurrent neural networks. The GRU is like a LSTM with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate. GRU's performance on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing was found to be like that of LSTM.GRUs have been shown to exhibit better performance on certain smaller and less frequent datasets.

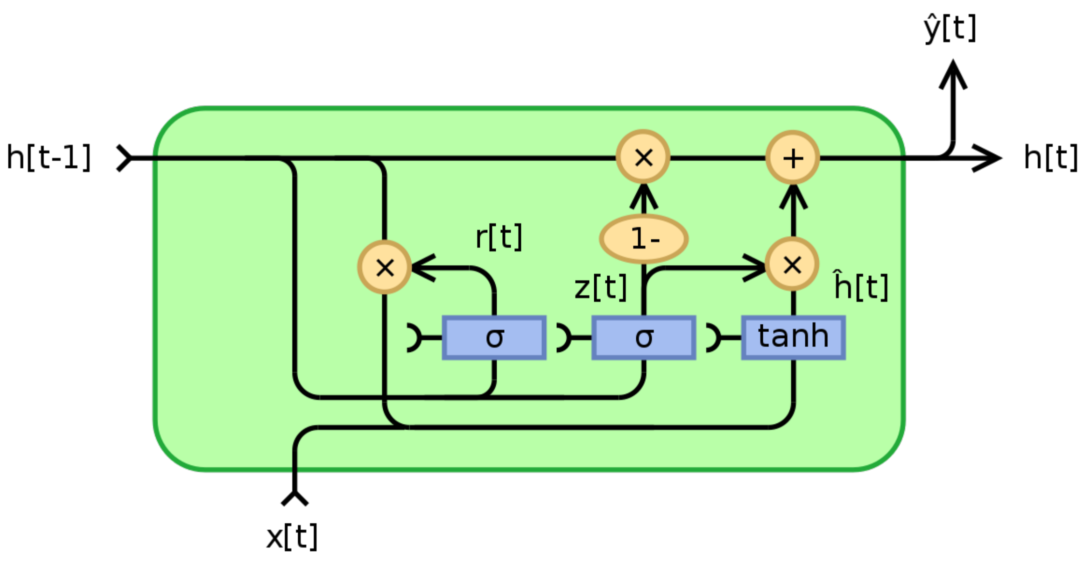


Figure 2.3: GRU architecture [3]

2.2 Activation functions

Activation functions are responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

2.2.1 ReLU

It is a function which will output the input directly if it is positive, otherwise, output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

2.2.2 Sigmoid

The sigmoid function has the property that they map the entire number line into a small range such as between 0 and 1, therefore it is used to convert a real value into a probability (which means it could be used as binary classifier(two-class classification problem)).

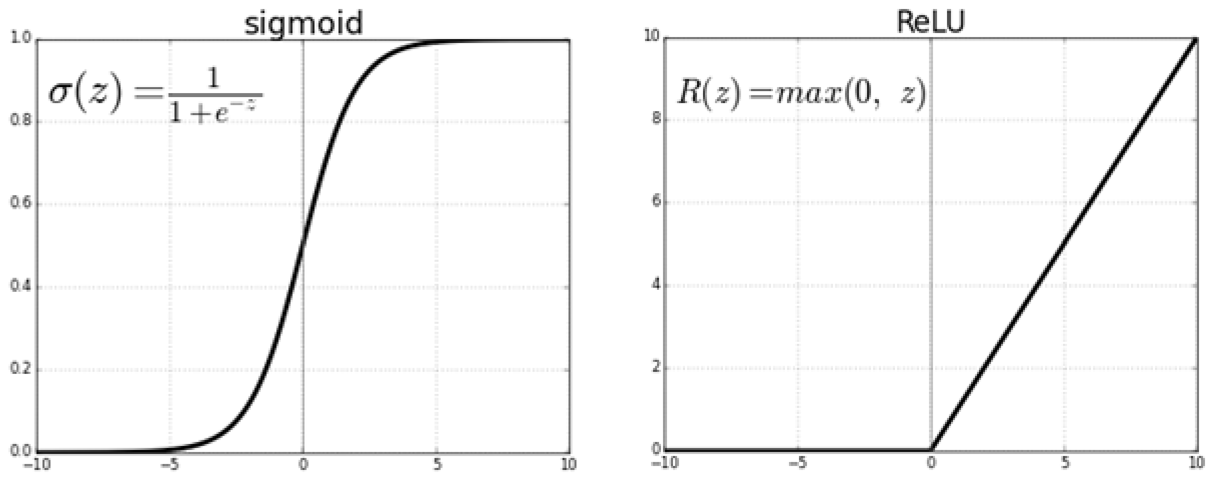


Figure 2.4: ReLU vs Sigmoid plot

2.2.3 Linear

Which is also called identity function and same to linear perceptron in traditional NN. Its mathematical formula is $Y = WX$. It takes the inputs X, multiplied by the weights for each neuron W and creates an output signal Y proportional to the input.

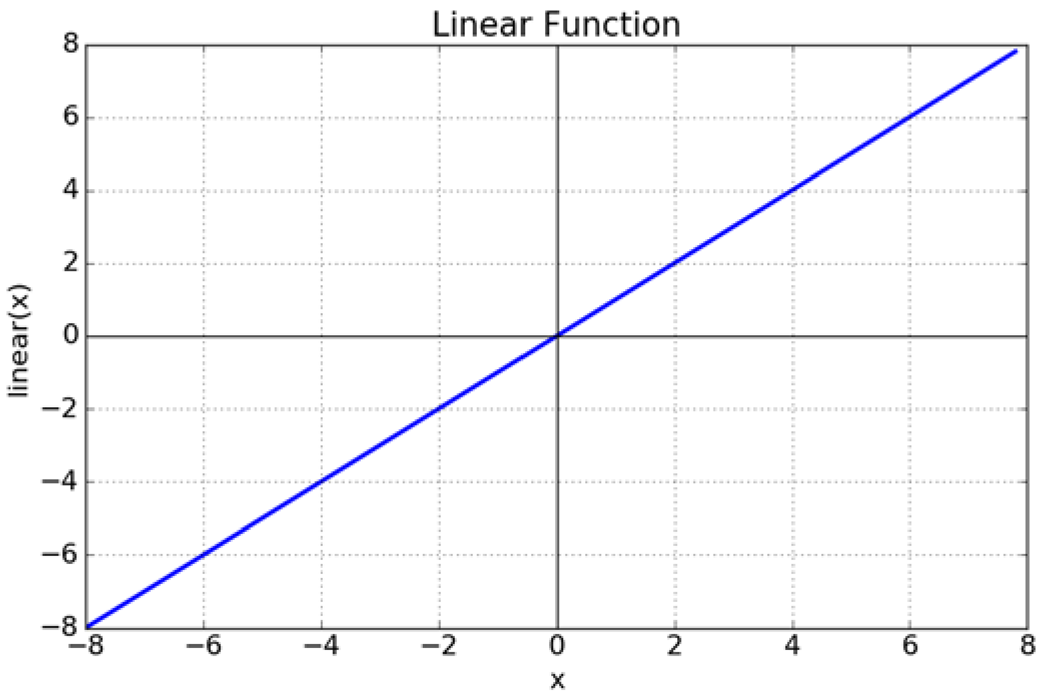


Figure 2.5: linear activation function plot

2.2.4 Tanh

The function takes any real value as input and outputs values in the range -1 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0.

$$Tanh(x) = \frac{\exp(x) - \exp(x)}{\exp(x) + \exp(x)}$$

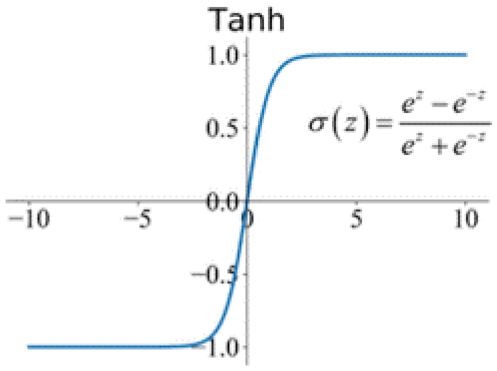


Figure 2.6: tanh function

2.2.5 softmax

it is a mathematical function that converts a vector of numbers into a vector of probabilities, where the probabilities of each value are proportional to the relative scale of each value in the vector. It's most common use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and it is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the SoftMax function is interpreted as the probability of membership for each class.

$$f_i(x) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

2.3 Loss function

2.3.1 Binary cross entropy

Binary cross entropy is a loss function that is used in binary classification tasks which is known as binary classifier. It is often used for a neural network with one output predicting two classes, most of the cases is that positive class membership for 1 and negative for 0 output.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Chapter 3

Methodology

3.1 Dataset

To evaluate our models, we used Multimodal Opinion-level Sentiment Intensity (MOSI) [16] data set which is a total of 93 videos selected from YouTube vlogs videos where speakers talk about different subjects. Each video contains one speaker looking straight at the camera. All videos are manually transcribed and segmented to extract the textual content of the video. Visual and audio features are extracted automatically. All three types of features are aligned. Each segment has been annotated as either positive or negative only (no in between).

3.2 Recurrent Attended Variation Embedding Network (RAVEN)

Humans communicate through both verbal and nonverbal behaviors; Verbal communication is the use of words to convey a message. Some forms of verbal communication are written and oral communication. Nonverbal communication is the use of body language to convey a message. Their intentions change frequently depending on vocal tone, pauses in speech , gestures, and facial expressions. Considering the change in nonverbal contexts in which a word appears during modeling human language. Intentions conveyed through uttering a sentence can display a significant shift in intensity and direction of the uttered words, leading to the phenomena that the uttered words exhibit dynamic meanings depending on different nonverbal contexts. Previous work in modeling human language often utilizes word embeddings pretrained on a large textual corpus to represent the meaning of language. However, these methods are not sufficient for modeling highly dynamic human multi modal language. From here the concept of Multimodal sentiment analysis is presented and discussed in “Words Can Shift: Dynamically Adjusting Word Representations Using Nonverbal Behaviors” paper. Yansen Wang et al. [4] proposed the Recurrent Attended Variation Embedding Network (RAVEN). The main objective for RAVEN architecture is to model human multi modal language through:

- Analyzing the fine-grained structure of the visual and acoustic sub word sequences that exists during the word segment to get the nonverbal representation for each of them.

- Capture the change in nonverbal intentions by dynamic shifting the word representations based on the accompanying nonverbal behaviors.
- Prove the hypothesis that the shifted embeddings learned by RAVEN exhibit meaningful distributional patterns with respect to the sentiment expressed by the speaker.

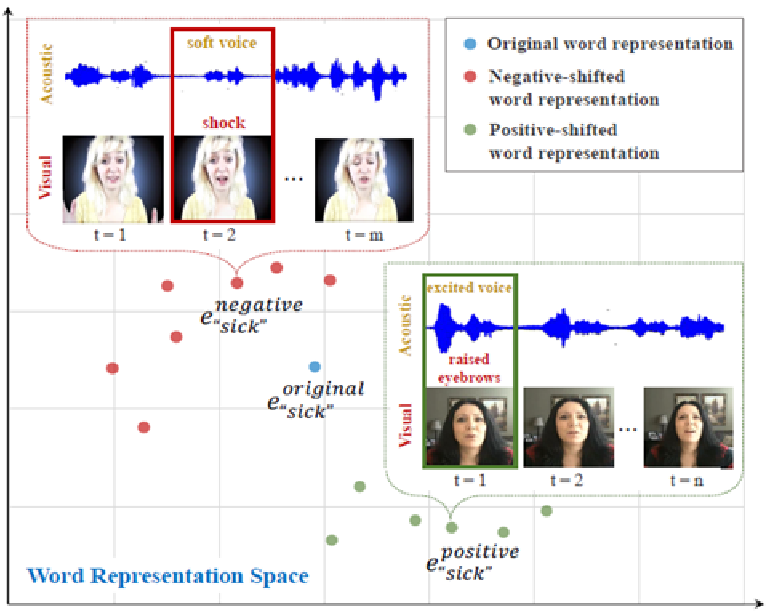

Acoustic
soft voice
shock
Visual
t = 1
t = 2
t = m
Original word representation
Negative-shifted word representation
Positive-shifted word representation
$e^{negative}_{\text{"sick"}}$
$e^{original}_{\text{"sick"}}$
excited voice
raised eyebrows
t = n
$e^{positive}_{\text{"sick"}}$
Word Representation Space

Figure 3.1: Conceptual figure illustrating that the word representation of the same underlying word "sick" can vary conditioned on different co-occurring nonverbal behaviors. The nonverbal context during a word segment is presented by the sequence of facial expressions and tone pitch. The speaker who has a relatively soft voice and frowning behaviors at the second time step displays negative sentiment. While The speaker who has a relatively excited voice and surprise behaviors at the second time step displays positive sentiment. [4]

The example in Figure 3.1 illustrates how the same underlying word can vary in sentiment when paired with different nonverbal context. Although the two speakers are using the same adjective "sick" to describe movies, they are conveying different sentiments and remarks by showing opposing facial expressions and tone pitch. This nonverbal contribution contained in the uttered words, including facial expressions, facial landmarks, and acoustic features are critical in identifying the precise intent expressed in verbal language. We hypothesize that the "exact intent" can often be derived from the representation of the uttered words combined with a shift in the embedding space introduced by the accompanying nonverbal behavior. In this regard, a dynamic representation for words in specific visual and acoustic background is required.

3.2.1 Architecture

RAVEN consists of three main components to achieve the Multimodal shifted word representation for each uttered word in a sentence and one component for the classification task:

1. Nonverbal Sub-networks.
2. Gated Modality-mixing Network.
3. Multimodal Shifting.
4. Transformer Block.

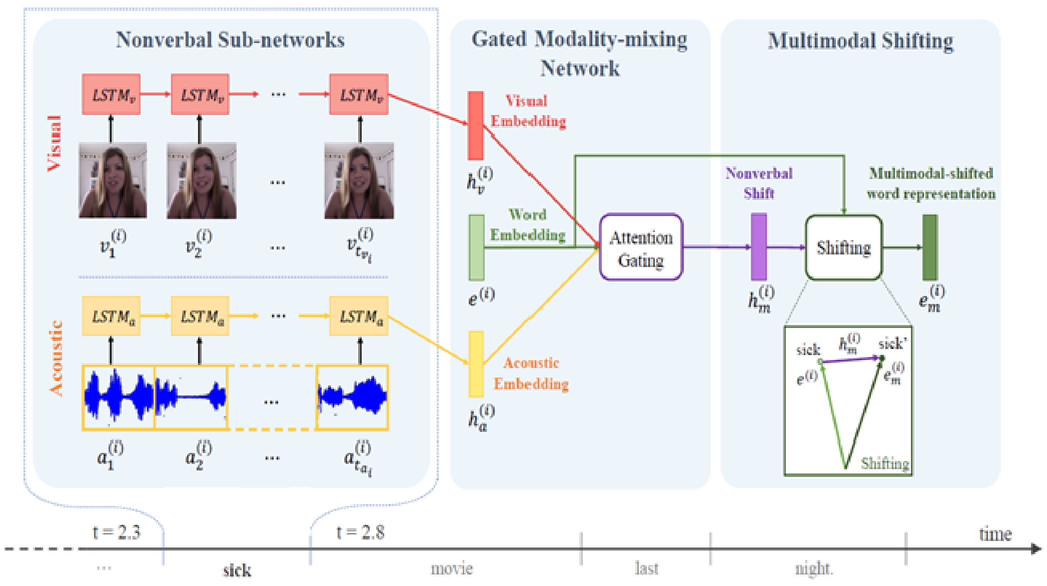


Figure 3.2: Conceptual figure illustrating example for the given word “sick” in the utterance, the Nonverbal Sub-networks first computes the visual and acoustic embedding through modeling the sequence of visual and acoustic features lying in a word-long segment with separate LSTM network. The Gated Modality-mixing Network module then implies the nonverbal shift vector as the weighted average over the visual and acoustic embedding based on the original word embedding. The Multimodal Shifting finally generates the multimodal-shifted word representation by integrating the nonverbal shift vector to the original word embedding. The multimodal-shifted word representation can be then used in the high-level hierarchy to predict sentiments or emotions expressed in the sentence.

3.2.1.1 Non-verbal sub-networks

This component is mainly concerned with the structure of the nonverbal behavior at the sub word level and how to model it. Using two separate RNN which are two separate LSTM networks to encode the temporal sequence of the visual and acoustic patterns within a word segment which are referred as the visual and acoustic sub word units and get the estimate output which is the nonverbal embedding for each visual and acoustic pattern , that will be the representation for nonverbal behaviors in RAVEN.

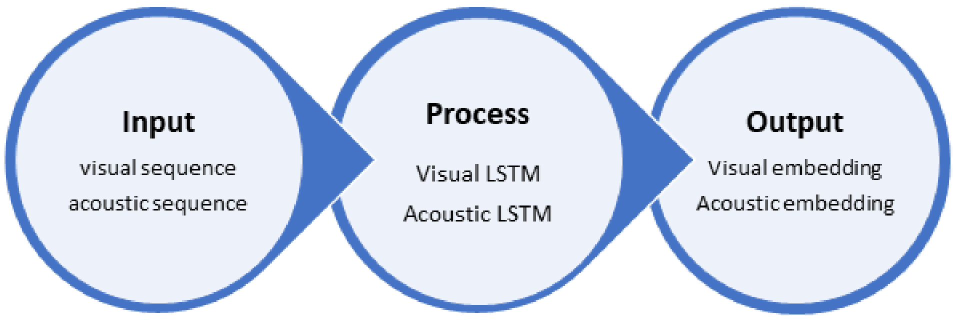


Figure 3.3: The input and output of non-verbal netwroks stage

- The sequence of uttered words: L.
- Certain uttered word in the sequence:$L^{(i)}$
- The accompanying sequence of visual modality within the word $L^{(i)}$: $V^{(i)} = [v_1^{(i)}, v_2^{(i)}, v_3^{(i)}, \ldots, v_{t_{v_i}}^{(i)}]$
- The accompanying sequence of acoustic modality within the word $A^{(i)}$: $A^{(i)} = [a_1^{(i)}, a_2^{(i)}, a_3^{(i)}, \ldots, a_{t_{a_i}}^{(i)}]$
- The visual embedding output : $h_v^{(i)}$
- The acoustic embedding output : $h_a^{(i)}$

The two LSTM'S are applied in parallel to each of the sub-word sequences for each word $L^{(i)}$ in the segment L in language modality for $I = 0, 1, 2, ..., n$.

$$h_v^{(i)} = LSTM\left(V^{(i)}\right)$$

$$h_a^{(i)} = LSTM\left(A^{(i)}\right)$$

3.2.1.2 Gated modality-mixing network

The main objective for this component is to compute the nonverbal shift vector by learning a non-linear combination between the visual and acoustic embedding using an attention gating mechanism. Our key insight is that depending on the information in visual and acoustic modalities as well as the word that is being uttered, the relative importance of the visual and acoustic embedding may differ. For example, the visual modality may demonstrate a high activation of facial muscles showing shock while the tone in speech may be not providing particularly useful or interesting information. To handle these dynamic dependencies, a gating mechanism is used to control the importance of each visual and acoustic embedding. For the model to control how strong a modality's influence is, we use modality-specific influence gates to model the intensity of the influence.

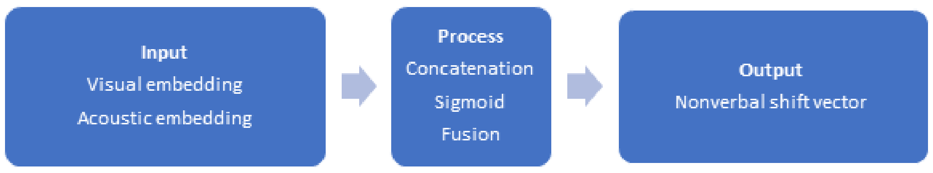


Figure 3.4: The input and output of gated modality-mixing network stage

- Certain uttered word in the sequence:$L^{(i)}$
- The original word representation for the uttered word :$e^{(i)}$
- The visual gate : $w_v^{(i)}$
- The acoustic gate : $w_a^{(i)}$
- weight vector for the visual gate:W_{hv}
- weight vector for the acoustic gate : W_{ha}
- scalar bias for visual gate : b_v
- scalar bias for acoustic gate : b_a
- weight matrix for the visual embedding : W_v
- weight matrix for the acoustic embedding : W_a
- bias vector: b_h
- Nonverbal shift vector : $h_m^{(i)}$

Some processes are applied on the word $L^{(i)}$ with its accompanying nonverbal embeddings; First concatenating its original word embedding $e^{(i)}$ with the visual and acoustic embedding $h_v^{(i)}$,and $h_a^{(i)}$respectively, then for each of the concatenated vector is applied to sigmoid function to get the visual and acoustic gates $w_v^{(i)}$, and $w_a^{(i)}$respectively. . A nonverbal shift vector h$_m^{(i)}$is calculated by fusing the visual and acoustic embeddings multiplied by the visual and acoustic gates.

$$w_v^{(i)} = \sigma\left(W_{hv}\left[h_v^{(i)}; e^{(i)}\right] + b_v\right)$$

$$w_a^{(i)} = \sigma\left(W_{ha}\left[h_a^{(i)}; e^{(i)}\right] + b_a\right)$$

$$\sigma\left(x\right) = \frac{1}{1+e^{-x}}, where\ x \in R$$

$$h_m^{(i)} = w_v^{(i)}.\left(W_v h_v^{(i)}\right) + w_a^{(i)}.\left(W_a h_a^{(i)}\right) + b_h^{(i)}$$

3.2.1.3 Multimodal Shifting

The Multimodal Shifting component learns to dynamically shift the word representations by integrating the nonverbal shift vector into the original word embedding. To ensure the magnitude of the nonverbal shift vector is not too large as compared to the original word embedding, we apply a scaling factor α to constrain the magnitude of the nonverbal shift vector to be within a desirable range. At the same time, the scaling factor maintains the direction of the shift vector.

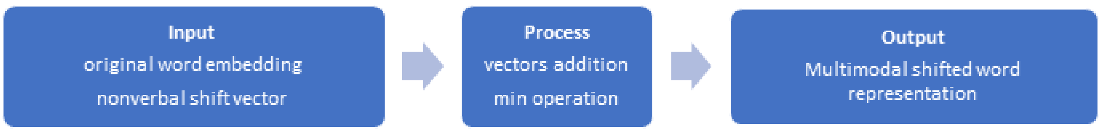


Figure 3.5: Input and output of multi modal shifting stage

- Multimodal shifted word representation for$L^{(i)}$: $e_m^{(i)}$
- Scaling factor to constrain the magnitude of the nonverbal shift vector to be in the desirable range and maintain its direction : α
- Threshold hyperparameter : β
- The new sequence : E

$$e_m^{(i)} = e^{(i)} + \alpha h_m^{(i)}$$

$$\alpha = min\left(\frac{||e^{(i)}||}{||h_m^{(i)}||}\beta, 1\right)$$

Finally , by applying the same method for every word in the segment L , the original temporal sequences are transformed to just one sequence which is the multi modal shifted representations, and that new sequence corresponds to a fusion of the word information and its accompanying nonverbal behaviors and ready to be used in high level hierarchy. The new sequence now is available to be used in our problem to predict multi modal sentiment to the whole utterance using LSTM as proposed in the paper, but we will use transformer encoder-decoder block as it provides more stable accuracy for the model than

LSTM. Output from transformer block h is passed into a fully connected layer to get the sentiment for the data set. $E = [e_m^{(1)}, e_m^{(2)}, e_m^{(3)}, \ldots, e_m^{(n)}]$

$$h = LSTM_e(E) \tag{3.1}$$

$$h = Transformer(E) \tag{3.2}$$

Equation 3.1 as proposed in paper,
and equation 3.2 as used in our project.

3.2.1.4 Transformer Block

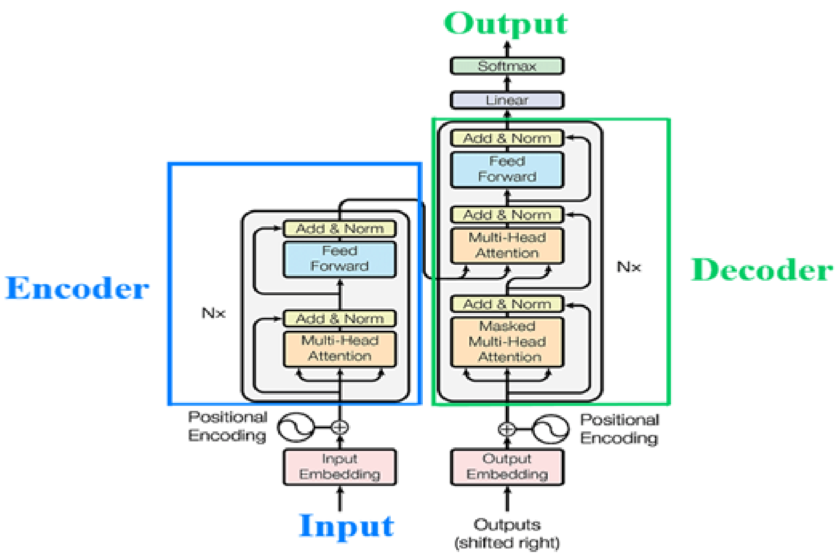


Figure 3.6: Architecture of the Transformer [5]

A transformer is a deep learning model that adopts the mechanism of attention, differentially weighing the significance of each part of the input data. It is used primarily in the field of Natural language processing (NLP). Like recurrent neural networks (RNNs), transformers are designed to handle sequential input data, such as natural language. However, unlike RNNs, transformers do not necessarily process the data in order. Rather, the attention mechanism provides context for any position in the input sequence. For example, if the input data is a natural language sentence, the transformer does not need to process the beginning of the sentence before the end. Rather, it identifies the context that gives meaning to each word in the sentence. This feature allows for more parallelization than RNNs and therefore reduces training times. This parallel processing is not possible in LSTMs or RNNs as they take words of the input sentence as input one by one. The Transformer describes a new way of positioning words (to remember the order of words) by using an explicit position encoding that will be added to the input integrations.

3.2.2 Design

- Goal : Achieving accuracy approximately equals to 78% using word level data set.
- What we have and expect from user : utterance level data set

3.2.2.1 Mission

- Preprocessing on data to get it in the required shape. Speech recognition service or model is required to get the transcript for the audio files and the timestamps of each word uttered in this audio. After getting the timestamps, splitting the video and audio sequences accompanying the uttered word during this time interval.
- Extracting the required features (Textual , Visual, Acoustic).
- Implementing the model to train it with the data set.
- Testing the model to check if it satisfies the requirements and goal or not.

3.2.2.2 Pipeline

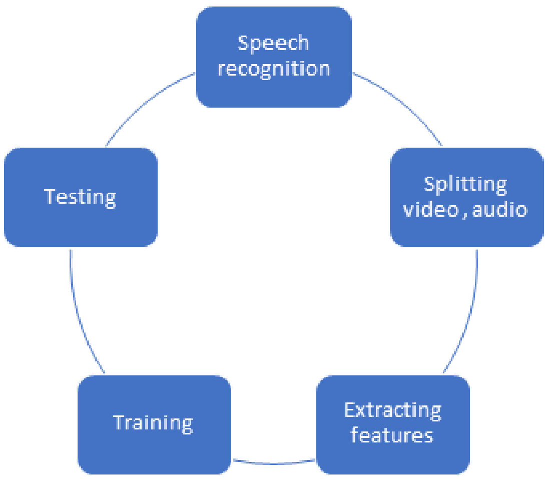


Figure 3.7: Four stages of the pipeline

First step in any training process is to prepare the data to be able to pass to the model. therefore, some preprocessing is required on data to get it in the required shape. The required shape is to have for each word in an utterance the corresponding video frames and audio frames. Speech recognition service or model is required to get the transcript for the audio files and the timestamps of each word uttered in this audio file. After getting the timestamps, splitting the video and audio sequences accompanying the uttered word during this time interval. At this step, we have the data set ready for feature extraction ; the required features are Textual , Visual, and Acoustic features. Implementing the model and using the suitable classifier to train it with the extracted feature. Finally, testing the model to check if it satisfies the requirements and goal or not.

Through each step will define what the operation is done in each file , what is the input and what is the expected output from this process:

1. Splittingdata.py Using Watson API to transcript the audio files and write the text in new text files, getting timestamps to extract the corresponding audio and video frames to each word saved in text file.
   Input: utterance level data (audio, video)
   Output: word level data (audio , video , text)

2. Datapreprocessing.py Loading the labels for the training data set , extracting features functions for training and testing data using the new shaped data.
   Input: word level data (audio, video , text)
   Output: Features for audio , video , and text ; some configurations calculated from the training data to be used with the testing data ; the needed labels for data.

3. AcousticFeatureExractor.py
   Input: audio file
   Output: Acoustic features

4. VisualFeatureExractor.py
   Input: video file
   Output: Visual features

5. TextFeatureExractor.py Getting the embedding matrix for all the words presented in the data set , extracting textual features using the tokenizer.
   Input: text file
   Output: Textual features

6. Raven.py : Containing a class for the model with all the required function as training the model, saving the model , loading the model , getting the accuracy after training, and predicting the sentiment for new uploaded video.

3.2.3 Implementaion

3.2.3.1 Preprocessing

Raven needs to work with word level data set , the mission is to get for each word uttered the corresponding sequence of video frames and audio frames from the original audio files.

1. Getting word timestamps:

Using IBM Watson™ Speech to Text service It provides APIs that use IBM's speech-recognition capabilities to produce transcripts of spoken audio. The service can transcribe speech from various languages and audio formats. In addition to basic transcription, the service can produce detailed information about many different aspects of the audio. It returns all JSON response content in the UTF-8-character set.

Using SpeechToTextV1 API One of the most important returned information is the timestamps for each word spoken in the audio file. By giving the authentication credentials to this method , it enables the service to use. Setting the timestamps parameter in SpeechToTextV1 recognize method to TRUE , the API returns time alignment (start time , end time) for each word spoken. Saving the word with its time alignment in text file for each audio file to prepare it for splitting process.

2. splitting utterances to words:

splitting process is using the timestamps for each word to get the corresponding audio and video file for each word in the original utterance to prepare it for feature extraction process.

Splitting audio file: Using pydub module [17], a module called pydub to work with audio files. pydub is a Python library to work with only .wav files.
By using this library, we can play, split, merge, export and edit ours . wav audio files. Importing the audio file using Audio Segment method to be able to use it. Extracting the needed part by slicing the start and end time from the audio file and exporting this part as a new .wav file.

Splitting video file: Using OpenCV module, It is Open-Source Computer Vision Library used mainly for image processing and It can process videos to identify objects and faces.

Open video file using Video Capture method by giving it the path of video file , calculating total number of frames using CAP_PROP_FRAME_COUNT method , calculating frame rate using CAP_PROP_FPS method , choosing the suitable compatible codec using VideoWriter_fourcc method ,calculating the exact size of each frame using CAP_PROP_FRAME_HEIGHT ,CAP_PROP_FRAME_WIDTH for height and width, respectively. Create Video Writer object using Video Writer method giving it all the configurations needed as frame size the codec used "mp4v" and the frame rate for the new video with its new path.

To start splitting from certain frame number using cv2.CAP_PROP_POS_FRAMES passing it to set method to determine the index of start frame for the new video , writing all the frames from the starting index till reaching the end index to the new file. The frame number is determined by multiplying the frame rate by the timestamps retrieved from the transcript files. Releasing the writer after finishing the creation of the new video file.

3.2.3.2 Feature Extraction

1. Acoustic Features

Using librosa package [18], Librosa is a python package for music and audio analysis. Librosa is basically used when we work with audio data like in music generation(using LSTM's), Automatic Speech Recognition.it is suitable for analyzing and extracting features of an audio signal. Loading audio files in time domain with sampling rate 16 KHZ as its default size is 22 KHZ using load method.

Using the Mel frequency cepstral coefficients (MFCCs) for feature extraction instead of any other extraction method the library offers as it is one of the most important method to extract a feature of an audio signal and is used majorly whenever working on audio signals. The MFCCs of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope so choosing 13 set of features was an optimal choice for the model.

2. Textual Features

using GloVe [19] which is a learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Using the 300-dimensional version trained on 840B tokens2. Therefore, each word is described using 300 features. We use the most 10,000 most frequent words in English, accompanied with all the words obtained from our data set, then we use the combination of both, access GloVe and start extracting the vector representation of each word and save them in the shape of tokens, where each word is identified by an index, and each index is mapped to its vector representation.

Extract_textual_features function is called, and it's given the data directory path. Then we loop on each utterance and extract each word from it and append all the words extracted to a list. Then the most 10,000 most frequent words file is opened, and all the words are also extracted from it.

A tokenizer is then initialized, and it's given the number of words needed. Then we use the tokenizer.fit_on_texts function and give it the words extracted from the data set along with the most frequent 10,000 English words. This method creates the vocabulary index based on word frequency. So, if you give it something like: "The cat sat on the mat." It will create a dictionary like this: (word_index["the"] = 1; word_index["cat"] = 2). After that we use tokenizer.texts_to_sequence and give it the data, and it transforms each word in our utterance to a sequence of integers that maps to the word in our tokenizer. Then pad_sequences is used for padding, so all utterances have the same number of words.

extract_textual_features _using_tokenizer: this function is same as the extract_ textual_features but we pass to it our tokenizer, so if new data needs to be trained, it is trained using our tokenizer that was made earlier to keep the word to index mapping the same.

pretrained_embeddings function: This function's goal is to create our embedding matrix. It takes the path of our GloVe file, the size of the matrix, the number of words and our word to index mapping. First, we loop on each line and store the word and its embedding vector. Then for each word index we start filling our word embedding matrix until all the words are stored in the embedding matrix, then the matrix is returned.

3. Visual Features

Visual feature extraction aims to extract the facial features of our utterances, using a face model detector and a landmarks detector to extract the features of each frame in the video. The first function get_face_detector returns our face model using OpenCV's pre-trained deep learning face detector model shipped with the library (res10_300x300_ssd_iter _140000.caffemodel) 2 files are used:

1. The .prototxt file which defines the model architecture (i.e., the layers themselves)

2. The .caffemodel file which contains the weights for the actual layers.

find_faces function: This function aims to detect the faces present in the frame. It takes arguments: the frame to investigate and our face model and returns the coordinates of the face.

extract_visual_features function: Takes arguments: 1) Split videos directory. 2) Text files directory. Then the face detector model is called along with the landmark model. The landmark model is implemented where we use 98 landmarks.

We loop over each word said and get the video frames in which this word was said. For each frame in the set of frames we detect the face, and for the face detected we start extracting the landmarks of that face using the detect_marks function where it takes the frame, the landmark model and the face as arguments and return the marks of the face. The features then are appended in a list of features for each word. After we loop over all the frames in each word, the list of features of this word is appended in a list. So, at the end we end up with a list of arrays where the number of arrays is the number of the words in the utterance. The data is then padded to the word with the max length of frames. After the extraction of the features of all the words, we start grouping the words by their utterances. A NumPy array of 4 dimensions is then acquired where the dimensions are: (Number of utterances, number of words in each utterance, number of frames in each word, the extracted features).

3.2.3.3 RAVEN model

- Our raven model is implemented in a class-based way, where we defined a class called 'RavenModel'. Implementing the discussed architecture in previous section.

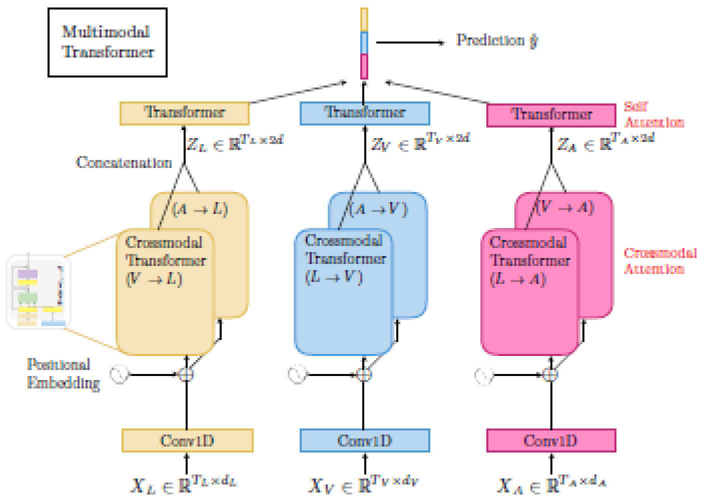
- Creating the classifier for multi modal sentiment as transformer keras layer instead of LSTM
  Transformer Block class :Transformer block does self-attention (which finds the scores for each word to other words in the sentences) and weighted sum it. The transformer is the structure based on nothing but just lots of Dense layers with concepts of residual; however, this makes the time series data losing its time dependence. So, for transformer, it needs to locate the position, which can consider as the additional information for this structure so that it will not miss the time dependence.
  Replacing LSTM with Transformer encoder-decoder block which adopt the mechanism of attention, Transformers are better than RNNs because they do not suffer from short-term memory ; RNN has short reference window, LSTM has larger reference window than RNN while attention mechanism has infinite reference window.

- TokenAndPositionEmbedding class: Positional encoding is added to give the model some information about the relative position of the words in the sentence. As in our model we need the word to be sorted the same as uttered in the original data set. Positional embedding is implemented using a standard keras.layers.Embedding layer, which according to the original paper produced nearly identical results to the sinusoidal version.

3.3 Multimodal Transformer (MulT)

3.3.1 Architecture

The (MulT) model absorbs a strong inspiration from the neural machine translation (NMT) trasnfromer to extend to a multi modal setting. MulT model therefore has no encoder-decoder structure, but it is built up from multiple stacks of pairwise and bidirectional crossmodal attention blocks that directly attend to low-level features (while removing the self-attention). Empirically, Yao-Hung Hubert Tsai et al. [6] show that their proposed approach improves beyond standard transformer on various human multi modal language tasks.

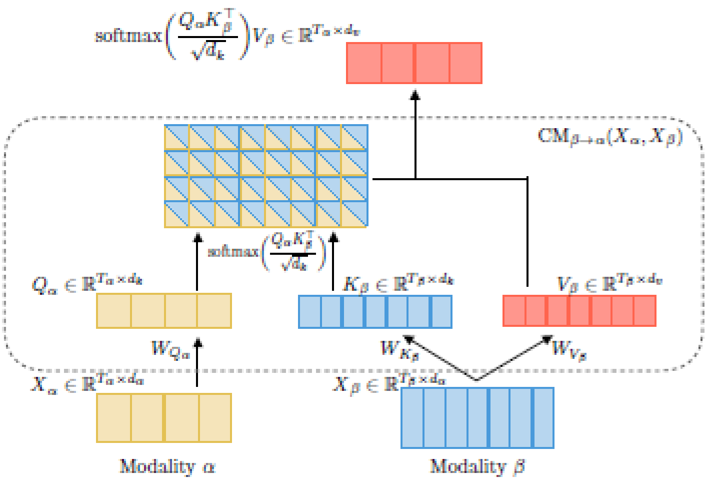


Multimodal Transformer
Prediction $\hat{y}$
Transformer
Transformer
Transformer
Self Attention
$Z_L \in \mathbb{R}^{T_L \times 2d}$
$Z_V \in \mathbb{R}^{T_V \times 2d}$
$Z_A \in \mathbb{R}^{T_A \times 2d}$
Concatenation
$(A \to L)$
$(A \to V)$
$(V \to A)$
Crossmodal Transformer $(V \to L)$
Crossmodal Transformer $(L \to V)$
Crossmodal Transformer $(L \to A)$
Crossmodal Attention
Positional Embedding
Conv1D
Conv1D
Conv1D
$X_L \in \mathbb{R}^{T_L \times d_L}$
$X_V \in \mathbb{R}^{T_V \times d_V}$
$X_A \in \mathbb{R}^{T_A \times d_A}$

Figure 3.8: Overall architecture for MulT on modalities (L, V, A). The crossmodal transformers, which suggests latent crossmodal adaptations, are the core components of MulT for multi modal fusion. [6]

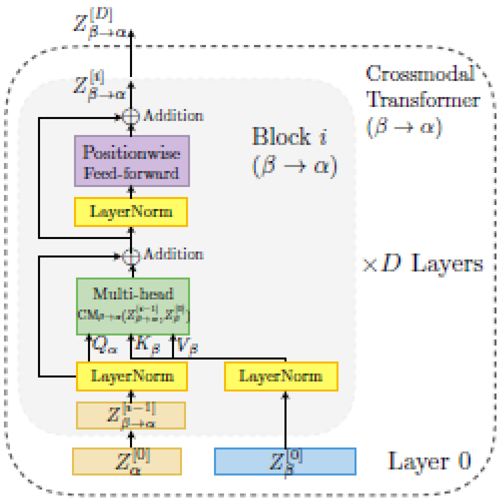
A MulT architecture hence models all pairs of modalities with such crossmodal transformers, followed by sequence models (e.g., self-attention transformer) that predicts using the fused features. The core of our proposed model is crossmodal attention module.

3.3.1.1 Cross-modal attention



$\mathrm{softmax}\left(\frac{Q_\alpha K_\beta^\top}{\sqrt{d_k}}\right)V_\beta \in \mathbb{R}^{T_\alpha \times d_v}$
$\mathrm{CM}_{\beta\to\alpha}(X_\alpha, X_\beta)$
$\mathrm{softmax}\left(\frac{Q_\alpha K_\beta^\top}{\sqrt{d_k}}\right)$
$Q_\alpha \in \mathbb{R}^{T_\alpha \times d_k}$
$K_\beta \in \mathbb{R}^{T_\beta \times d_k}$
$V_\beta \in \mathbb{R}^{T_\beta \times d_v}$
W_{Q_α}
W_{K_β}
W_{V_β}
$X_\alpha \in \mathbb{R}^{T_\alpha \times d_\alpha}$
$X_\beta \in \mathbb{R}^{T_\beta \times d_\alpha}$
Modality α
Modality β

(a) Cross-modal attention $CM_{\beta\to\alpha}(X_\alpha, X_\beta)$ between sequences X_α, X_β from distinct modalities



$Z^{[D]}_{\beta\to\alpha}$
$Z^{[i]}_{\beta\to\alpha}$
Addition
Positionwise Feed-forward
LayerNorm
Addition
Multi-head
$\mathrm{CM}_{\beta\to\alpha}(Z^{[i-1]}_{\beta\to\alpha}, Z^{[0]}_\beta)$
Q_α
K_β
V_β
LayerNorm
LayerNorm
$Z^{[i-1]}_{\beta\to\alpha}$
$Z^{[0]}_\alpha$
$Z^{[0]}_\beta$
Block i $(\beta \to \alpha)$
Crossmodal Transformer $(\beta \to \alpha)$
$\times D$ Layers
Layer 0

(b) A cross-modal transformer is a deep stacking of serveral cross modal attention blocks

Figure 3.9: Architectural elements of a crossmodal transformer between two time-series from modality α and β. [6]

Yao-Hung Hubert Tsai et al. [6] considered two modalities α and β, with two (potentially non-aligned) sequences from each of them denoted $X_\alpha \in R^{T_\alpha \times d_\alpha}$ and $X_\beta \in R^{T_\beta \times d_\beta}$, respectively. For the rest of the report, T and d are used to represent sequence length and feature dimension, respectively. Inspired by the decoder transformer in NMT (Vaswani et al., 2017) that translates one language to another, we hypothesize a good way to fuse crossmodal information is providing a latent adaptation across modalities; i.e., β to α. Note that the modalities consider in the paper may span very different domains such as facial attributes and spoken words.

We define the Querys as $Q_\alpha = X_\alpha W_{Q_\alpha}$, keys as $K_\beta = X_\beta W_{K_\beta}$, and values as $V_\beta = X_\beta W_{V\beta}$, where $\mathrm{W}_{\mathrm{Q}_\alpha} \in R^{d_\alpha \times d_k}$, $\mathrm{W}_{\mathrm{K}_\beta} \in R^{d_\beta \times d_k}$ and $W_{V_\beta} \in R^{d_\beta \times d_v}$ are weights. The latent adaptation from β to α is presented as the cross-modal attention.

$$Y_\alpha = CM_{\beta \to \alpha} \left(X_\alpha,\ X_\beta\right) \in R^{T_\alpha \times d_v}$$

$$\begin{aligned} Y_\alpha &= CM_{\beta \to \alpha} \left(X_\alpha,\ X_\beta\right) \\ &= softmax\left(\frac{Q_\alpha K_\beta^T}{\sqrt{d_k}}\right) V_\beta \\ &= softmax\left(\frac{X_\alpha W_{Q_\alpha} W_{K_\beta}^T X_\beta^T}{\sqrt{d_k}}\right) X_\beta W_{V_\beta} \end{aligned} \tag{3.3}$$

Note that Y_α has the same length as Q_α (i.e., T_α), but is meanwhile represented in the feature space of V_β. Specifically, the scaled (by $\sqrt{d_k}$) softmax in equation (3.3) computes a score matrix softmax (.)$\in R^{T_\alpha \times T_\beta}$, whose (i, j)-th entry measures the attention given by the i-th time step of modality α to the j-th time step of modality β. Hence, the i-th time step of Y_α is a weighted summary of V_β, with the weight determined by i-th row in softmax(.). We call equation (3.3) a singlehead cross-modal attention.

We add a residual connection to the cross-modal attention computation. Then, another positionwise feed-forward sublayer is injected to complete a cross-modal attention block each cross-modal attention block adapts directly from the low-level feature sequence and does not rely on self-attention, which makes it different from the NMT encoder decoder architecture.

3.3.2 Design

Three major modalities are typically involved in multi modal language sequences: language (L), video (V), and audio (A) modalities. We denote with $X_{\{L,V,A\}} \in R^{T_{\{L,V,A\}} \times d_{\{L,V,A\}}}$ the input feature sequences (and the dimensions thereof) from these 3 modalities.

3.3.2.1 Temporal Convolutions

To ensure that each element of the input sequences has sufficient awareness of its neighborhood elements, we pass the input sequences through a 1D temporal convolutional layer:

$$\hat{X}_{\{L,V,A\}} = conv1D\left(X_{\{L,V,A\}}, k_{\{L,V,A\}}\right) \in R^{T_{\{L,V,A\}} \times d} \tag{3.4}$$

Where $k_{\{L,V,A\}}$ are the sizes of the convolutional kernels for modalities $\{L, V, A\}$, and d is a common dimension. The convolved sequences are expected to contain the local structure of the sequence, which is important since the sequences are collected at different sampling rates. Moreover, since the temporal convolutions project the features of different modalities to the same dimension d, the dot-products are admittable in the cross-modal attention module.

3.3.2.2 Positional Embedding

To enable the sequences to carry temporal information, following (Vaswani et al., 2017), we augment positional embedding (PE) to $\hat{X}_{\{L,V,A\}}$:

$$Z^{[0]}_{\{L,V,A\}} = \hat{X}_{\{L,V,A\}} + PE\left(T_{\{L,V,A\}}, d\right) \tag{3.5}$$

where $PE\left(T_{\{L,V,A\}}, d\right) \in R^{T_{\{L,V,A\}} \times d}$ computes the (fixed) embeddings for each position index, and $Z^{[0]}_{\{L,V,A\}}$ are the resulting low-level position aware features for different modalities.

3.3.2.3 Cross-modal transformers

Based on the crossmodal attention blocks, we design the crossmodal transformer that enables one modality for receiving information from another modality. In the following, we use the example for passing vision (V) information to language (L), which is denoted by "$V \rightarrow L$". We fix all the dimensions for$\left(d_{\{\alpha,\beta,k,v\}}\right)$ each crossmodal attention block as d. Each crossmodal transformer consists of D layers of crossmodal attention blocks (see Figure 3.9(b)). Formally, a cross-modal transformer computes feed-forwardly for $i = 1, \ldots\ldots, D$ layers:

$$\begin{gathered} Z^{[0]}_{V\rightarrow L} = Z^{[0]}_{L} \\ Z^{[i]}_{V\rightarrow L} = CM^{[i],mul}_{V\rightarrow L}\left(LN\left(Z^{[i-1]}_{V\rightarrow L}\right), LN\left(Z^{[0]}_{V}\right)\right) + LN\left(Z^{[i-1]}_{V\rightarrow L}\right) \\ Z^{[0]}_{V\rightarrow L} = f_{\theta^{[i]}_{V\rightarrow L}}\left(LN\left(\hat{Z}^{[i]}_{V\rightarrow L}\right)\right) + LN\left(\hat{Z}^{[i]}_{V\rightarrow L}\right) \end{gathered} \tag{3.6}$$

where f_θ is a positionwise feed-forward sublayer parametrized by θ, and $CM^{[i],mul}_{V\rightarrow L}$ means a multi-head (see (Vaswani et al., 2017) for more details) version of $CM_{V\rightarrow L}$at layer i (note: d should be divisible by the number of heads). LN means layer normalization (Ba et al., 2016).

In this process, each modality keeps updating its sequence via low-level external information from the multi-head crossmodal attention module. At every level of the crossmodal attention block, the low-level signals from source modality are transformed to a different set of Key/Value pairs to interact with the target modality. Empirically, we find that the cross-modal transformer learns to correlate meaningful elements across modalities. The eventual MulT is based on modeling every pair of cross-modal interactions. Therefore, with 3 modalities (i.e., L, V, A) in consideration, we have 6 cross-modal transformers in total (see Figure 3.8).

3.3.2.4 Self-Attention transformers and prediction

As a final step, we concatenate the outputs from the cross-modal transformers that share the same target modality to yield $Z_{\{L,V,A\}} \in R^{T_{\{L,V,A\}} \times 2d}$. For example, Z$_L$ $=\left[Z^{[D]}_{V\rightarrow L}; Z^{[D]}{}_{A\rightarrow L}\right]$. Each of them is then passed through a sequence model to collect temporal information to make predictions. We choose the self-attention transformer (Vaswani et al., 2017). Eventually, the last elements of the sequences models are extracted to pass through fully-connected layers to make predictions.

3.3.3 Implementaion

3.3.3.1 Data preparation

Splitting the video into utterances and get the audio of each utterance and the text as well se we have for the same video a folder that have three folders one for the video utterances and one for the audio utterances and one for the transcript.

3.3.3.2 Feature extraction

Textual features: We used GloVe pre-trained word embedding matrix (glove.840B.300d) with a vector dimension of 300.

Visual features: We aim to extract the facial features by using a face detecting model and get the landmarks of the face so we can have insights of the person's facial expression.

Acoustic features: Using the Mel frequency cepstral coefficients (MFCCs) for feature extraction as it is one of the most used methods to extract features from an audio signal The MFCCs of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope.

3.3.3.3 MulT model

We implemented a class called MultModel beginning with a three Conv1D layers one for each modality so they have the same shape then a positional encoding layer for each modality to get some information about the position of the words in the sentence then some encoder layers that have the Multi head attention in it and we perform this encoder layer on two modalities in the same time to get the combination of text to video and audio to video then the video to text and the audio to text the video to audio and the text to audio then we concatenate each two in a concatenation layer concatenate the (video to text) with the (audio to text) and concatenate the (text to video) with the (audio to video) and concatenate the (text to audio) with the (video to audio) then input each one of them to a transformer layer the concatenate all three together and flatten it and feed it to a dense layer with a sigmoid activation function.

3.4 Multimodal Uni-Utterance-Bimodal Attention (MMMU-BA) And Multi-Attention Recurrent Neural Network (MA-RNN)

In this section, we will discuss Multimodal Multi-Utterance-Bimodal Attention (MMMU-BA) and Multi-Attention Recurrent Neural Network (MA-RNN) for performing sentiment analysis on multi modal data. despite each model has different architecture but we were able to design a similar solution for both models in terms of data preprocessing and feature extraction of the data.

3.4.1 Architecture

3.4.1.1 MMMU-BA architecture

In the proposed framework, it aim to leverage the multi-modal and contextual information for predicting the sentiment of an utterance. Utterances of a particular speaker in a video represent the time series information and it is logical that the sentiment of a particular utterance would affect the sentiments of the other neighboring utterances.

To model the relationship with the neighboring utterances and multi-modality, Deepanway Ghosal et al. [7] propose a recurrent neural network based multi-modal attention framework. The proposed framework takes multi-modal information (i.e. text, visual & acoustic) for a sequence of utterances and feeds it into three separate bi-directional Gated Recurrent Unit (GRU) [20] . This is followed by a dense (fully-connected) operation which is shared across the time-steps or utterances (one each for text, visual & acoustic). We then apply multi modal attention on the outputs of the dense layers.

The objectives to learn the joint-association between the multiple modalities & utterances, and to emphasize on the contributing features by putting more attention to these. In particular, Deepanway Ghosal et al. [7] employ bi-modal attention framework, where an attention function is applied to the representations of pairwise modalities i.e. visual-text, text-acoustic and acoustic-visual.

Finally, the outputs of pairwise attentions along with the representations are concatenated and passed to the softmax layer for classification. Deepanway Ghosal et al. [7] called the proposed architecture Multi-Modal Multi-Utterance- Bi-Modal Attention (MMMU-BA) framework.

Multi-modal Multi-utterance - Bi-modal Attention (MMMU-BA) Framework
Assuming a particular video has 'u' utterances, the raw utterance level multi-modal features are rep- resented as $T_R \in R^{u\times 100}$ (raw text), $V_R \in R^{u\times 4096}$ (raw visual) and $A_R \in R^{u\times 88}$ (raw acoustic). Three separate Bi-GRU layers with forward & backward state concatenation are first applied on the raw data followed by the fully-connected dense layers, resulting in $T \in R^{u\times d}$ (text), $V \in R^{u\times d}$ (visual) and$A \in R^{u\times d}$ (acoustic), where 'd' is the number of neurons in the dense layer. Finally, pairwise-attentions are computed on various combinations of three modalities- (V, T), (T, A) & (A, V). In particular the attention between V and T is computed as follows:

- Bi-modal Attention: Modality representations of V & T are obtained from the Bi-GRU network, and hence contain the contextual information of the utterances for each modality. At first, Deepanway Ghosal et al. [7] compute a pair of matching matrices M1 , M2 $\in R^{u \times u}$ over two representations that account for the cross-modality information.

$$M1 = V.T^T \; M2 = T.V^T \tag{3.7}$$

- Multi-Utterance Attention: As mentioned earlier, in the proposed model we aim to leverage the contextual information of each utterance for the prediction. Deepanway Ghosal et al. [7] compute the probability distribution scores ($N1 \in R^{u \times u}$ & $N2 \in R^{u \times u}$) over each utterance of bi-modal attention matrices M1 & M2 using a softmax function. This essentially computes the attention weights for the contextual utterances. Finally, soft attention is applied over the multi-modal multi-utterance attention matrices to compute the modality-wise attentive representations (i.e. O1 & O2).

$$N1(i,j) = \frac{e^{M1(i,j)}}{\sum_{k=1}^{u}(e^{M1(i,j)})} for\; i,j = 1,2,\ldots\ldots,u \tag{3.8}$$

$$N2(i,j) = \frac{e^{M2(i,j)}}{\sum_{k=1}^{u}(e^{M2(i,j)})} for\; i,j = 1,2,\ldots\ldots,u \tag{3.9}$$

$$O1 = N1.T \;\&\; O2 = N2.V \tag{3.10}$$

- Multiplicative Gating & Concatenation: A multiplicative gating function following [21] is computed between the multi-modal utterance specific representations of each individual modality and the other modalities. This element-wise matrix multiplication assists in attending to the important components of multiple modalities and utterances.

$$A1 = O1 \,\textbf{.}\, V \;\&\; A2 = O2.T \tag{3.11}$$

$$MMMU - BA_{VT} = concat[A1, A2] \tag{3.12}$$

- $MMMU - BA_{AV}$ & $MMMU - BA_{TA}$ *computations*: Similar to $MMMU - BA_{VT}$, Deepanway Ghosal et al. [7] follow the same VT procedure to compute$MMMU–BA_{AV}$ &$MMMU–BA_{AT}$. For a data source comprising of raw visual (VR), acoustic (AR) & text (TR) modalities, at first, they compute the bi-modal attention pairs for each combination i.e. $MMMU - BA_{VT}$, $MMMU - BA_{AV}$ & $MMMU - BA_{TA}$. Finally, motivated by the residual skip connection network [7] , they concatenate the bi-modal attention pairs with individual modalities (i.e. V, A & T) to boost the gradient flow to the lower layers. This concatenated feature is then used for final classification.

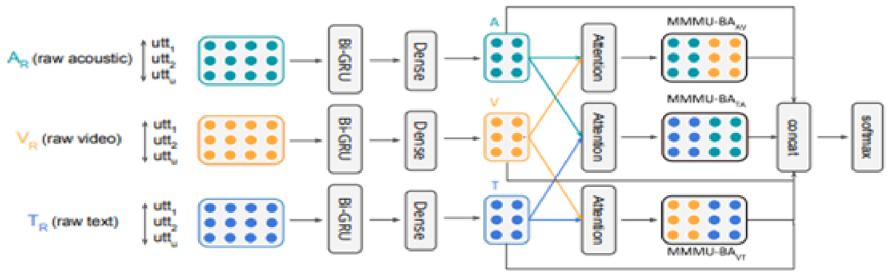


Figure 3.10: Overall architecture of the proposed MMMU-BA framework. [7]

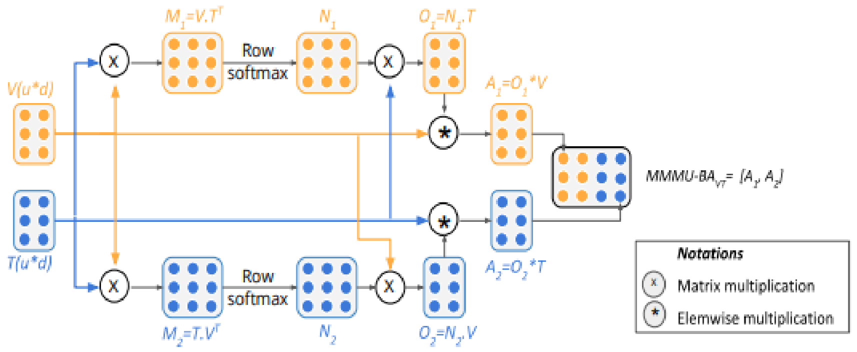


Figure 3.11: $MMMU - BA_{VT}$ attention computation. [7]

Multi-Modal Uni-Utterance - Self Attention (MMUU-SA) Framework MMUU-SA framework does not account for information from the other utterances at the attention level, rather it utilizes multi-modal information of single utterance for predicting the sentiment. For a video having 'q' utterances, 'q' separate attention blocks are needed, where each block computes the self-attention over multi-modal information of a single utterance.

Let $X_{UP} \in R^{3\times d}$ is the information matrix of the pth utterance where the three 'd' dimensional rows are the outputs of the dense layers for the three modalities. The attention matrix $A_{up} \in R^{3\times d}$ is computed separately for, p = 1st , 2nd , ... qth utterances.

Finally, for each utterance p, A_{up} and X_{up} are concatenated and passed to the output layer for classification. Please refer to the appendix for more details. Let $X_p \in R^{3\times d}$ is the information matrix of the p th utterance where the three 'd' dimensional rows are the outputs of the time-distributed dense layer for the three modalities.

Computation in the p th attention block proceeds as follows:

$$M_{up} = X_{up}.X_{up}^T \tag{3.13}$$

$$N_{up}(i,j) = \frac{e^{Mup(i,j)}}{\sum_{k=1}^{3}(e^{Mup(i,j)})} for\ i,j = 1,2,3 \tag{3.14}$$

$$O_{up} = N_{up}.X_{up} \tag{3.15}$$

$$A_{up} = O_{up}.X_{up} \tag{3.16}$$

The attention matrix Aup ∈ R 3×d is computed separately for, p = 1 st , 2 nd , ... q th utterances. Finally, for each utterance p, Aup and Xup are concatenated and passed to the output layer for classification.

Multi-Utterance - Self Attention (MU-SA) Framework In MU-SA framework, Deepanway Ghosal et al. [7] apply self attention on the utterances of each modality separately, and use these for classification. In contrast to MMUU-SA framework, MU-SA utilizes the contextual information of the utterances at the attention level. Let,$T \in R^{u\times d}$ (text), $V \in R^{u\times d}$ (visual) and $A \in R^{u\times d}$ (acoustic) are the outputs of the dense layers.

For the three modalities, three separate attention blocks are required, where each block takes multi-utterance information of a single modality and computes the self attention matrix. Attention matrices A_t , A_v and A_a are computed for text, visual and acoustic, respectively. Finally A_v, A_t , A_a, V , T & A are concatenated and passed to the output layer for classification.

$$M_v = V.V^T \tag{3.17}$$

$$N_v(i,j) = \frac{e^{Mu(i,j)}}{\sum_{k=1}^{u}(e^{Mu(i,j)})} for\ i,j = 1,2,.....u \tag{3.18}$$

$$O_v = N_v.V \tag{3.19}$$

$$A_v = O_v.V \tag{3.20}$$

3.4.1.2 MA-RNN architecture

The model consists of two attention layers and a BiGRU. The first attention layer is used to fuse multi modal data and reduce dimensionality. To capture the key parts of the contextual information among utterances, an attention-based BiGRU was implemented. The combination of these two networks constitute the overall structure of the MA-RNN model.

Multimodal Data Fusion The dimensions of the extracted unimodal features of utterances(d_t, d_a, d_v) where d_t is for text, d_a is for audio, and d_v is for video, were equalized using a fully-connected layer of size d. Then, by concatenating the unimodal features, a multi modal feature $D \in R^{d\times 3}$ was generated as given below:

$$D = [d_t, d_a, d_v] \tag{3.21}$$

Scaled Dot-Product Attention for Multimodal Data The network uses multi modal data consisting of text, audio, and video as its input. The first attention layer fuses these heterogeneous data and reduces dimensionality. It is applied to the multi modal feature vector D, the concatenation of the individual unimodal features as in Figure 3.12.

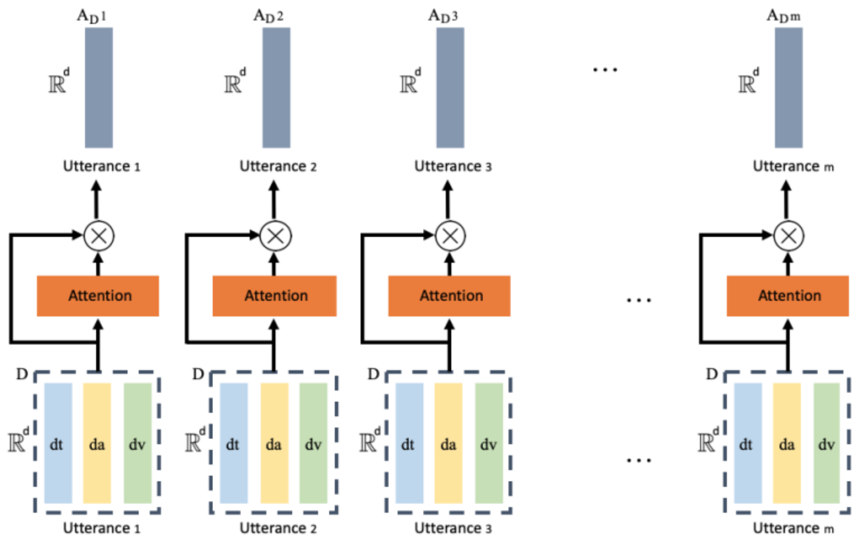


Figure 3.12: Scaled Dot-Product Attention for Multimodal Input. [8]

The attention-based multi modal data fusion mechanism is the scaled dot-product attention. The equation for the scaled dot-product in the network is as follows:

$$A_D = D\, softmax(\frac{\tanh(W_F\, D)^T\, w_F}{\sqrt{d}}) \tag{3.22}$$

where $W_F \in R^{d' \times d}, D \in R^{d \times 3}, w_F \in R^{d' \times 1}, and\, A_D \in R^{d \times 1}$. Here, W_F and w_F are parameters to be learned during training, d is the size of the fully-connected layer used in a process of the feature extraction, and $d\prime$ is a dimension of an attention weight vector. The output,A_D, produces an attention score for each modality. The dimensions of d' and d, it achieved the best performance when d' was set to $4d$ and use $d = 100$ and $d\prime = 400$ in the model.

Multi-head Attention Multi-head attention can be beneficial in terms of helping a model to use the data from different representation subspaces at different positions. The result of multi-head attention$w_a \in R^{d' \times 1}$ was derived by summing the matrix multiplication of w_F and w_F^T as follows:

$$w_a = sum(w_F\, w_F^T) \tag{3.23}$$

where $w_F \in R^{d' \times r}$ is the concatenated attention and r is the number of multi-heads. Therefore, to enable our attention layer to have multiple representation subspaces at different positions, the multi-head attention was applied. However, they used $w_a \in R^{d' \times 1}$ as calculated using Eq. (3.23) instead of $w_F \in R^{d' \times 1}$ in Eq. (3.22). It achieves the best performance when r is set to 150.

Attention-based BiGRU Since the sequential utterances in a video are temporally and contextually dependent, understanding inter-utterance relationships and finding contextual evidences for sentiment classification of target utterance are crucial. To capture the key parts of the sequential data, recent works have been augmenting LSTM/GRU models with the attention mechanism. The model achieved a 6.2 % improvement with GRU compared with LSTM.

Gated Recurrent Unit(GRU) network When M utterances are given in a video, the input $X \in R^{d \times M}$ can be represented as$[x_1, x_2, ..., x_M]$ where $x_t \in R^d$ for $t = 0\, to\, M - 1$. Each cell in GRU can be computed as follows:

$$h_t^j = (1 - z_t^j)\, h_{t-1}^j + z_t^j \tilde{h_t^j} \tag{3.24}$$

$$z_t = \sigma\,(W_z\, x_t + U_z\, h_{t-1}) \tag{3.25}$$

$$\tilde{h^t} = tanh\,(W x_t + r_t\, \odot\, U H_{t-1}) \tag{3.26}$$

$$r_t = \sigma\,(W_r x_t + U_r h_{t-1}) \tag{3.27}$$

where $W_z, W_r, U_z, U_r \in R^{d \times d}$ are the parameters to be learned during the training, σ is the sigmoid function and $\odot$ is element-wise multiplication. The output of Eq. (3.24) is connected to the BiGRU layer that obtains temporally and contextually-aware utterance H where $H = [\overrightarrow{h_t}\, \overleftarrow{h_t}]$ and $H \in R^{2d \times M}$.

Attention network To augment the BiGRU such that it captures the key parts of the contextual information in the utterances, an attention network is implemented as shown in Figure 3.13.

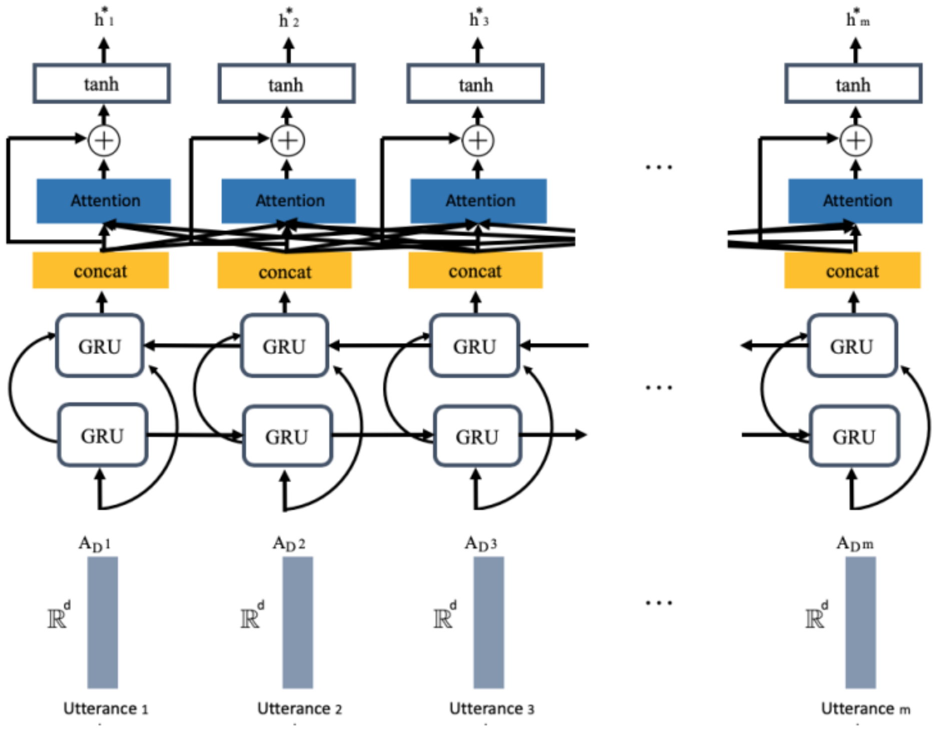


Figure 3.13: Attention-based Bidirectional GRU. [8]

An attention weight vector α_t for utterance information represented by h_t at time t is calculated as follows:

$$p_t = tanh\,(W_h[t]H + b_t) \tag{3.28}$$

$$\alpha_t = softmax(p_t^T w_t) \tag{3.29}$$

$$r_t = H\alpha_t \tag{3.30}$$

where $W_h[t] \in R^{2d \times 2d}$,$b_t \in R^{2d}$,$p_t \in R^{2d \times M}$,$w_t \in R^{2d \times 1}$, $\alpha_t \in R^{M \times 1}$, and $r_t \in R^{2d \times 1}$. A projected vector, p_t, was multiplied with a randomly initialized weighting vector w_t to obtain a weighted hidden representation r_t. Our final BiGRU representation with the attention mechanism for t^{th} utterance was obtained as follows:

$$h_t^* = tanh\,(r_t + h_t) \tag{3.31}$$

where $h_t^* \in R^{2d}$.

Classification For sentiment classification, the output of the BiGRU cell, h_t^*, was fed into a softmax layer.

$$z_t = softmax(W_{soft}[t]h_t^* + b_{soft}[t]) \tag{3.32}$$

$$\hat{y}_t = \underset{j}{argmax} j(z_t[j])\ \forall j \in class \tag{3.33}$$

where $W_{soft}[t] \in R^{ydim \times 2d}$, $b_{soft}[t] \in R^{ydim}$, $z_t \in R^{ydim}$, $ydim$ = number of classes, and $\hat{y}_t$ is the predicted class.

3.4.2 Design

3.4.2.1 Features Extraction

We first extract the features from the Raw videos splitted into small utterances each of length 3 to 8 seconds, then for each utterance we extract visual features from the utterance video, we extract text features from the transcript of each utterance and we extract acoustic features from the audio of each utterance.

The details of the features extraction from each modal is as follows.

Visual features extraction We used the C3D [9] pre-trained model which were trained on Sports-1M data set [22] with necessary tools for extract video features.C3D have been used as a feature extraction technique.C3D are deep 3-dimensional convolutional neural networks with a homogenous architecture containing 3 x 3 x 3 convolutional kernels followed by 2 x 2 x 2 pooling at each layer. They are trained on a large scale supervised video data set such as Sports 1M. The key properties that make it stand against traditional pre-trained models such as ResNets, AlexNet are the following:

1. Generic feature extraction: The 3D convolutions extracts both spatial and temporal components relating to motion of objects, human actions, human-scene or human-object interaction and appearance of those objects, humans and scenes. Thus it is not limited to appearance representation as in ResNets or AlexNet. This makes it a very generic video feature representation for various video related tasks such as action localization, and event detection without the need for fine-tuning for each task.

2. Compact representation: The C3D model is given an input video segment of 16 frames (after downsampling to a fixed size which depends on data set used) and the outputs a 4096-element vector. It gives you a compact representation of a video segment which can be further fed to either a temporal network for action recognition.

3. Fast, efficient inference: Due to its homogenous architecture with small kernel sizes of 3 x 3 x 3, it can facilitate for optimized implementation of these architectures for embedded platforms. This can have huge implications for smart camera environments where processing of real-time feed and extraction of meta data is critical for surveillance and smart city applications.

The network architecture contains 8 convolutional, 5 pooling layers and 2 fully connected layers as shown in Figure below.

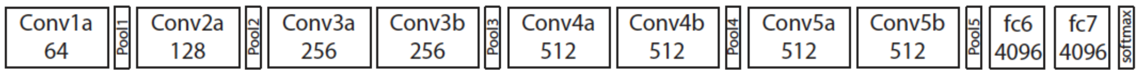


Figure 3.14: C3D Architecture [9]

For simplicity, from now on they refer video clips with a size of c × l × h × w where c is the number of channels, l is length in number of frames, h and w are the height and width of the frame, respectively.they also refer 3D convolution and pooling kernel size by d×k ×k, where d is kernel temporal depth and k is kernel spatial size.

Videos are split into non-overlapped 16-frame clips which are then used as input to the networks. The input dimensions are 3 × 16 × 112 × 112.

The first convolution layer of size 1x3x3 followed by a pooling layer of size 1x2x2. This is to preserve the temporal information in the first layer and build higher level representation of the temporal information at subsequent layers of the network. Every other convolution layer and pooling layer would have a size of 3x3x3 and 2x2x2 where the strides are 1 and 2 respectively. The fully connected layers have a size of 4096 dimensions with softmax outputs reflecting the 487 classes of the Sports 1M data set.

C3D is a very common video segment descriptor which embeds both motion and temporal characteristics. Thus, C3D can be used as a feature extractor for other video analysis tasks. To extract C3D feature, a video is split into 16 frame long clips with a 8-frame overlap between two consecutive clips. These clips are passed to the C3D network to extract fc6 activations. These clip fc6 activations are averaged to form a 4096-dim video descriptor. We refer to this representation as C3D video descriptor/feature.

Text features extraction We trained a CNN model on the Sentiment140 [10] data set with 1.6 million tweets where the tweets have been annotated (0 = negative, 4 = positive) and they can be used to detect the sentiment of a text, and then the model is used for features extraction from text data.

Twitter messages have many unique attributes, which differentiates from other datasets:

1. Length: The maximum length of a Twitter message is 140 characters. From our training set, we calculate that the average length of a tweet is 14 words or 78 characters. This is very different from the other sentiment classification data set

that focused on classifying longer bodies of work, such as movie reviews, which is not suitable for our case.

2. Language model: Twitter users post messages from many difierent media, including their cell phones. The frequency of misspellings and slang in tweets is much higher than in other domains.

3. Domain: Twitter users post short messages about a variety of topics unlike other sites which are tailored to a speciflc topic. This difiers from a large percentage of past research, which focused on speciflc domains such as movie reviews.

First we have to preprocess the sentences as follows:

1. Remove unecessary words and charachters: Users often include Twitter usernames in their tweets in order to direct their messages by including the @ symbol before the username (e.g. @alecmgo). So we have to remove those usernames and the @ symbol. Also users very often include links in their tweets like \http://tinyurl.com/cvvg9a". So we have to remove those URLs.

2. Tokenize the sentence: After cleaning the sentences, we have to tokenize each word in the sentence by creating a vocabulary of words, It is important to define a vocabulary of known words. The more words, the larger the representation of documents, therefore it is important to constrain the words to only those believed to be predictive. We have used the vocabulary from the pretrained embeddings word2vec on Google News data set (about 100 billion words), and if a word is not found in the vocabulary, we add it to the vocabulary with random embeddings.

3. Sentence to index: After creating the vocabulary, for each sentence in the input data set we have to convert it to a sentence of indexs instead of sentence of words, so that the model could train on it.

The model architecture is as follows:-

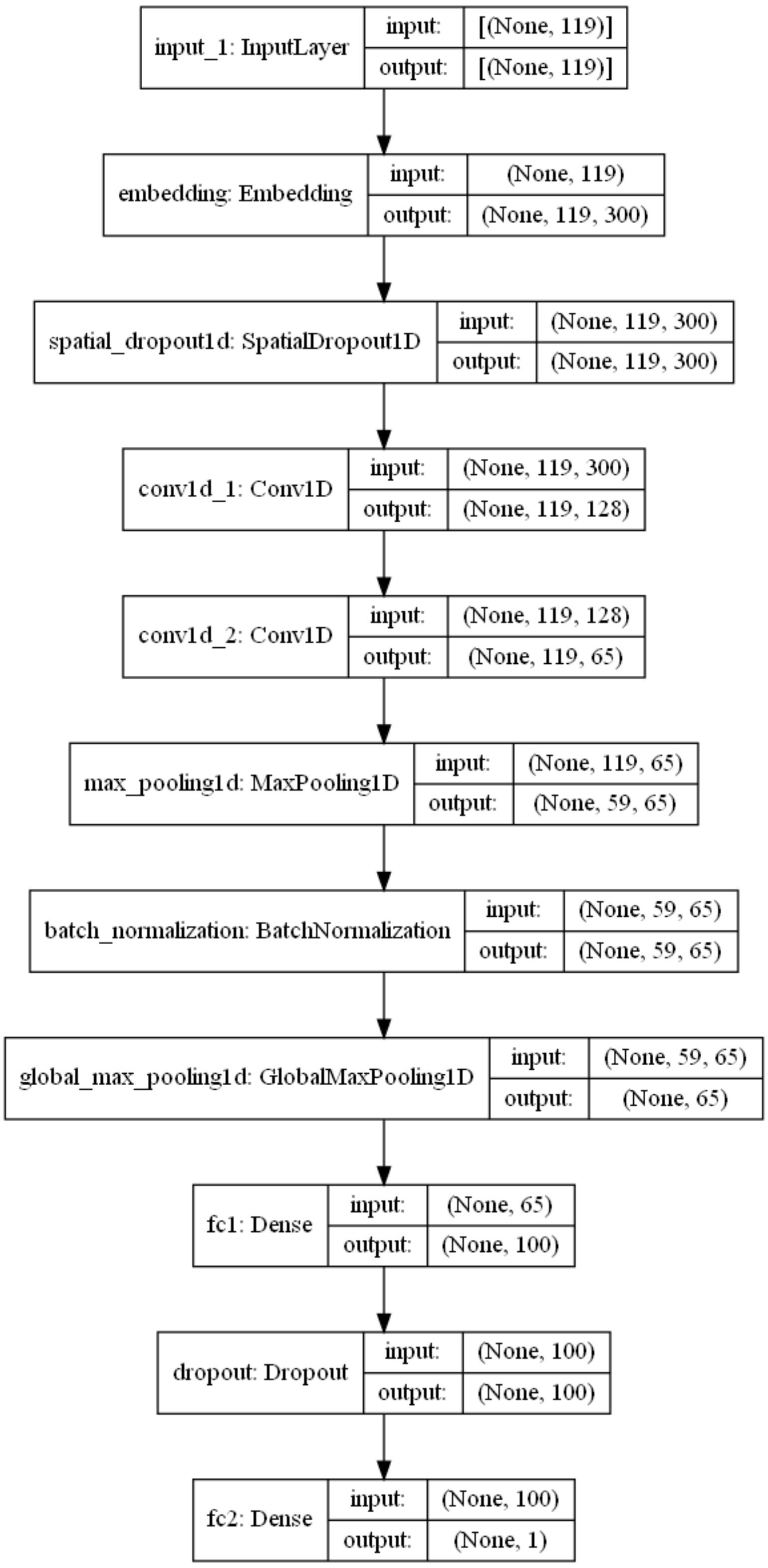


Figure 3.15: CNN architecture [10]

Input layer takes as input a sentence with max length of 119 words, so its shape is (no of samples , 119). Embedding layer with input dimension equal the number of words in the vocabulary and the output dimension is 300 which is pretrained word embeddings from the Google News word2vec pretrained word embeddings. A dropout layer applied on the output of the embedding layer with probability of 0.4 . Two 1D convolution layers of dimensions (f x k) where f is the number of kernel filters and k is size of the kernel , the first with dimensions of (128 x 7), the other with dimensions of (65 x 7) , followed by a max pooling layer with dimensions of (2). A batch normalization layer followed by a Global max pooling layer where it downsamples the input representation by taking the maximum value over the filters dimension which is similar to flattening. A dense fc1 layer with 100 neurons with relu activation followed by a dropout layer with probability of 0.3 followed by another dense layer fc2 with sigmoid output for classification.

Then after training, the first dense layer fc1 with 100 neurons is used for feature extraction of a given sentence.

Acoustic features extraction The Munich Open-Source Media Interpretation by Large feature-space Extraction (openSMILE) Framework is a modular and flexible feature extractor for signal processing and machine learning applications.

The primary focus is clearly put on audio-signal features. However, due to their high degree of abstraction, openSMILE components can also be used to analyse signals from other modalities, such as physiological signals, visual signals, and other physical sensors, given suitable input components.

It is written purely in C++, has a fast, efficient, and flexible architecture, and runs on common platforms such as Linux, Windows, MacOS, Android and iOS. openSMILE is designed for real-time online processing but can also be used off-line in batch mode for processing of large data-sets.

This is a feature rarely found in other tools for feature extraction. Most of related projects are designed for off-line extraction and require the whole input to be present. openSMILE can extract features incrementally as new data arrives.

By using the PortAudio library, openSMILE supports platform-independent live audio input and live audio playback, enabling the extraction of audio features in real-time.

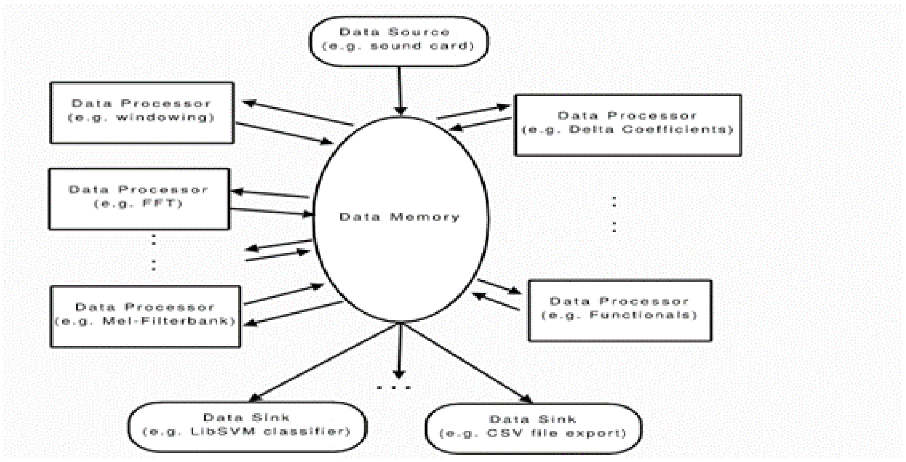

Data Source (e.g. sound card)
Data Processor (e.g. windowing)
Data Processor (e.g. Delta Coefficients)
Data Processor (e.g. FFT)
Data Memory
Data Processor (e.g. Mel-Filterbank)
Data Processor (e.g. Functionals)
Data Sink (e.g. LibSVM classifier)
Data Sink (e.g. CSV file export)

Figure 3.16: Overview on openSMILE's component types and openSMILE's basic architecture. [11]

This figure shows the overall data-flow architecture of openSMILE, where the data memory is the central link between all dataSource, dataProcessor, and dataSink components.

openSMILE toolkit [11] , was used to extract audio features. Audio features were provided with a 30 Hz frame rate and sliding window of 100 ms; then, to identify samples with and without voice, voice normalization was performed using Z-standardization. The features provided by openSMILE contained several low-level descriptions (LLD), such as MFCC, pitch, voice intensity, and their statistics, e.g., mean and the root mean square values. We input the audio utterance into OpenSmile feature extractor with extension (.wav,.mp3,or any audio extension) then we save the output of the audio features into a csv file, this csv file will contain our 88 features then we concatenate it with the utterance label in order to use it in our training. we loop over all of the utterances and save them all into the same csv file. example:- if we have 2199 utterances we take all these 2199 utterances and extract the 88 audio features from them and save them into one csv file. This csv file will have 2199*88 data.

3.4.2.2 Features preprocessing

In features preprocessing we created a script which takes the input files (CSV files of text, audio, video features and labels) and output a preprocessed features (pickle files) in order to ease the use of the features given to our models.

Firstly, this script takes each of the csv files of audio or video or text features separately, it takes the name and the number of each utterance of each video separately, then it takes the number of this utterance between the other utterances of all the videos, then it takes the maximum number of utterances of a single video in order to make a mask for all the videos which have fewer number of utterances than this video utterances example:

If we have 2 videos the video number one have 6 utterances and the video number two has 5 utterances, we mask the video number two in order to have 6 utterances like the video number one. This makes it easier for our model to deal with data all have the same number of utterances.

We apply this method for both training and testing data. Then we save a pickle file for each model (text, audio, video) containing:

1. 3d array of training data (videos, utterances, features)
2. Array of training labels
3. 3d array of testing data (videos, utterances, features)
4. Array of testing labels
5. Maximum length of utterances in a single video within the whole videos
6. Number of videos in training
7. Number of videos in testing

3.4.3 Implementaion

3.4.3.1 MMU-BA implementation

We have used in our implementation Tensorflow and Keras frameworks in additional to Numpy we wrote some functions in order to implement this paper:

1. calc_test_result: which calculates the test accuracy and it can give the confusion matrix and classification report

2. create_one_hot_labels: this function creates one hot label for the test and training labels, which means it makes two columns one for the positive label and one for the negative label

3. create_mask: this function returns mask for the train and testing data

4. bi_modal_attention: this function makes bi-modal attention which is part of MMMU-BA architecture

5. self_attention: this function is used to make self-attention which is used in MMUU-SA and MU-SA

6. contextual_attention_model: this function contains the whole architecture of the three models (MMMU-BA, MMUU-SA, MU-SA) its input is the model's name which we want to work with and its output is the final model

7. train: this function is used to train our model we gave it internally some hyperparameters such as (#epochs, batch size, number of runs and seed) and it returns the results for our training

3.4.3.2 MA-RNN implementation

We implemented the model on a Python 3 enviroment, using the Tensorflow 2 framework. We also used the openCV library to preprocess the utterances videos before extracting the features from them.

For the model implementation using Tensorflow, we implemented the Scaled dot product Attention layer by treating the term "$\frac{tanh(W_f.D)}{\sqrt{d}}$" as a neural network for each utterance "u" with one layer, input layer $x = D[u]$ with dimensions $d = 3x100$, output layer with dimension $d' = 3x400$ and a tanh activation funtion which results to a neural network weights W_f of dimensions $dxd' = 100x400$, and that by using a TimeDistributed Dense layer from Tensorflow.

Then we treat the output of the previous neural network "$\frac{tanh(W_f.D)}{\sqrt{d}}$" with "$w_f$" as the term "$Softmax(\frac{tanh(W_f.D).w_f}{\sqrt{d}})$" as another neural network for each utterance with one layer, input layer $x = \frac{tanh(W_f.D)}{\sqrt{d}}[u]$ with dimensions $d' = 3x400$, output layer with dimensions $=3x150$ and a Softmax activation function which results to a neural network weights w_f of dimension $d'xr = 400x150$, and that also by using a TimeDistributed Dense layer from Tensorflow.

Then to implement the Multi-head attention layer, we extracted w_f as the weights from the second Dense layer, then we multiplied "w_f" with "w_f^T " and summing their product to get "w_a", then we multiply "w_a" with the output of "$\frac{tanh(W_f.D)}{\sqrt{d}}$" for each utterance and then we concatenate their output.

Finally, we multiply the last multiplication output with the input features to get the attention scores.

The AttentionBased BiGRU layer is implemented also using both the Bidirectional GRU layer and TimeDistributed Dense layers using Tensorflow.

Training After the preprocessing of the features, we trained the model on the features of each modal using the Adam optimizer with $\beta1 = 0.9$ and $\beta2 = 0.999$, and applied dropout to the output of each sublayer with the rate of 0.2. The learning rate of 0.0001 was selected with the batch size of 20. The training was completed after 100 epochs. It gets an accuracy of 72.27% on the test set of the CMU-MOSI data set.

3.5 Dataset Creator Tool

assembling data is such a hard and frustrating process which puts limitation on machine learning models training, imagine having to not only assemble this data but also to split, transcript, convert from one form into another, labeling the data manually on an excel sheet. Its a hard, long, and inapplicable approach. We have considered these problems in our application and came up with a tool called data set creator which helps in the automation of the mentioned process so that user has only to feed data into it, wait for splitting, transcripting and audio extraction to complete, and then label his data with the most user friendly way by only clicking a few buttons.

3.5.1 Obstacles and Methodology to overcome

Our models MULT,MARNN,MUBBA,RAVEN all share one thing in common, multi modal data is required, so a full video must be split into utterances and audio, video utterance, transcription extracted from it so these three types of data is then passed to our preprocessing to extract features then to our model architecture to train or predict on.

The very first obstacle was that splitting on full sentences so that information isn't lost when we split the video from certain time to another, so we've used "split on silence" techniques, yet another problem was that we had to provide the silence time to this technique, so we've implemented a function that estimates the mean silence time for each video using the psychoacoustic model.

We've observed that an audio wave settles at a certain level when speaker is silent and peaks occur on talking, so most of the signal lies in a specific range except for the peaks that represent talking. The goal was to get all intervals where signals lied in that range then calculate the mean of these intervals, but we had first to know the threshold of this range which by nature starts from 0 dB.

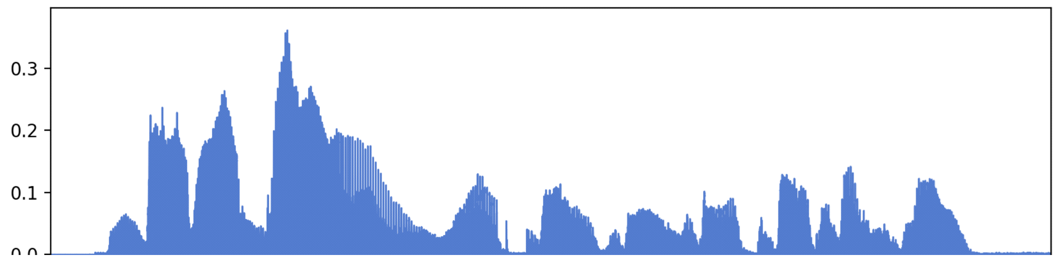


Figure 3.17: Audio wave sample

Another function objective was to calculate the silence threshold for each video independently using the signal distribution to know where most of this signal lies which is we already know represents silence. These helper functions return values are passed to "split on silence" and segments is retrieved along with each segment start, end time.

Audio segments are saved in a separated directory, start and end time are used to split the video to obtain the video utterance, and audio segment is sent finally to "Google-Cloud-Speech" to generate transcription for each segment and save it along with the video, audio utterances.

3.5.2 Usage and UI

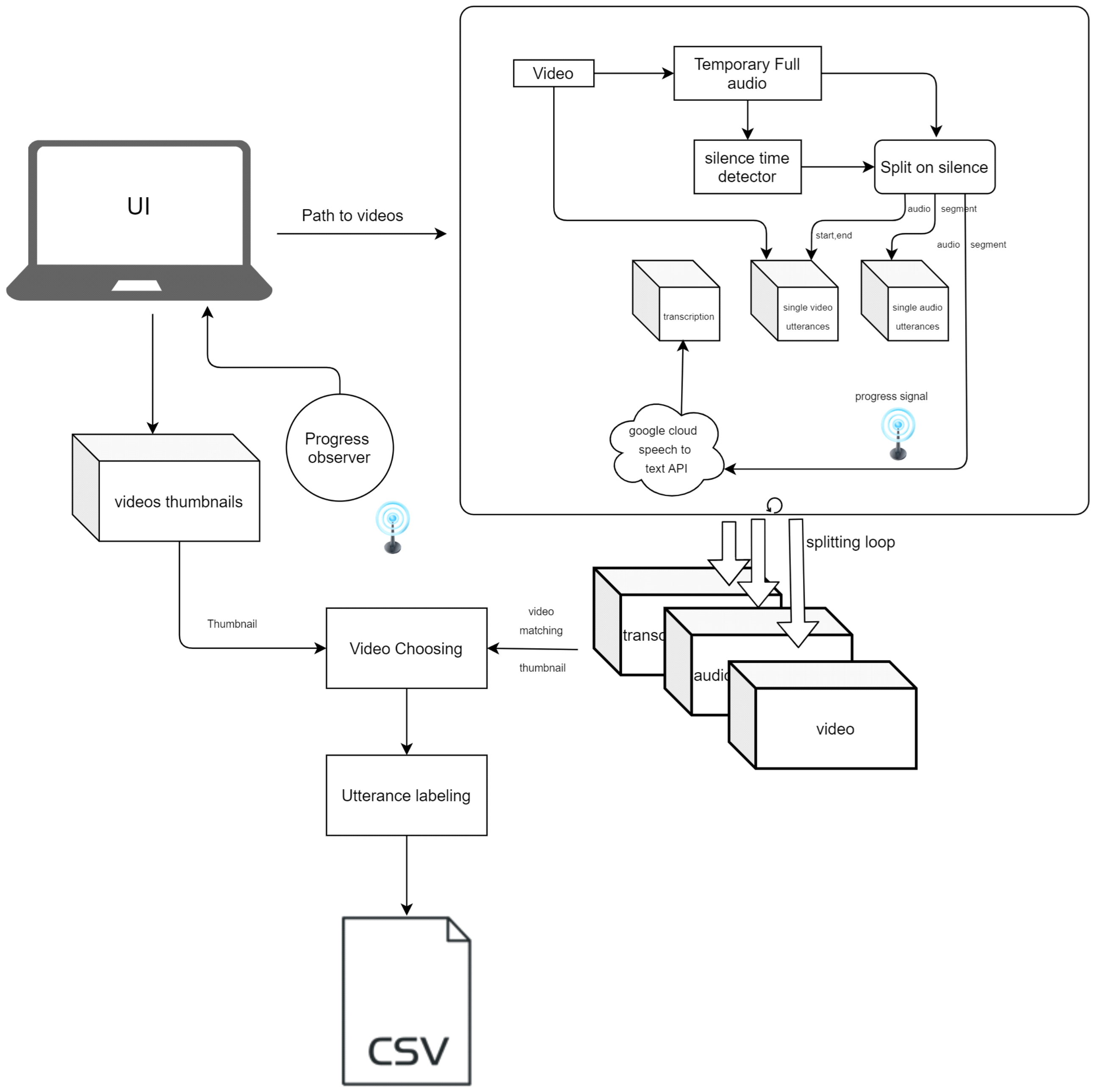


Figure 3.18: Data creator tool pipeline

As shown in figure 3.18 the user provides the tool with the directory of the videos then for each video a thumbnail photo and temporary audio file is generated. Temporary audio file is fed into "split on silence" block along with the silence time detected from it. "split on silence" results in audio chunk along with start, end time to be used to split the video itself, and then this chunk is sent in a request to google-speech-to-text to generate transcription. A signal is emitted from the thread on which split is occurring to update a progress bar for the user to know the progress of the whole process.

After all videos are split, and the three directories of utterances, audio utterances, transcriptions are generated, the user chooses which video he wants to label using the thumbnail generated earlier. After that and through a media player in our GUI the user watches the utterance and label it by clicking on a user-friendly buttons, then he go back and do the same to other videos, at the end he just click "save CSV" and a CSV is generated where each utterance has a label of positive "1" or negative "0".

3.5.3 Conclusion

This tool is aimed to be user's easy way to interact with our web application, so it provides all needed functionality and makes everything ready for model creation process on our web app, so that Multimodal Sentiment Analysis becomes easier than ever.

3.6 Website

We provide a website with an easy and intuitive user interface and user experience that allows the user to upload his data, choose one of the four models previously discussed to be trained by the user's data, and make predictions on new data.

3.6.1 Analysis and requirements

We want to satisfy the requirements of website that allow users to signup into our site and store their information in the database system then allow users to log in and verify his information, so the user is able to (create, like, bookmark) a model or more, then the user is able to upload his data set to server container and send a request to the server to start training the model using his generic data set, at this time model status for this user is "training" until training process finishes then model status become ready to use after finishing the training process time, a ".h5 file" for the model must be saved and user can store it if he wants.

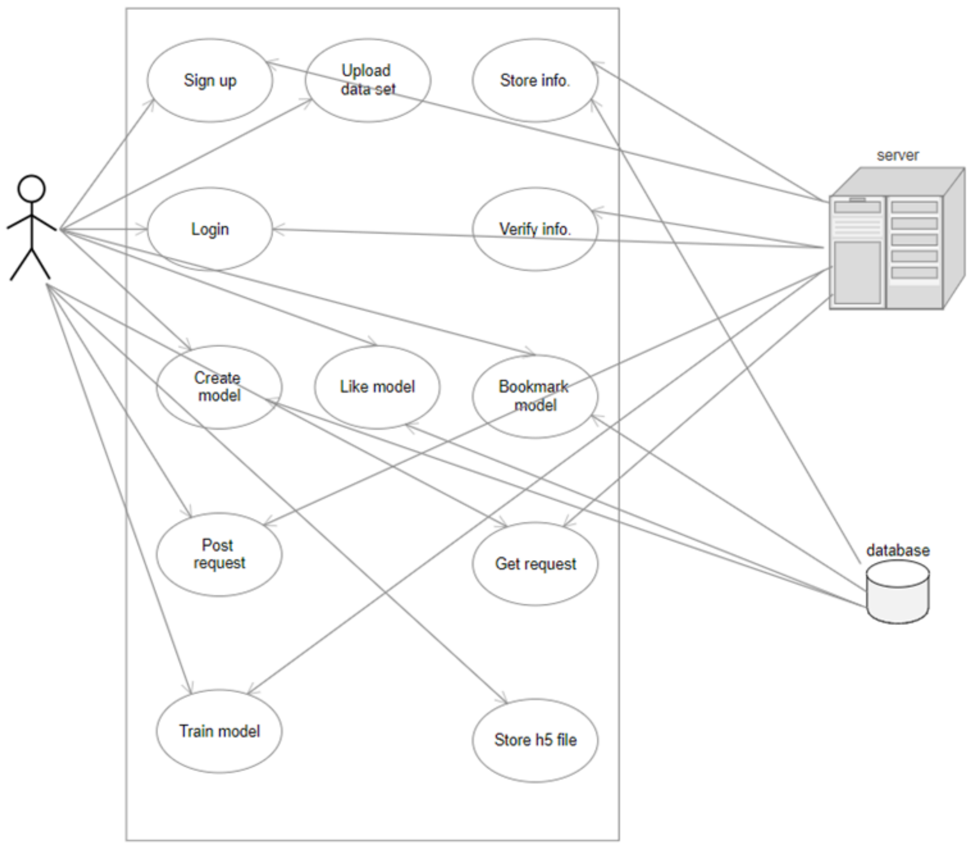


Figure 3.19: Use case diagram of the website

3.6.2 ER Diagram

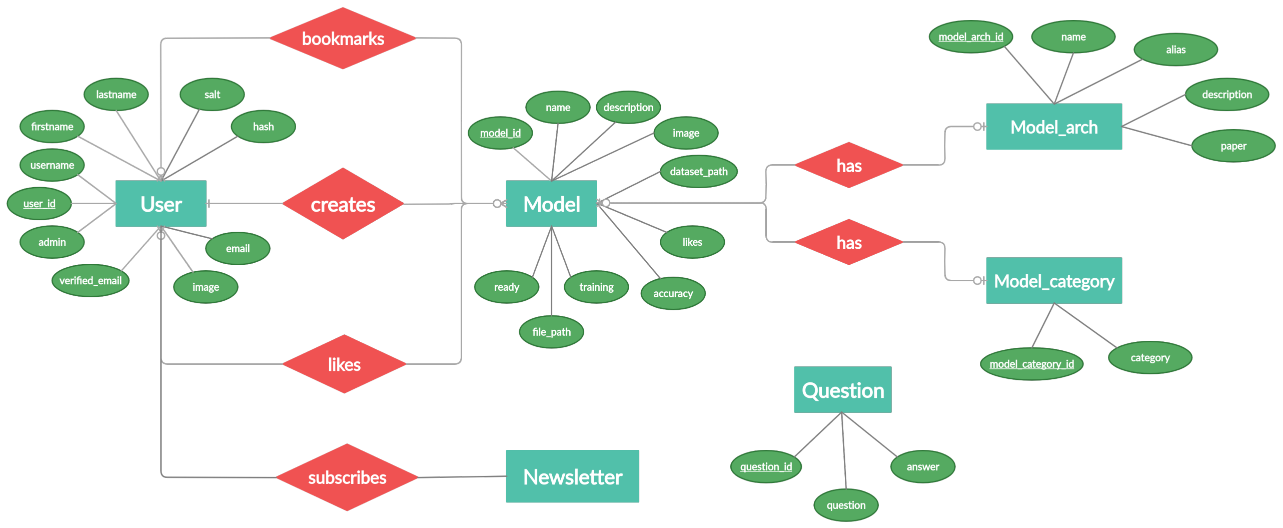


Figure 3.20: ER diagram of the database

The main system architecture shows all entities of the system (User, Model, newsletter, model architecture, and model category), each entity has some attributes that are stored for it, for example, User has attributes like (First name, last name, username, email, image, ...) all these information is stored for each user, the same thing for the model, where each model has attributes like (model_id, name, ...), the main idea is that each registered user can (create, like, bookmark) a model or more, where each model has "model_architecture" which are the scripts of our four models (MARNN, RAVEN,. . .) that surly have different implementations, "model_category" is the category of the model, it can be about car reviews, movie reviews,. . . .etc. user can also subscribe to a newsletter and make a new account if he wants, there is also an entity question that represents the most common questions about models and the site and their answers.

3.6.3 Database design

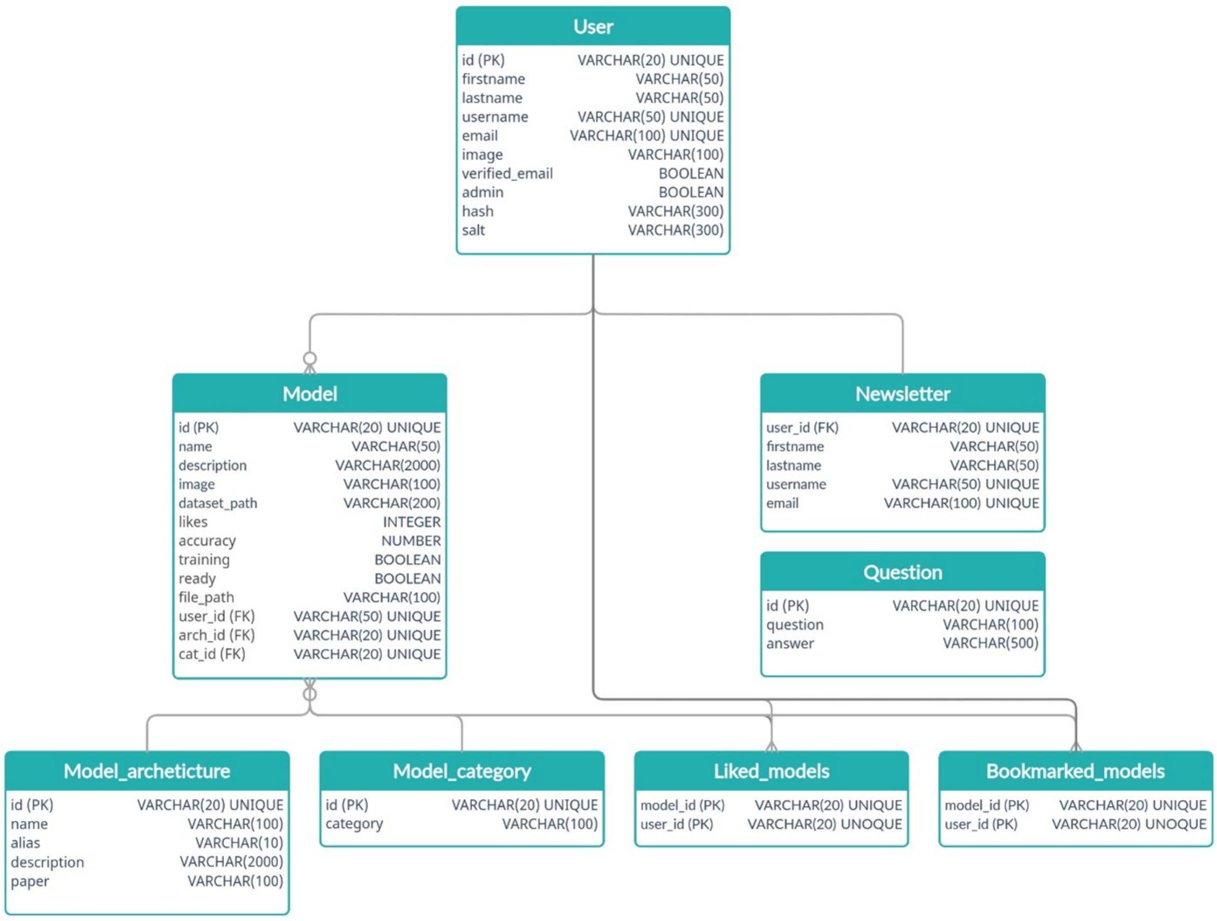

User
id (PK) VARCHAR(20) UNIQUE
firstname VARCHAR(50)
lastname VARCHAR(50)
username VARCHAR(50) UNIQUE
email VARCHAR(100) UNIQUE
image VARCHAR(100)
verified_email BOOLEAN
admin BOOLEAN
hash VARCHAR(300)
salt VARCHAR(300)
Model
id (PK) VARCHAR(20) UNIQUE
name VARCHAR(50)
description VARCHAR(2000)
image VARCHAR(100)
dataset_path VARCHAR(200)
likes INTEGER
accuracy NUMBER
training BOOLEAN
ready BOOLEAN
file_path VARCHAR(100)
user_id (FK) VARCHAR(50) UNIQUE
arch_id (FK) VARCHAR(20) UNIQUE
cat_id (FK) VARCHAR(20) UNIQUE
Newsletter
user_id (FK) VARCHAR(20) UNIQUE
firstname VARCHAR(50)
lastname VARCHAR(50)
username VARCHAR(50) UNIQUE
email VARCHAR(100) UNIQUE
Question
id (PK) VARCHAR(20) UNIQUE
question VARCHAR(100)
answer VARCHAR(500)
Model_archeticture
id (PK) VARCHAR(20) UNIQUE
name VARCHAR(100)
alias VARCHAR(10)
description VARCHAR(2000)
paper VARCHAR(100)
Model_category
id (PK) VARCHAR(20) UNIQUE
category VARCHAR(100)
Liked_models
model_id (PK) VARCHAR(20) UNIQUE
user_id (PK) VARCHAR(20) UNOQUE
Bookmarked_models
model_id (PK) VARCHAR(20) UNIQUE
user_id (PK) VARCHAR(20) UNOQUE

Figure 3.21: Database design

This part shows the database design of the system, and information about each entity in the system, for example, a user entity has a primary key that is unique for each user, it takes some information about the user that is represented by (first name, last name, username, email, ...). User has relation with created models, liked_models and bookmarked models as shown, also model has relation with model_architecture and model_category that are defined previously, all these entities have attributes like id, name as shown.

3.6.3.1 Implementation

In this application, we use node.js for building the main structure of our website (registration and login forms), and basic functions like (create, like, bookmark,....) models. We use PostgreSQL for database design (later we will talk about the detailed design of the database). Flask is used to get and post requests from/to server during calling the whole script of models [, MULT, MMMUBA, RAVEN] according to user demand and performing data preprocessing, training models, and saving h5 file for each trained model. Connecting between flask and node is an important point in the application, when the user create a model (that takes a specific ID) and start to train it using a generic data set (sending a request to the server to start train), model status should be stored as "training" in the database for that user until training process finishes, then model status changes to be ready to use (using node.js).

3.6.3.2 Discussion

We will discuss bases on what we make our choices.

Why Node, not any other language!

In general, we adopt node.js as it is more familiar to deal with Javascript front-end designs and React. Node.js offers the luxury of writing server-side applications in JavaScript. This allows us to write both the front-end as well as the back-end web application in JavaScript using a run-time environment. And we don't need to use any other server-side programming language. It also makes the deployment of web applications simpler because almost all web browsers support JavaScript.

What are the main differences between MongoDB and PostgreSQL?

PostgreSQL is a traditional RDBMS (relational database management system) SQL database, like Oracle and MySQL. PostgreSQL is free, so we prefer to work with structured data formed in tables as it's more scalable and maintainable, and retrieving information is easier in that case. MongoDB is a no-schema, NoSQL, JSON database. MongoDB has a free version, but they also have hosted and enterprise-paid versions. Even the free version includes free cloud monitoring hosted on their site for your local installation.

Flask vs Django

Flask provides support for API while Django doesn't have any support for API, so we choose flask to make a linkage between front-end (written in react) and back-end while dealing with the server in form of (Post, Get, ...) requests during training of models, Flask is a Python web framework built for rapid development whereas Django is built for easy and simple projects. Flask offers a diversified working style while Django offers a Monolithic working style.

AWS Vs Azure Vs Google Cloud

In general, we prefer to deal with google cloud as it has free services that adopt our needs (uploading user data sets into containers, training models in a suitable time, deploy our project to production to be available for all users), we also take benefit of 300$ free trail.

Here is a simple comparison between other alternatives:

Amazon Web Services:

Amazon Web Services is a subsidiary of amazon.com, which provides an on-demand Cloud Computing platform to individuals, companies, and governments on a paid-subscription basis. Amazon Web Services is the oldest and the most experienced player in the cloud market. As one of the oldest cloud providers, it has established a bigger user base, as well as bigger trust and reliability factors. AWS was publicly launched in 2006 with service offerings such as Elastic Compute Cloud (EC2), Simple Storage Service (Amazon S3), etc. By 2009, Elastic Block Store (EBS) was made public, and services such as Amazon CloudFront, Content delivery network (CDN), and more formally joined the AWS Cloud Computing Service offerings.

Microsoft Azure:

Microsoft Azure, initially called Azure, was launched in 2010 with the intent to provide a competent Cloud Computing platform for businesses. Azure was renamed as 'Microsoft Azure' in 2014, though the name 'Azure' is still commonly used. Since its inception, Microsoft Azure has shown great progress among its competitors.

Google Cloud Platform:

Google Cloud Platform (GCP), which is offered by Google, is a suite of Cloud Computing services that runs on the same infrastructure that Google uses internally for its end-user products such as Google Search engine, YouTube, and more. Google Cloud Platform began its journey in 2011, and in less than a decade it has managed to create a good presence in the cloud industry. The initial intent of Google Cloud was to strengthen Google's own products such as Google Search engine and YouTube. But now, they have also introduced their enterprise services so that anyone can use Google Cloud Platform which shares the same infrastructure as that of Google Search or YouTube.

3.6.3.3 Conclusion

We manage to make a complete website that allows the user to sign up to our site and log in to it, then [create, like and bookmark] model/s [MARNN, MULT, MMMUBA, RAVEN], this website is deployed to google cloud at this link https://sentimeter.dev/

Chapter 4

Results and discussion

4.1 Training results

In this section we will present the results of all models on MOSI data set. We managed to implement all models with results very close to the results stated at original paper of each model.

Table 4.1: Accuracy, Mean absolute error, and correlation for each of the trained baseline models.

Models	Accuracy	F1	Corr
RAVEN	79.0 %	0.667	0.468
MA-RNN	72.27%	0.732	0.449
MMMU-BA	79.55%	0.749	0.583
MMUU-SA	78.18%	0.771	0.562
MU-SA	79.55%	0.775	0.587
MulT	79.31%	0.707	0.530

All model trained on same data set MOSI and it took different number of epochs for different models, figure 4.1 shows the validation loss during the training:

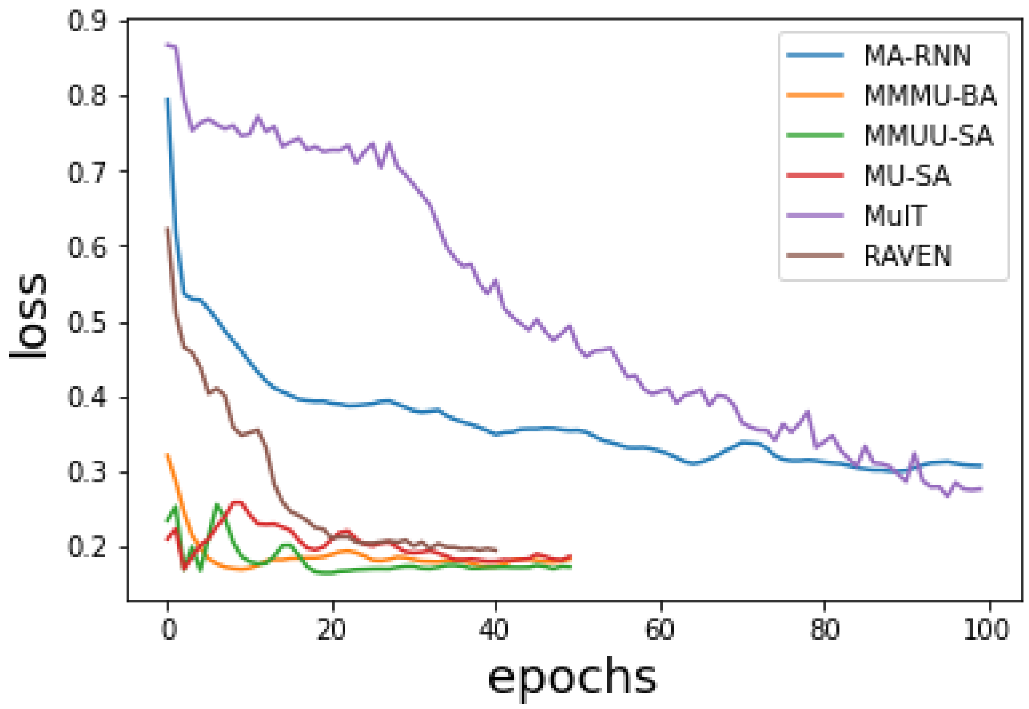


Figure 4.1: loss vs epochs of all models

4.2 Experiments

4.2.1 RAVEN

We faced a lot of challenges till we reach 79% accuracy:

1. For the visual features extraction, after splitting all the utterances' videos into multiple videos each video represents a word in the utterance. We started extracting the features for each frame in our new videos. But we found out that this process was time and memory-consuming. So, we thought that there's no way that the facial features could change drastically for a period that a word was said. So, for a given video, we check the number of frames, if it's less than 5 we only get the features for the middle frame. If the number of frames is more than 5, we generate 5 random numbers less than the length of video frames as indices and extract the features for the indexed frames with the same number as these indices.

2. For the acoustic features extraction, to split the utterances into multiple videos, each video contains one word, we needed the timestamps for each word that was said in the utterance. At first, we tried to use the sphinx speech recognition package which didn't yield good accuracy at all, then after looking for other alternatives, we finally landed on using the IBM Watson™ Speech to Text service and its accuracy was good and it returned each word with its sharply-timed timestamps.

3. Using LSTM for encoding the shifted word representation for each word in the utterance, at first, we reached 76% testing accuracy but as the accuracy didn't match the same as the paper, so we thought that there is a problem in the feature extraction process as we used other techniques (for audio features: using Libros, facial features: pre-trained deep learning face detector model). We didn't use the proposed method in

the paper to extract the features as COVAREP for extracting audio features as this tool was based on MATLAB, so we preferred to design the code in a consistent way all in python. Therefore, we decided to improve the accuracy by using the transformer block instead of the LSTM other than changing the feature extraction methodology. In much research, it was proved that transformer gets better results than LSTM in many NLP tasks. By using it we reached 79% testing accuracy.

4. Every time we faced an issue in the model, we started debugging by removing some layers and check if the model over-fits then it is working, so we check what is the input for the removed layer and determine if the input has a problem or the features extraction method has a problem.

5. We made fine-tuning manually for all the hyper-parameters until we reach a stable accuracy like the batch size and number of epochs. Every time we checked if the accuracy gets better or not. Drop-out layers' parameters affect the accuracy, so we change it multiple times.

6. Splitting the audio and video with ffmpeg_extract_subclip function using moviepy library didn't split them correctly and took a lot of time for the process. Therefore, we tried using the OpenCV library for splitting the video and it was very fast compared with moviepy , also splitting the audio file using pydub package was very fast compared with moviepy.

4.2.2 MulT

- In the beginning, we tried to use the multi-head attention layer that is embedded in tensorflow.keras.layers but it didn't get good accuracy. So, we implemented a custom multi-head attention layer and followed the instructions of the paper in building this layer, when we implemented this layer, the accuracy got better but didn't reach the paper's accuracy.
- We also found out that the features of the three modalities have a lot of zeros in them because of the padding so we added a masking layer and got a better accuracy.
- We first used the same hyper-parameters as the paper like the learning rate (1e-3) and a dropout rate in the Cross-modal Attention Block layer (0.2) but when we change these parameters to learning rate (1e-4) and the dropout to (0.1) the accuracy got higher and reached 79.31%.
- We didn't reach the paper's accuracy because of the feature extraction we used on the data we use a face detecting model to get the face landmarks and we used the Mel-frequency spectrum to get the audio features and glove for the text, but in the paper, they used "Facet" for the visual features and "COVAREP" for audio features. When we used the preprocessed data from the paper our architecture got the same accuracy but when we used our feature extraction we got less accuracy but we can't use "Facet" and "COVAREP" in our project because they extract the features manually so we couldn't use them in our website.

4.2.3 MMMU-BA, MMUU-SA, MU-SA & MARNN

4.2.3.1 Feature extraction

We faced a lot of problems in features extraction (especially in text feature extraction) which leads to a huge downgrade in the accuracy of MARNN to the paper. This confused us so we talked to the author of the "Multi-Attention Multimodal Sentiment Analysis" paper (prof.Taeyong Kim) and asked him about this downgrade in the accuracy after extracting text features the same way the paper mentioned. And his answer was that their main concern was the model itself not the way of feature extraction, because they wanted to reach the state of the art, they used the same data processed of other papers. when we used the processed data he mentioned, we found a huge increase in the accuracy, but we can't use any already processed data in our project.

4.2.3.2 MMMU-BA, MMUU-SA, MU-SA

We faced fewer problems in the implementation of MMMU-BA, MMUU-SA, MU-SA and this is due to the great details given in this paper. these problems were in the overall accuracy of the model, but after normalizing the input data after extracting it from the pickle file these problems were solved.

4.2.3.3 MA-RNN

We faced some problems in MARNN such as in multi-head attention we have implemented it in a different way which gives us a decrease by 1% over single attention (a small change in the implementation of the attention mechanism leads to a huge change in the accuracy). we also faced the same problem of normalization (which we discussed previously in MMMU-BA, MMUU-SA, MU-SA) so after normalizing each model we observed a little change in the accuracy by 2%

4.3 Efficiency

We evaluated the efficiency of all the models using PC with AMD Ryzen 9 3900X 3.8 GHz 12-Core AM4 processor, 32GB RAM, and RTX 2080Ti GPU.

Table 4.2: Time efficiency comparison between different models

Models	Training time	Feature extraction time	Additional data processing time	Total time
RAVEN	25 min	1 h, and 30 min	2 h	4 h
MA-RNN	30 min	1 h, and 30 min	-	2 h
MMMU-BA	20 min		-	1 h, and 50 min
MMUU-SA			-	
MU-SA			-	
MulT	30 min	40 min	-	1 h, and 30 min

The additional data processing time in RAVEN is due to that RAVEN works on word-level. In other words, we need to split each video utterance into small utterances in which contains a person spell only one word. this process is a very time-consuming process which makes RAVEN not very efficient in terms of time needed for training.

Table 4.3: Number of parameters comaprison between different models

Models	Number of parameters	
	Total	Trainable
RAVEN	5,043,753	1,081,953
MA-RNN	595,687	595,311
MMMU-BA	9,524,102	9,524,102
MMUU-SA	9,523,502	9,523,502
MU-SA		
MulT	725,161	725,161

Chapter 5

Conclusion

Human language and emotions are very complicated and can't be described by just the meaning of the word the person's say it involves the voice tone (loud, quiet, sad, happy) and the facial expressions if the user is (smiling, laughing, sad) all these features can't be known just from the meaning of the words we should keep these features in mind to get the sentiment of the talking person and know how he feels so we need to create a multi-modal sentiment analysis model that involves all three modalities textual, visual, and acoustic in the same time to get the right guess of the sentiment that we need, we built a four multi modal models so we can get the best performance and output from the multi modal sentiment analysis algorithm and have some variety in the domains that we can use these models in. The main purpose of our website is to provide the user an easy way to create a multi modal sentiment analysis model without needing to write his own code or hire a software team to create him a model for his data so he can use it afterwards all the user need to do is to create an account on our website and create his model by uploading his data set and choosing the suitable architecture for his data from four architectures we provide on or website (RAVEN, MULT, MARNN, MMMUBA) then the user need to wait until the model finishes training on the data set he uploaded after the training finishes the user can download the model file so he can use it in his application afterwards and we provide the user an API that he can use on our website after the training finishes that API allow the user to get the sentiment of a video that he uploaded to the website so he can test his model and know if it works will or not.

References

[1] O. Boufeloussen, “Simple explanation of recurrent neural network (rnn),” Sep 2020. [Online]. Available: https://medium.com/swlh/simple-explanation-of-recurrent-neural-network-rnn-1285749cc363

[2] M. Farzaneh and R. M. Toroghi, “GGA-MG: generative genetic algorithm for music generation,” *CoRR*, vol. abs/2004.04687, 2020. [Online]. Available: https://arxiv.org/abs/2004.04687

[3] Jeblad, “File:gated recurrent unit, type 1.svg,” Feb 2018. [Online]. Available: https://commons.wikimedia.org/wiki/File:Gated_Recurrent_Unit,_type_1.svg

[4] Y. Wang, Y. Shen, Z. Liu, P. P. Liang, A. Zadeh, and L.-P. Morency, “Words can shift: Dynamically adjusting word representations using nonverbal behaviors,” 2018.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017.

[6] Y.-H. H. Tsai, S. Bai, P. P. Liang, J. Z. Kolter, L.-P. Morency, and R. Salakhutdinov, “Multimodal transformer for unaligned multimodal language sequences,” 2019.

[7] D. Ghosal, M. S. Akhtar, D. Chauhan, S. Poria, A. Ekbal, and P. Bhattacharyya, “Contextual inter-modal attention for multi-modal sentiment analysis,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3454–3466. [Online]. Available: https://aclanthology.org/D18-1382

[8] T. Kim and B. Lee, “Multi-attention multimodal sentiment analysis,” 06 2020, pp. 436–441.

[9] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” 2015.

[10] “A twitter sentiment analysis tool.” [Online]. Available: http://help.sentiment140.com/

[11] F. Eyben, M. Wöllmer, and B. Schuller, “Opensmile: The munich versatile and fast open-source audio feature extractor,” in *Proceedings of the 18th ACM International Conference on Multimedia*, ser. MM ’10. New York, NY, USA: Association for Computing Machinery, 2010, p. 1459â1462. [Online]. Available: https://doi.org/10.1145/1873951.1874246

[12] Affectiva, “Affectiva media analytics.” [Online]. Available: https://go.affectiva.com/affdex-for-market-research

[13] Juliako, “What is azure video analyzer for media (formerly video indexer)? - azure video analyzer for media.” [Online]. Available: https://docs.microsoft.com/en-gb/azure/azure-video-analyzer/video-analyzer-for-media-docs/video-indexer-overview

[14] R. Inc., “Video content analytics platform: Repustate video ai.” [Online]. Available: https://www.repustate.com/video-analysis/

[15] “Sentiment analysis,” May 2019. [Online]. Available: https://gengo.com/sentiment-analysis/

[16] A. Zadeh, R. Zellers, E. Pincus, and L.-P. Morency, “Mosi: Multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos,” 2016.

[17] J. Robert, M. Webbie *et al.*, “Pydub,” 2018. [Online]. Available: http://pydub.com/

[18] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, “librosa: Audio and music signal analysis in python,” in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[19] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[20] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” 2014.

[21] S. Poria, E. Cambria, D. Hazarika, and P. Vij, “A deeper look into sarcastic tweets using deep convolutional neural networks,” 2017.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-scale video classification with convolutional neural networks,” in *CVPR*, 2014.

Appendix A

Website API End-points

The following tables show all methods performed by client and server and their descriptions:

Table A.1: Users Table

METHOD	ROUTE	AUTHENTICATION	Description
GET	"/users/users/ <username>"	NONE	Get user data by username
POST	"/users/login"	NONE	Login user using user credentials
POST	"/users/signup"	NONE	Register user using user data provided
POST	"/users/verify_email"	NONE	Verify user email using user
POST	"/users/ resend_verification_e_mail"	NONE	Resend verification email to user email
GET	"/users"	ADMIN	Get all users' data from database
POST	"/users/password /change"	NONE	Change user's password using id
POST	"/users/password /forget"	NONE	Send password reset mail using user
POST	"/users/password /reset"	NONE	Reset user's password
GET	"/users/me"	USER	Get current user's data
PUT	"/users/me"	USER	Update current user's data
DELETE	"/users/me"	USER	Delete current user from database
POST	"/users/image/change"	USER	Delete user's image

Table A.2: Newsletter Table

METHOD	ROUTE	AUTHENTICATION	Description
GET	"/dashboard/ newslet- ter/isSubscribed"	USER	Get whether current user is subscribed or
GET	"/dashboard/ newsletter/subscribe"	USER	Subscribe current user to newsletter
GET	"/dashboard/ newslet- ter/unsubscribe"	USER	Unsubscribe current user from newsletter
GET	"/dashboard/ newsletter"	USER	Get all newsletter's subscribers
POST	"/dashboard/ newsletter/send"	ADMIN	Send mail to current subscribers

Table A.3: Questions Table

METHOD	ROUTE	AUTHENTICATION	Description
GET	"/dashboard/faq"	USER	Get all questions answers
POST	"/dashboard/faq"	ADMIN	Add a new question
PUT	"/dashboard/faq/"	ADMIN	Update question using question id
DELETE	"/dashboard/faq/"	ADMIN	Delete question using question id

Table A.4: Executable file

METHOD	ROUTE	AUTHENTICATION	Description
GET	"/exec_files/windows"	None	Get link to data set Creator for Windows
GET	"/exec_files/linux"	None	Get link to data set Creator for Linux

Table A.5: Routes with Flask API

METHOD	ROUTE	AUTHENTICATION	Description
POST	"/flask/start_training / <model _id>"	None	Set 'training: TRUE' for given model using id
POST	"/flask/finish_training /<mod el_id>"	None	Set 'ready: TRUE' for given model using id

Table A.6: Models Architecture table

METHOD	ROUTE	AUTHENTICATION	Description
GET	“/models/arch”	USER	Get all model architectures
GET	“/models/arch/”	USER	Get one model architecture using id
POST	“/models/arch”	USER	Create a new model architecture
PUT	“/models/arch/”	ADMIN	Update a model architecture using id
DELETE	“/models/arch/”	ADMIN	Delete a model architecture using id

Table A.7: Models Categories Table

METHOD	ROUTE	AUTHENTICATION	Description
GET	“/models/category”	USER	Get all model categories
GET	“/models/category/”	USER	Get one model category using id
POST	“/models/category”	USER	Create a new model category
PUT	“/models/category/”	ADMIN	Update a model category using id
DELETE	“/models/category/”	ADMIN	Delete a model category using id

Table A.8: Models Table

METHOD	ROUTE	AUTHENTICATION	Description
GET	"/models"	USER	Get all models
GET	"/models/me"	USER	Get all models created by current user
GET	"/models/me/"	USER	Create a model created by current user
GET	"/models/isOwner/"	USER	GET whether a model is created by current user or not
GET	"/models/"	None	GET a model using id
POST	"/models/create"	USER	Create a model by current user
PUT	"/models/"	USER	Update a model by its owner using id
POST	"/models/delete/"	USER	Delete a model by its owner using id
GET	"/models/likes"	USER	GET current user's liked models
GET	"/models/likesId"	USER	GET current user's liked model's id
GET	"/models/likes/:id"	USER	GET model's like
GET	"/models/likesId"	USER	GET current user's liked model's id
POST	"/models/like"	USER	Current user likes a model
POST	"/models/unlike"	USER	Current user unlikes a model
GET	"/models/bookmarks"	USER	GET current user's bookmarked models
GET	"/models/bookmarksId"	USER	Get current user's bookmarked model's ids
POST	"/models/bookmark"	USER	Current user bookmarks a model
POST	"/models/unbookmark"	USER	Current user unbookmarks a model
POST	"/models/image/"	USER	Upload model's image by it's owner
DELETE	"/models/image/"	USER	Delete model's image by its owner
POST	"/models/data set/"	USER	Upload model's data set by its owner