Multi-modal Sentiment Analysis

Sim Theen Cheng THEENCHENG_SIM@MYMAIL.SUTD.EDU.SG

Aditya Vishwanath ADITYA_VISHWANATH@MYMAIL.SUTD.EDU.SG

Phang Heng Yan Gabriel GABRIEL_PHANG@MYMAIL.SUTD.EDU.SG

Editor:

Abstract

Combining information from multiple modalities can be beneficial for improving the performance of deep learning models for sentiment analysis. Our paper aims to prove that, through utilising audio, textual and visual modalities. We will first go into detail about how we sourced and gathered different datasets. We then lead into how we trained and evaluated each modality individually. We then proceed to talk about combining the models and evaluating our approach.

Keywords: Multi-modality, Sentiment Analysis

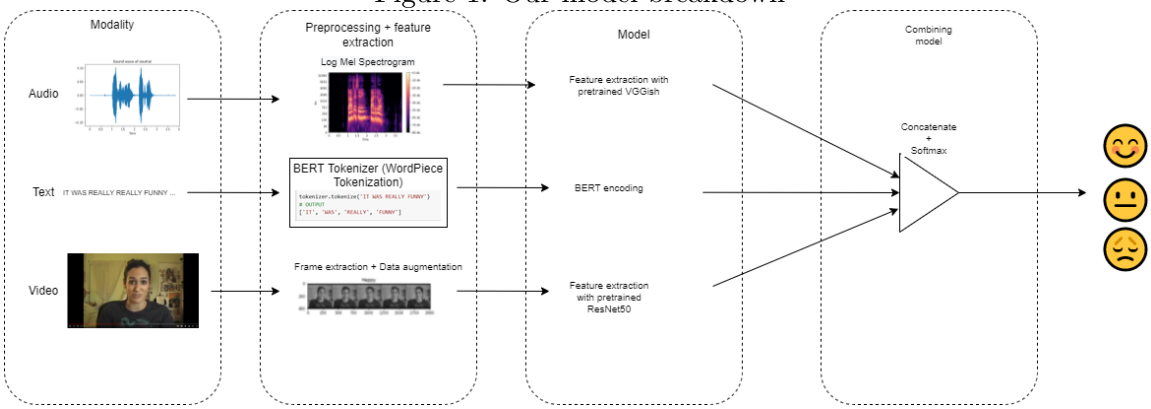
1. Introduction

Analysing text from videos is insufficient and often unrepresentative / inconclusive of a subject's true emotion. However, behavioral cues present in both vocal and visual modalities can also complement text-based sentimental analysis and can lead to a more accurate representation of emotion. The vocal modulations and facial expressions in the visual and audio data, along with textual data, provide important cues to better identify affective states of the opinion holder. Therefore, we decided to adopt the Multimodal Sentiment Analysis.

Multimodal Sentiment Analysis is an extension of sentiment analysis, which is traditionally done through analysis of text. Sentiment analysis is also done through the analysis of images, videos and audios as well. These analyses have a variety of uses from gaining insights on a company or brand's reviews to developing a rating of a movie based on comments on it. Performing sentiment analysis through only text or only video or audio is what we would call unimodal sentiment analysis since we only extract and analyse one feature to gain our results. Likewise, using any 2 of these 3 features is what we would call bimodal sentiment analysis. Multimodal sentiment analysis is a further extension to this where we use all 3 features; text, image/video and audio to perform our study and gain insights through the results we find. Similar to the traditional sentiment analysis, one of the most basic task in multimodal sentiment analysis is sentiment classification, which classifies different sentiments into categories such as positive, negative, or neutral. This involves extracting each feature from its respective dataset (if different datasets are used for each) and performing sentiment analysis using models like Logistic Regression, for example, and then fuse the

analyses of all three to form one model which shows the combined findings and even enable us to compare which feature is a better indicator of sentiments.

Figure 1: Our model breakdown



2. Dataset and Collection

When we first took up this project and started working on it, our idea was to source 3 completely different datasets for each feature and then proceed to fuse them together to form our final model. As we had presented in our initial presentation, we intended to use The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) **?** for audio analysis as well as the IMDB dataset Maas et al. (2011) for text analysis. We had not decided on a dataset yet for the video analysis at that point and that was part of the KPIs we had in place going forward. However, we later felt that as much as we would love the challenge to fuse these different datasets and models, the constraint of time and work from other modules meant that it would possibly be too demanding for us.

With this in mind, we went forward with using the CMU-MOSI dataset Zadeh et al. (2016) for the audio and video analysis. CMU-MOSI is a standard benchmark for multimodal sentiment analysis. It contains 2199 opinion utterances with sentiment annotated between very negative to very positive in seven Likert steps. Using this dataset not only makes multimodal fusion easier but also enables us to compare which feature has a greater impact on our analysis and is a better measure of sentiments.

3. Video

3.1 Data preprocessing

We captured 5 random frames from each segmented video in order to get a representative portion of the subject's attitudes in the clip. Using the application of data augmentation, we obtained 10,995 examples of 224 by 224 images from 2199 video clips. The figure below shows an example of the frames used in a video clip that was eventually deemed as happy.

Figure 2: Frames used in video clip

While using 5 random frames might not be the most optimum number or feature all the moods presented throughout a clip, we decided to proceed with 5 as the training and test results showed fairly accurate results. However, in future, we could look at looking at different segments of the video and taking frames from different segments of a specific clip to boost this accuracy even higher.

3.2 Model

In training the models using the videos, we tried to train our own convolutional neural network from scratch, but the model took too long to train. Therefore, we utilised a pretrained ResNet50 He et al. (2015) model obtained from Keras to assist us with the weighting system. By fixing the pretrained weights using the ResNet50 library, it was actualised that the time spent training was greatly reduced since gradients do not need to be re-computed. With training time being our initial limitation in running our models, this library helped in reducing the time taken for training and allowed us to focus on improving accuracy of our model. We ran the training for 30 epochs.

4. Text

4.1 Data preprocessing

We used the IMDB dataset prepared by L.Maas, the dataset contains 25,000 movie reviews for training and testing respectively which are labelled, for supervised learning as well as 50,000 unlabelled data for unsupervised learning which we will not be using for this project. This means that the only pre-processing we really needed to do wastokenising as well as removing unlabelled reviews.

1. Tokenizing

To feed our data inot the BERT model, we need to process it and (1) create InputExample objects out of our train and test datasets and then (2) tokenize the InputExample objects, create the required input format with the tokenized objects, and finally, create an input dataset that we can feed to the model.

We did these with 2 functions:

- convert-data-to-examples
- convert-examples-to-tf-dataset

2. Removing Unlabelled Reviews

As for removing unlabelled reviews from the IMDB dataset, we perform the following operations, splitting the dataset into directories and subdirectories and then removing the subdirectory with data used for unsupervised learning which are the unlabelled reviews. The last line of code then print out the train folder after deleting the unsupervised subdirectory just for us to confirm that the deletion was successful.

Figure 3: Code to remove unlabelled reviews

```
import os
import shutil
# Create main directory path ("/aclImdb")
main_dir = os.path.join(os.path.dirname(dataset), 'aclImdb')
# Create sub directory path ("/aclImdb/train")
train_dir = os.path.join(main_dir, 'train')
# Remove unsup folder since this is a supervised learning task
remove_dir = os.path.join(train_dir, 'unsup')
shutil.rmtree(remove_dir)
# View the final train folder
print(os.listdir(train_dir))
```

4.2 BERT

BERT, or Bidirectional Encoder Representations from Transformers is a machine learning model often used for NLP tasks. BERT was only recently developed in 2018 and was initially trained on English Wikipedia (2,500 million words) and BooksCorpus (800 million words) which made it achieve very good accuracies in most NLP tasks. Devlin et al. (2018)

Figure 4: BERT Model Summary

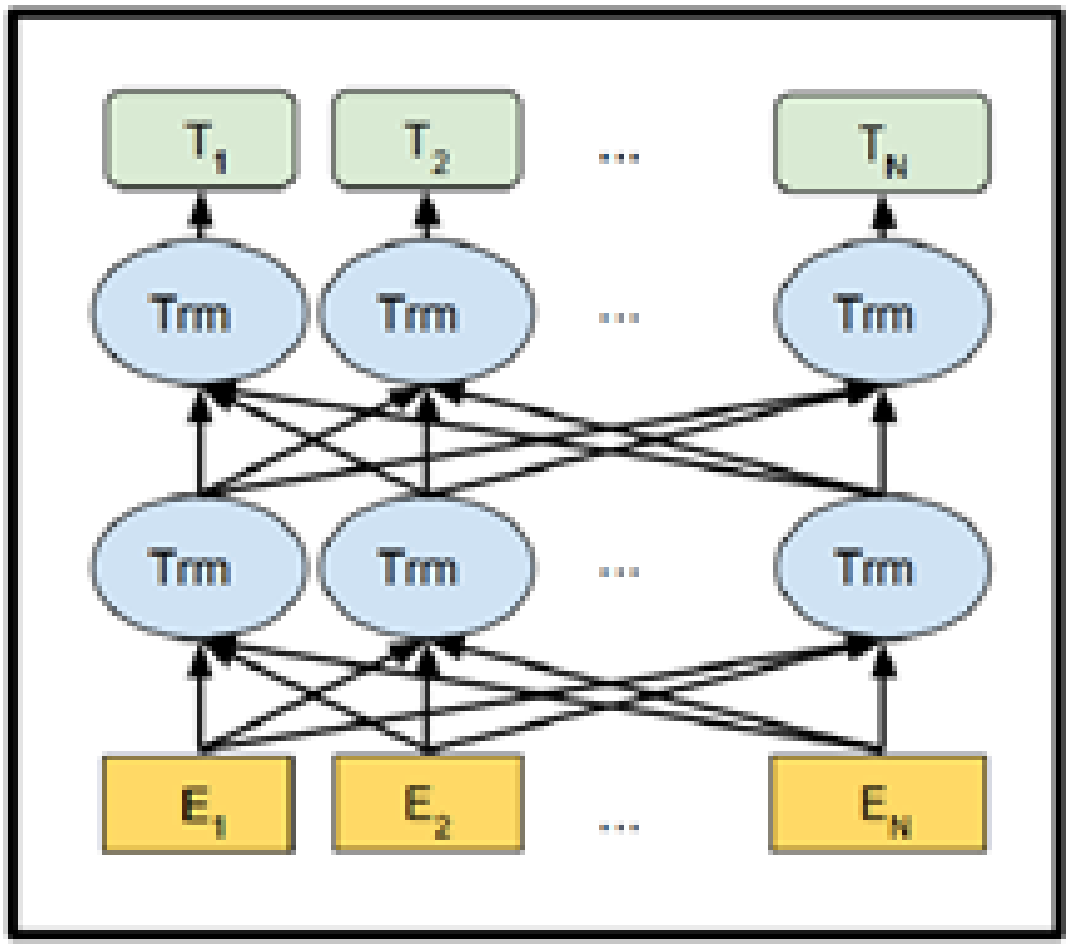
```
model.summary()

Model: "tf_bert_for_sequence_classification"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bert (TFBertMainLayer)       multiple                  109482240

dropout_37 (Dropout)         multiple                  0

classifier (Dense)           multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
```

For the text analysis aspect of our project, we used Tensorflow with the Hugging Face Transformer Library. Transformers provides a lot of pre-trained models that perform various forms of text analytics like classification, question answering, translation along with thousands more!

4.3 Model

We initially wanted to build a model from scratch as well or make an extension of the Logistic Regression Model we implemented in Week 6 Lab which was also trained on the IMDB dataset. However, we felt that using a classifier would make more sense for text analysis since it would also be easier in combining the different modalities later. We decided to explore something new and therefore implemented the BERT model which as we mentioned earlier, is really good at performing NLP tasks like these. A word starts with its embedding representation from the embedding layer. Every layer does some multi-headed attention computation on the word representation of the previous layer to create a new intermediate representation. All these intermediate representations are of the same size. The diagram below shows the BERT model architecture where, for example, E_n is the embedded representation, T_n is final output and T_{rm} are the intermediate representations. The BERT model we implemented achieves 95% accuracy on the sentiment analysis of the IMDB review set.

Figure 5: BERT model architecture



5. Audio

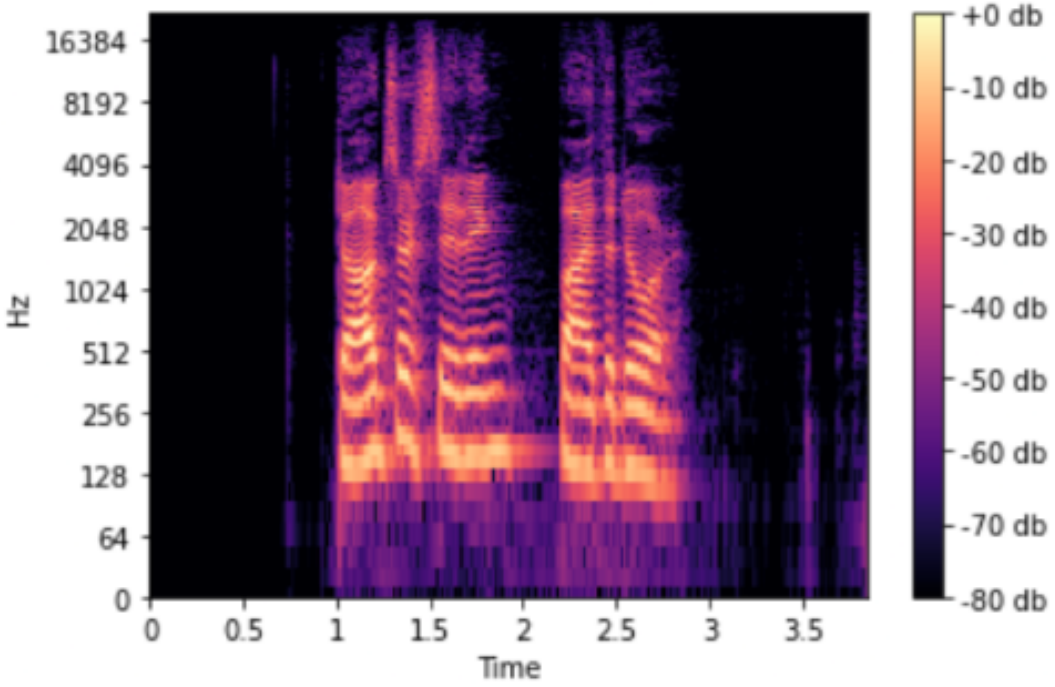
5.1 Data preprocessing

In order to train our model, we had to first pre-process our raw audio WAV files. We used the Python library Librosa to handle audio pre-processing. At a high level, we first load in the WAV file as a floating point time series, resampled at a fixed sampling rate of 44100Hz. We then tried two approaches for feature extraction: Mel Frequency Cepstral Coefficients (MFCCs), which can then be used for modelling as tabular data, and the log mel spectrogram, which allows us to visualize audio as an image. These images can then be used for deep learning through convolutional networks.

We ended up using Mel spectrograms for our features. A mel spectrogram is a spectrogram where the frequencies are converted to the mel scale, which is a unit of pitch such

that equal distances in pitch sounded equally distant to the listener, proposed by Stevens, Volkmann, and Newmann in 1937.

Figure 6: Mel spectrogram



In summary, this is how we pre-process the data:

1. We map the audio signal from the time domain to the frequency domain using the fast Fourier transform, and perform this on overlapping windowed segments of the audio signal.

2. We converted the frequency to a logarithmic scale and the amplitude to decibels to form the spectrogram.

3. We finally map the frequency onto the mel scale to form the mel spectrogram.

5.2 Model

We utilized VGGish Hershey et al. (2017), a pre-trained convolutional neural network (CNN), from Google for feature extraction. As the name implies, the architecture was adapted from the VGG models used for image classification. The model takes in a log mel spectrogram as audio input, and outputs a 128-wide embedding layer. We kept the pretrained weights for the model. To test the accuracy of the model, we had a final fully connected layer activated by softmax. We ran this for 30 epochs.

Figure 7: VGGish architecture

```
import vggish_slim
import vggish_params
import vggish_input
import tensorflow.compat.v1 as tf
import vggish_postprocess
def CreateVGGishNetwork(hop_size=0.96):   # Hop size is in seconds.
  ###
  # Define VGGish model, load the checkpoint, and return a dictionary that points
  # to the different tensors defined by the model.
  ###
  vggish_slim.define_vggish_slim()
  checkpoint_path = 'vggish_model.ckpt'
  vggish_params.EXAMPLE_HOP_SECONDS = hop_size
  vggish_slim.load_vggish_slim_checkpoint(sess, checkpoint_path)
  features_tensor = sess.graph.get_tensor_by_name(
      vggish_params.INPUT_TENSOR_NAME)
  embedding_tensor = sess.graph.get_tensor_by_name(
      vggish_params.OUTPUT_TENSOR_NAME)
  layers = {'conv1': 'vggish/conv1/Relu',
            'pool1': 'vggish/pool1/MaxPool',
            'conv2': 'vggish/conv2/Relu',
            'pool2': 'vggish/pool2/MaxPool',
            'conv3': 'vggish/conv3/conv3_2/Relu',
            'pool3': 'vggish/pool3/MaxPool',
            'conv4': 'vggish/conv4/conv4_2/Relu',
            'pool4': 'vggish/pool4/MaxPool',
            'fc1': 'vggish/fc1/fc1_2/Relu',
#             'fc2': 'vggish/fc2/Relu',
            'embedding': 'vggish/embedding',
            'features': 'vggish/input_features',
          }
  g = tf.get_default_graph()
  for k in layers:
    layers[k] = g.get_tensor_by_name( layers[k] + ':0')
  return {'features': features_tensor,
          'embedding': embedding_tensor,
          'layers': layers,
          }
```

6. Combining Modalities

Our approach was to utilize concatenation to combine the embeddings from each of the models addictively. This is achieved with the Concatenate layer in Keras, followed by a softmax activation. Unfortunately, we were not able to finish concatenating the different modalities for ourselves in time given coupled with our inexperience in the actually implementing the process of concatenating modalities like we have spoken about in this paper so far. We ran into a lot of hurdles in this process and that therefore makes it had for us to have the outputs to show from our own work how well a multi modal sentiment analysis model works. However, through the process of working on this project, we as a team did

a lot of research on existing models like the one we aim to build as well as the different models we used for each feature, giving us the ability to still learn about them and also still provide a theoretical evaluation of the multi modal sentiment analysis model we built.

7. Evaluation

We expect that in theory, through combining information from multiple modalities, we should be able to achieve a more accurate prediction of sentiment compared to the individual modalities. However this is often not the case. It is challenging to leverage different modalities to improve model performance due to the intricate relations between modalities. This additive approach fails to consider how some modalities might be a better indicator of sentiment than others. An example could be how the choice of words could be sarcastic and contrasting to the true sentiment, and facial expressions and tone of voice could be more indicative of the true sentiment. Though theoretically the deep neural network model, given a large enough training set, has the capability to learn the weights and re-balance accordingly. However, in practice, it is very difficult to train and regularize such parameters, and our simplistic approach also further inhibits the model's ability to correct itself. The paper *Learn to Combine Modalities in Multimodal Deep Learning*(Liu et al. (2018)) covers more in detail about how a multiplicative combination layer seeks to overcome this limitation.

8. Conclusion

Looking back, we first took up this project because of how cool we found the idea of being able to read sentiments through even one feature, let alone 3 and not to mention, to be able to bring all 3 together to form one model. We knew it would be a challenge to bring this to fruition but we agreed as a team that it would be a great opportunity to learn so much about the field of Data Science a well as the various models and applications of everything we learnt in lessons as well as exploring beyond it like the concept of concatenating different modalities. Overall, we cannot deny that we could have done a lot of things better in this project but we still feel enriched by the experience to do what is most important, learn.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017. URL `https://arxiv.org/abs/1609.09430`.

Kuan Liu, Yanen Li, Ning Xu, and Prem Natarajan. Learn to combine modalities in multimodal deep learning, 2018.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Amir Zadeh, Rowan Zellers, Eli Pincus, and Louis-Philippe Morency. MOSI: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *CoRR*, abs/1606.06259, 2016. URL `http://arxiv.org/abs/1606.06259`.