Monika KIRIEJEVAITĖ (3035853195), Justinas ŠVYKAS (3035850466), Vlad GRIGORAŞ (3035852165)

BeepTweet

Final project report | COMP3362 - Hands-On AI: Experimentation & Applications

Objectives

The present work attempts to build a fun application that draws on our visual and auditory perceptual integration by generating music based on a given tweet (a short string of 1-2 sentences). The project explores natural language processing methods, sentiment analysis on tweets and object-oriented generation of music.

Highlights of the project

- Created a unique dataset with Twitter API and auto-labelled tweets based on tags and emojis
- Sentiment analysis on tweets: multi-classification for emotions and regression for VAD values
- Sounds generation from sentiment (categorical and dimensional)

Emotion (VAD) analysis from text

EDA

The first dataset that we decided to work with was Emobank (Buechal et. al., 2017). This data set has sentences annotated with Valence, Arousal and Dominance sentiment similar to those that we see on the right. Each sentence is labelled from 1 to 5. The data set has no null values nor the sentences are particularly bigger than any other. After doing some basic operations and drawing histograms and boxplots (Figure 1.), we discovered that VAD values follow a normal distribution. Also, the data doesn't seem skewed. As valence arousal and dominance is independent values we

```
Text with min dominance
D                                                    1.78
text    I shivered as I walked past the pale man's bla...
```

```
Text with max dominance
D        4.2
text    "NO"
```

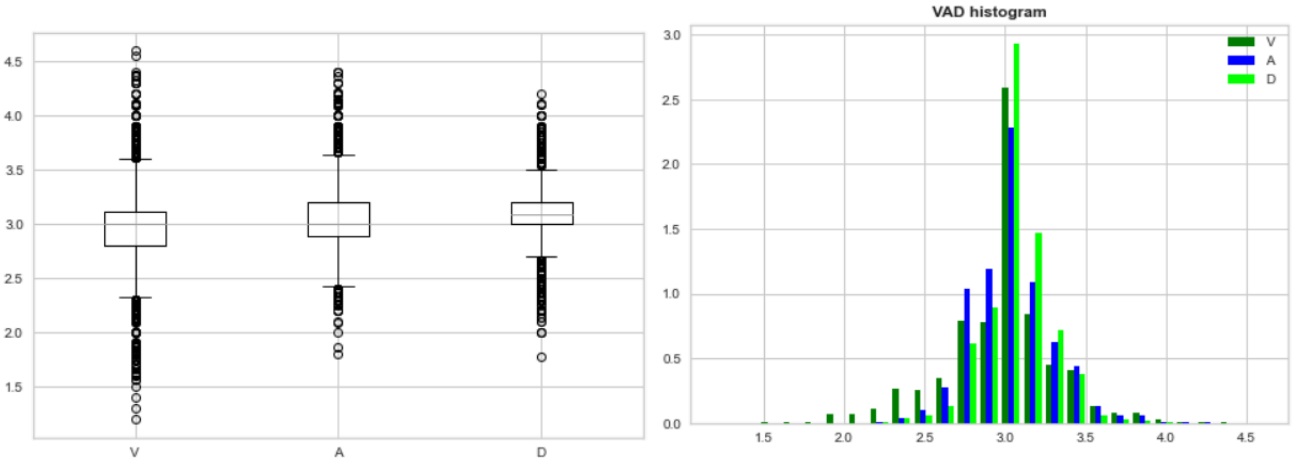


Figure 1. VAD values distribution in Emobank dataset

decided to create 3 independent models as well (Russell & Mehrabian, 1974). In order to gain better predictions, by hypertuning parameters for each model individually.

$$P(v, a, d|X) = P(v|X)P(a|X)P(d|X).$$

Feature extraction

So our raw data from which we will try to predict values is text. So we started looking into how to preprocess this data and what features to extract for the computer to comprehend the text easier. First, we cleaned text by decoding HTML, cleaning text from specific characters like @mentions or hashtags, discarded links, cleaned numbers, and upper case characters turned into lowered cases (except cases when upper case characters are subsequent). After cleaning we have tried out a few feature extraction methods for text. Mainly word embeddings and n-gram vectors (with Tf-idf encoder) methods out of which we chose the latter since it brought better prediction results.

	Word embeddings	N-gram vectors
MSE	0.12	0.03

Experiments with Models

We have tried out 4 models: Linear regression, Ridge regression, Neural Networks and Recurrent Neural Networks. From experiments (Table 1.), we found out that Linear regression dramatically overfits the data. While Ridge regression NN, RNN performs well enough. We chose to use Ridge regression since the performance of Ridge regression seems to be the best. Also, our model suffers from feature multicollinearity since we are using n-gram feature extraction. Ridge regression is known to deal with it very well. (for simplicity I am using here evaluations for valence only)

Evaluation for Valence	Linear Regression	Ridge Regression	NN	RNN
MSE train	0	0.03	0.1384	0.1284
MSE test	5902845	0.09		

Table 1. Results from the experiments on different models

Hypertuning parameters

For hypertuning parameters a good evaluation is critical. Since we have a relatively small dataset we decided to use k-fold cross-validation. For a function, we settled down on a basic GridSearch. After some tuning we found that the best parameters for the Valence Ridge regression model is when regularisation strength is equal to 1 (2,5 and 3 for Arousal and Dominance models respectfully), the solver is set o 'sag', fit_intercept set to True and normalise is set to True. Also, we added max_iter = 5000 because the sag solver needs a huge amount of iterations. In the left residual graph, we can see model predictions before and after tuning (Figure 2.). After tuning we couldn't see a drastic change in our data prediction.

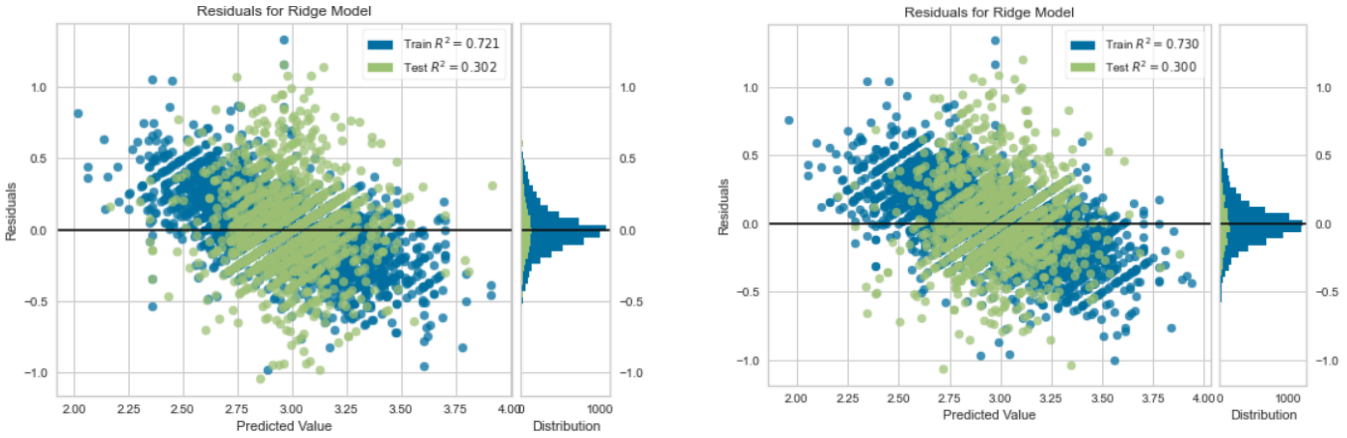


Figure 2. Residuals before and after tunning

Error Analysis

After researching for a bit, we understood we made a mistake by using all of the data in hyper tuning. This brings over-optimistic results. We did it wrong in 2 ways. First by using the same dataset in training and in hypertunning. The consequence is that suboptimal hyperparameters will be chosen. In particular, there will be a bias toward high capacity models that will overfit. Second, data that has already been used to tune hyperparameters is being re-used to estimate performance. This will give a deceptive estimate. The best solution is to introduce a development set, by splitting the training dataset into training and dev sets. After understanding our mistakes we decided to follow this algorithm.

1. training dataset is used to train a few candidate models
2. validation dataset is used to evaluate the candidate models
3. one of the candidates is chosen
4. the chosen model is trained with a new training dataset (combined training with dev)
5. the trained model is evaluated with the test dataset

Also by researching some more we learned that our candidate models were not evaluated properly. Because cv should be used on unseen data - dev set. So we decided to evaluate the data on all candidate models again, but this time using the dev set as well. We got similar results, as before, so the best option is still the ridge model. Thus, we hypertuned the parameters using training and dev set and then evaluated the model with the test dataset, ensuring the avoidance of bias.

Another reason for using a train/dev/test structure of data is that we are planning to introduce another dataset. This way we want to see how much error could data mismatch cause to our model.

Evaluation

During our model improvement process we were always looking at mostly 3 scores to determine whether we are moving in the right direction. Also since we are working with a comparatively small dataset we decided to use k-fold cross-validation. Through trial and error, we started to understand how mean-absolute-error (MAE), mean-square-error (MSE) and r2 scores correlate with each other not only in theory but also in practice. In the beginning, we were choosing feature extraction methods and candidate models mostly by MSE for some reason. After hypertuning the model with MAE we produced this graph which from a superficial glance looks

very well-tuned, all training values are close to testing values. If we look closer we made the model predict everything close to the neutral value (graph on right all values are between 2.9 and 3.05). While a low r2 score shows us that this model tends to discard features. Making our model unfit to predict emotional tweets. That's why we changed our tuning score to r2 so the model would look more closely at features rather than the expected value. As for our final evaluation, we have decided to look closely at all 3 scores (Table 2.). We concluded that the Valence model predicts values best. Although MSE and MSE are a bit higher than the other 2 models, the R2 score is significantly better.

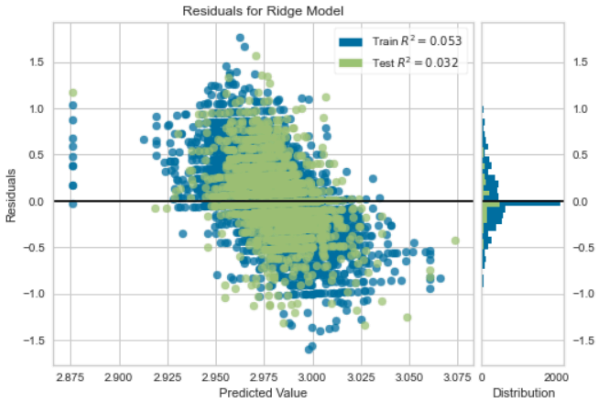


Figure 3. Residuals after tunning with MAE

	Train MSE	Test MSE	Train MAE	Test MAE	Train R2	Test R2
Valence	8.7%	10.4%	21.1%	23.1%	72.7%	32%
Arousal	5.9%	6.5%	18.4%	19.2%	46.1%	11%
Dominance	4%	4.6%	14.9%	15.8%	40%	6.3%

Table 2. Final evaluation of VAD models

We have also evaluated the dataset ourselves following some simple rules of thumb. If a tweet sounds positive it should have high Valence and vice versa. If a tweet has a lot of energy it has high Arousal. If a tweet feels powerful and dominant it should have high Dominance. After doing some experiments we noticed that Arousal and Dominance models predict mostly neutral values, while the Valence model is able to predict more emotional tweets (Table 3.). This seems to agree with our evaluation table above.

Tweet	Valence	Arousal	Dominance
I LOVE VLAD	4.034244	3.473958	3.090888
fun fun fun, EXCITED!	3.764874	3.310719	3.159180
You disgust me	2.692486	3.268714	3.068745
I have so much energy!	3.075187	3.044524	3.095407
I feel calm	2.921721	2.944817	2.966800
I am so ANGRY	2.947362	3.155442	3.103365
I fear of dark	2.665727	3.132917	2.963477
NEUTRAL	2.919893	3.024405	3.027902

Table 3. Examples of predicted VAD values

Combining Datasets

The reason to introduce one more dataset called Emotion detection from text is to have the correct target data (Pashupati Gupta, 2021). Because the data we are working with right now is only sentences that have nothing to do with tweets. But by introducing this data set with actual tweets, we can evaluate how well we are doing on the application we actually want to apply our model. We did not use this dataset at first because it did not have VAD values, only categorical sentiments. Combining these two datasets might sound like a problem, but we found 2 datasets called ANEW and Norms of valence, arousal, and dominance for 13,915 English lemmas dataset which has words associated with VAD values (Bradley et al., 1999, Warriner et al., 2013). This way we were able to combine these datasets to create a bigger and more application targeted dataset.

When we combined the datasets, we instantly observed that the mean squared error increased dramatically and the r2 score decreased by a lot (Figure 4.). Meaning that model is more inclined to ignore features. Basically, the model started to try to classify text into 13 constant VAD values. After seeing that, we decided not to combine the datasets, as to avoid unreliable model prediction.

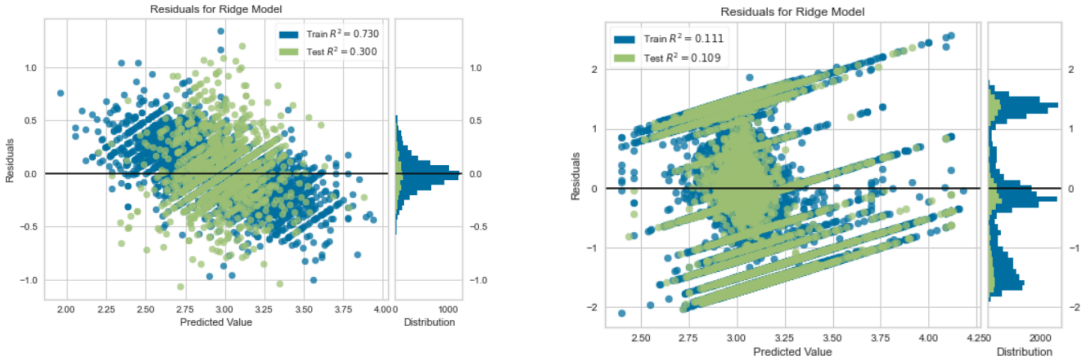


Figure 4. Residuals before and after combining data sets

Multiclass tweet emotion categorization

First attempt

In this part of the project we started working with a dataset on kaggle.com "Emotion detection from text" (@PASHUPATI GUPTA, 2021). This dataset contained 40000 rows and 3 columns: tweet_id, content, and sentiment. The latter had 13 different emotion categories. The distribution of data was very unequal. First simple experimentations using this dataset on the classifier gave very high loss and low accuracy (~30%), thus we decided to create our own dataset from the real data on Twitter.

Data collection

For data collection, we used Twitter API 2.0. We decided to classify tweets based on the hashtags so only tweets with hashtags of certain emotional words were filtered. For an easier classification problem, we chose to use 6 main emotions, but later on, added 2 more categories "love" and "neutral". Labelling tweet emotion based on the hashtag raises some accuracy problems. First, a tweet sometimes has multiple hashtags with different emotions. To avoid ambiguity we removed such tweets from the dataset. Second, a hashtag with emotion does not necessarily represent the true emotion, for example, when the user expresses sarcasm or irony by writing something sad in the main text and adding a "happy" hashtag. However, this problem cannot be resolved with simple methods and we assumed that these kinds of tweets will serve us as noise in the dataset. Additionally, we believe that there is a potential for future studies to recognise the sarcasm or irony in such cases using highly accurate emotion categorization or VAD prediction for the main text and comparing it to existing hashtags.

Auto labelling for emotion categories was done in 2 ways: tags-based and emoji-based. In 2012 a paper that first introduced automated emotion labelling by emotion-related hashtags reached an accuracy of 65.57% using 2 million tweets (Wang et al, 2012). We chose to apply the same method and used 5-6 words expressing an emotion, one of which is the exact category word in noun form (i.e., sadness), one is the adjective of the same word (i.e., sad), and the rest are the synonyms of the main category as suggested by the Wang et al. (2012). The category "neutral", however, was created by criteria of not containing emotion-related hashtags.

The second method, emoji-based auto labelling, was chosen for experimentation purposes. We assumed that Twitter users might better express their emotional sentiments by using emojis. Thus, we used the online Twitter emoji list emojipedia.org/twitter/ which provides Unicodes for emojis and their descriptions. We searched for emojis of the same emotion words used in the tags-based method and used 5-6 for each category. The "neutral" category in the emoji-based dataset was not created.

In total, 77912 tweets were collected. Figure 5. illustrates category distribution in the created dataset.

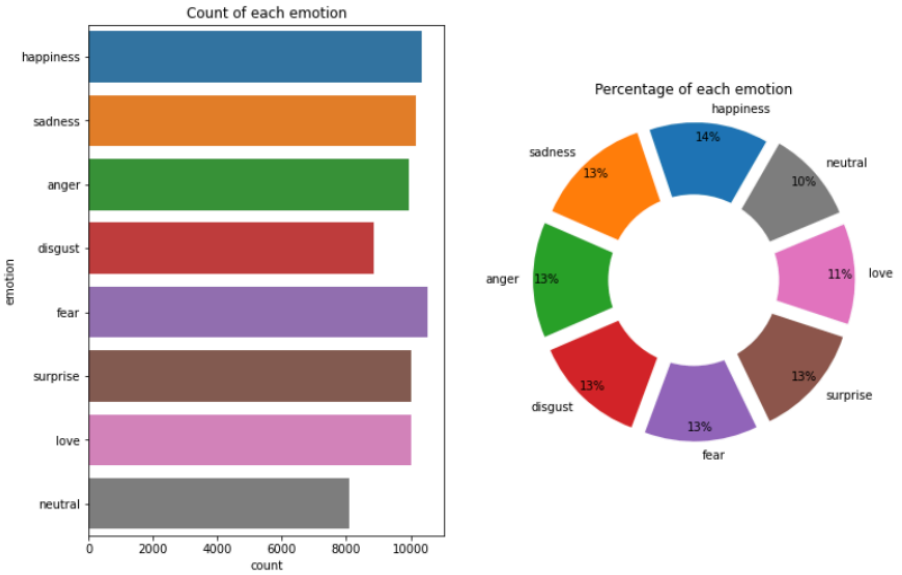


Figure 5. Emotion category distribution in the auto-labelled dataset from Twitter API

Data exploration and preprocessing

From figure 3, we can see that the number of words in tweets is highly positively skewed and most of the tweets tend to be short, 10-15 word sentences. Character-wise tweets contain 0-280 characters, peaking at 50, then 150, and at the limit of characters.

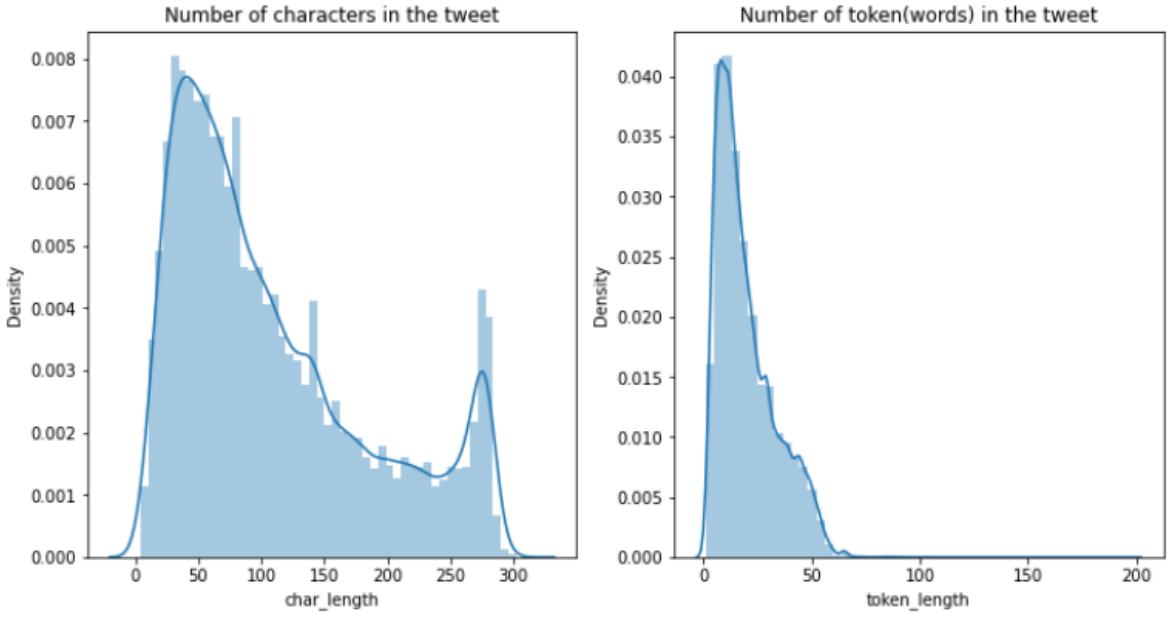


Figure 6. Character length and token length distribution in created dataset

Sentiment-wise character distribution indicates that tweets having “disgust” emotion tend to be the longest, “neutral” tweets tend to be the shortest, while “anger” categorised tweets tend to be the middle, around 140 character length (figure 7.)

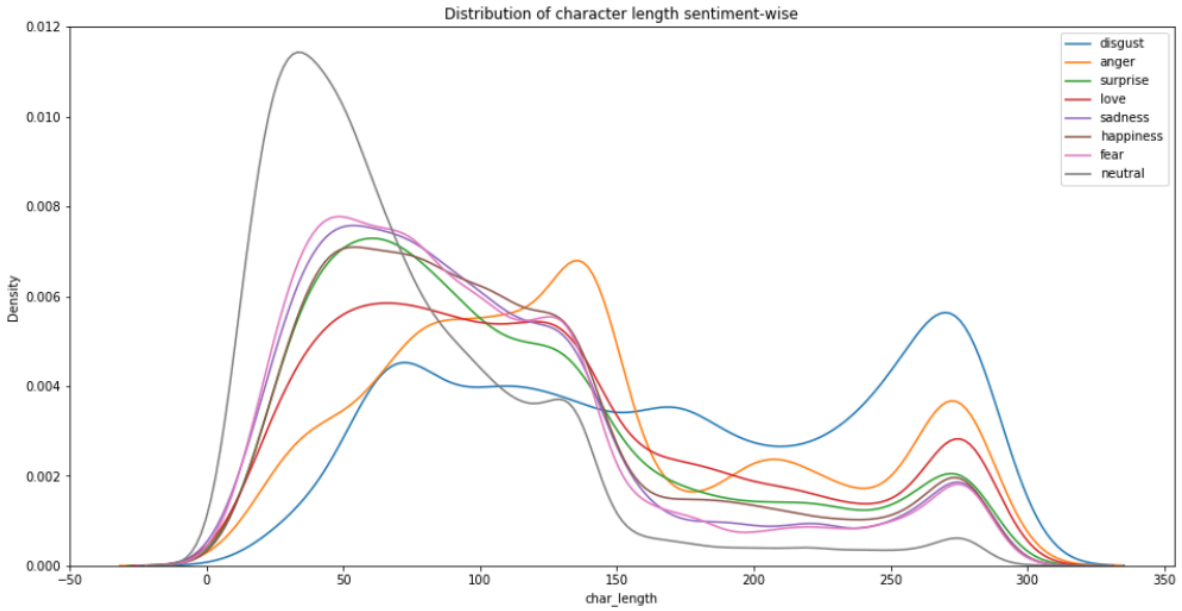


Figure 7. Sentiment-wise character length distribution in created dataset

During data preprocessing, we followed several steps. In the first simple and fast attempt, we did word stemming, lemmatization, and special character removal. After such data preprocessing, various datasets were used to train the model to identify which dataset to work on further. After identifying the best-promising dataset, a full text normalisation was done. For tokenization, we chose to work on a word level, which is a general norm followed by the industry. Non-ASCII words and stopwords were removed from the text so that more focus can be given to those words which define the meaning of the text. In the word embedding, 20 000 of the most frequent features were chosen, which is generally sufficient.

Model building

For the multiclass emotion categorization task Recurrent Neural Network (RNN) model has been chosen. For building the model Tensorflow Keras sequential tool has been used. The model contains an Embedding layer using a Glove (50 dimensional) dictionary, 3 bidirectional Long-Short Term Memory (LSTM) layers with dropout layers in between to prevent overfitting, a dense (fully-connected) layer with 256 hidden units and “relu” activation function, as well as the final dense layer with “softmax” activation function. The bidirectional LSTM layer was chosen because it preserves information from both, past and future, thus better understanding the context of the words and performing better in classifying the category.
Different model hyperparameters were used during the experimentation part. The final model was chosen to be used with 4 epochs and a batch size of 128.

Results

Different datasets with a simple data preprocessing were used to train the model in order to get an understanding of which dataset has the most potential in producing the highest accuracy.
From the table Table 4. we can see that the worst model performance was using our created dataset with emoji-based auto labelling (No 5.). It also contained the fewest data points. The second worst model

performance was using a dataset from Kaggle (No. 3). Auto-labelled using only hashtags dataset showed the best training and validation accuracy, however, during the testing part it still gave low results (~45%).

The combined dataset of data from Kaggle and our created tags-based dataset was chosen for the final model optimization as it gave the best test accuracy on unseen data. First, the dataset was improved by additional data processing, then different hyperparameters were used to get the best results.

No	Dataset	Size	Comments	nClass	Train acc	Val acc	Test acc[1]	Training time
1	Auto-labelled tag_emoji_based	69830	Without "Neutral"	7	0.6949	0.6464	0.43	~18 min
2	Auto-labelled tag_emoji_based	77912	With "Neutral"	8	0.6430	0.5918	0.3875	~12 min
3	Kaggle	40000		6	0.5064	0.3259	0.3	~60 min
4	Auto-labelled tag_based	42911	With "Neutral"	8	0.97	0.97	0.45	~40 min
5	Auto-labelled emoji_based	35001	Without "Neutral"	7	0.3622	0.2981	0.2417	~ 45 min
6	Combined Kaggle+tag_based	77844	Assumed "worry" == "fear", combined "anger" and "hate"	8	0.7129	0.6876	0.4533	~1h 12 min
7	Combined Kaggle+tag_based	77844	full data preprocessing	8	0.6987	**0.6988**	0.4672	~1h 14 min

Table 4. Train-Validation-Test accuracies of the model of different datasets

Despite many experimentations and the use of different datasets, the results of the classifier are not satisfying. The highest accuracy we achieved was 69.88%. Training and validation plots (Figure 8.) show that learning conservers after 4 epochs.

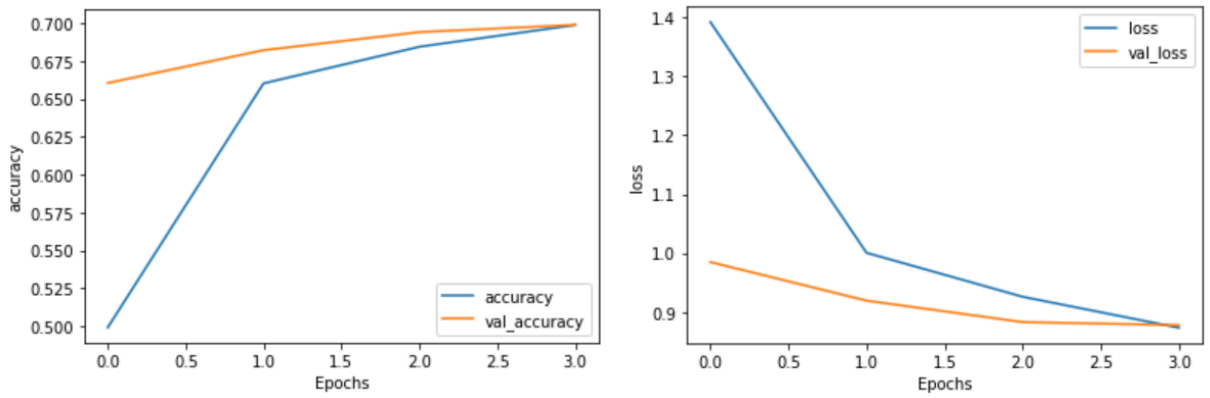


Figure 8. Training and validation accuracy and loss plots of the final model

[1] Data from different day than Train and Validation to check generalisation power

However, then using the model on totally new data from Twitter, the model can correctly identify the emotion of 47/100, which is quite a low accuracy. Experiments showed that having fewer categories might increase the model's accuracy, however, this decreases the practical implementation since there are numerous emotional categories. In the final model categories 'fear', 'happiness', and 'neutral' have the worst precision while 'disgust', 'surprise', and 'anger' have the best precision. More importantly, recall is the worst for 'fear', 'sadness', and 'surprise' categories, while 'disgust', 'anger', 'love', and 'neutral' have the best recall. Overall, 'disgust', and 'anger' are the most accurately predicted categories.

In Figure 9., we can see the examples of tweets, their supposed emotion (from hashtags in the tweet) and predicted emotion.

tweet	predicted	supposed	correct
📌Today the leader of the Republican Party surrendered to Piers Morgan. #sad	sadness	sadness	True
Invest love. 😍 Where you invest your love you invest your life. #lovealert #love #affirmation #quote #goodvibes #motivation #inspiration #encouragement	love	love	True
Golly, I need sooo much help on my #essay I wish there was someone out there to help me 😬 My paper that I need to write is so difficult #sad 😔	sadness	sadness	True

tweet	predicted	supposed	correct
Good night. #life #love #peace	happiness	love	False
Is it normal not to show emotion when someone close passes away? Delayed reaction perhaps? #death #emotionless #Unexpected	fear	surprise	False
#IDC who u are, if your #upset #watching #catvideos will #eventually put a #smile on ur #face	fear	sadness	False

Figure 9. Correctly and incorrectly predicted emotion categories by the final RNN model

Melody Generator - Attempt 1 (Making Music)

Data Collection and Engineering

The training and testing data used for this model were pieces by different composers found in the Classical Music MIDI dataset.

In order to familiarise ourselves with sound file manipulation in python we started by trying out the easiest method: working with the music21 library and using inbuilt midi-processing functions. After conversion, we are left with a stream of chords organised based on instruments. Each cord is composed of multiple notes, so we used a function to extract the string of notes from the stream as a vector. We then built a dictionary of all possible notes by indexing them.

Model Building

Due to the nature of the problem as an analyser and generator of time series data, we consulted the available literature and quickly concluded that in order to have the most efficient generative model, making use of a Long Short-Term Memory recurrent neural network would be optimal. Designed by Hochreiter & Schmidhuber (1997), the architecture of an LSTM model utilises a feedback connection as well as a feedforward connection, solving the vanishing gradient problem (which in traditional RNNs can prevent the network from training due to the partial derivative of the error function with regards to the current weight being too small, or in other words, *vanishing*).

We scoured the internet for different examples we could work with and found this kaggle repository to be the most informative for our project.

In order to train the model, our first attempt was done by only utilising pieces by Schumann, which produced this initial result. We were happy with our initial song (although it sounds as if a 5-year-old composed it...), but we decided to use the entire Classical music library to see if it would make a difference. We will let you be the judge of that.

Model Analysis

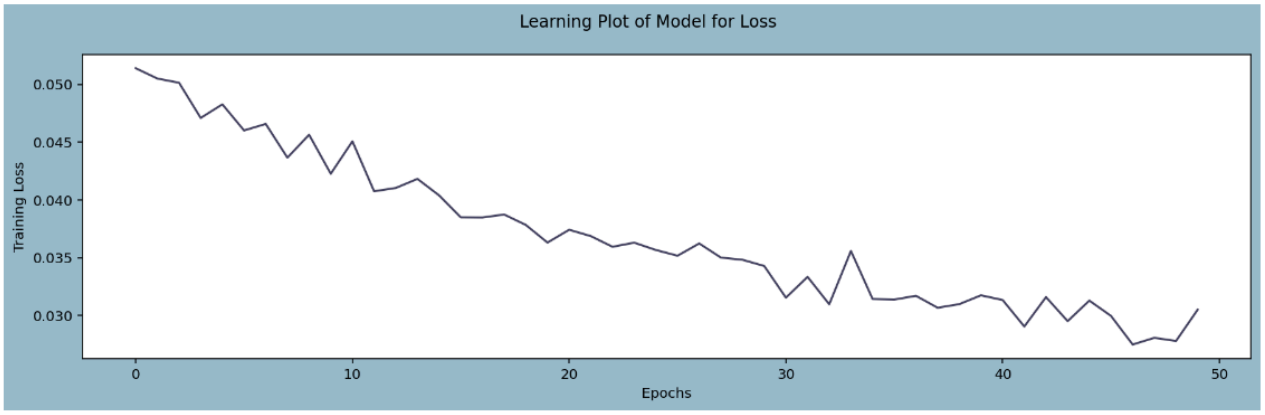


Figure 10. LOSS PLOT Schumann Model

From this figure we can see how hypertuned our model was to Schumann's pieces, starting with an already minuscule loss value. The graph does not look like a healthy loss plot and the only conclusion we can infer is that artists truly do have very specific styles, which an AI model can pick up on.

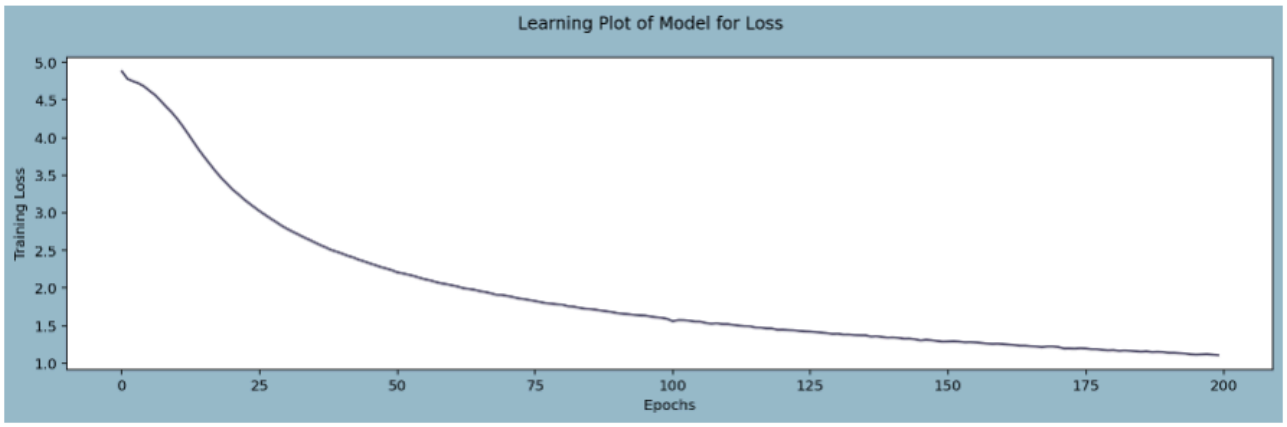


Figure 11. LOSS PLOT All Composers Model

The loss plot looks way better now that we have included multiple composers, but despite this, the songs produced still sound awkward. It is a great first step, but it made it clear that we need to employ more advanced methods and dig deeper within the field.

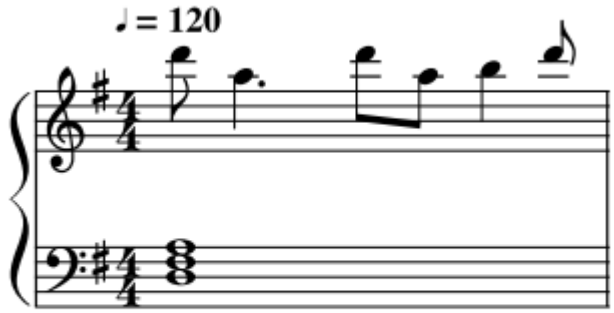
Melody Generator - Attempt 2 (Binary Generator)

Data Collection and Engineering

For this model we used Ferreira's VGMIDI data-set and the MUSIC-SENTNEURON model. The dataset is composed of annotated video game tracks, using only V/A values (this seems to be the standard in the

literature, dominance being abstracted in sentimental music generation models). Although the cohort rated the V/A-values using a 2D continuous Cartesian model, for the purpose of their work, Ferreira turned away from regression and decided to describe each piece as either positive or negative sentiment wise.

The unique part about the data manipulation is that instead of using traditional music21 midi encoding methods, we utilised a special method developed by Ferreira (2019).




```
t_120 v_76 d_whole_0 n_50 n_54 n_57
v_92 d_eighth n_86 . . v_84
d_quarter_1 n_81 . .
```

Here, we have the following symbols:

- n_[pitch]: play a note with a given pitch (int range 0 to 127)
- d_[duration]_[dots]: change the duration of the following note with a certain amount of dots
- v_[velocity]: change the loudness of the note (int range 4 to 128, bins of 4)
- t_[tempo]: tempo of the piece (int range 24 to 160, bins of 4)
- ".": end of timestep (1/16 of a note)

Model Building

The second attempt used the first-ever object/goal-oriented generative LSTM model with regards to music time series (Ferreira & Whitehead, 2019). In order to achieve this, they realised that they could build a generative LSTM, and train a sentimental logistic ***classifier***, which then alters the activated neurons in the LSTM by using a genetic algorithm. Essentially, this method creates two models linked by the genetic algorithm. The LSTM has first been trained to output a coherent time series of notes (a stream), which is then analysed by the logistic classifier in order to see which neurons get activated and how when the stream is positive or negative (binary classification). This then allows for the generation of music with the binary sentiment.

The generative model itself is refined, as the authors use a multiplicative LSTM (mLSTM), rather than a simple LSTM, the key difference being that in a multiplicative recurrent neural network, the current input affects all hidden-state dynamics, by having a specific hidden-to-hidden weight matrix for ***each*** input (defined as a function of the current input), rather than a general one.

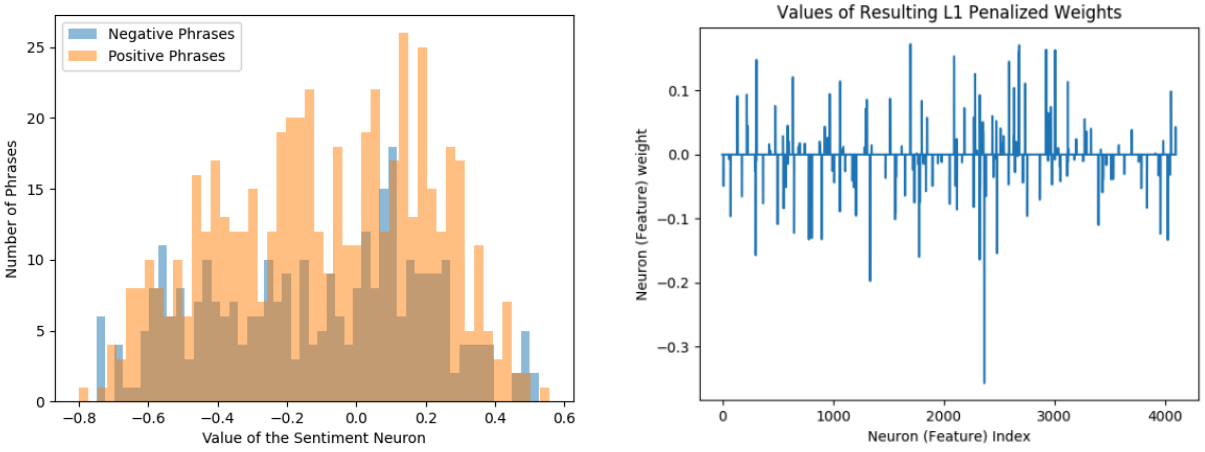


Figure 13. Example of a specific neuron (left) and results of all neurons (right)

In figure 13 (left), we showcase an ordinary neuron's number of phrases based on sentimental values, the classifier utilising this information to inform the genetic algorithm on how to override neurons depending on the desired (positive/negative) generation. On the right, you can see this extrapolated for all neurons, some of them being clear controllers for negative generation, others for positive and some dead neutral.

Model Analysis

Sadly, due to the GPU farm restarting during the night after 8 hours of generative model building, we have lost the loss plot and only have access to the model checkpoint (which does not provide a history of the loss value for previous epochs).

As you can probably remember from our project review meetings, we had plenty of issues with utilising this model. We have worked on debugging it for the better part of this project, yet even when it finally became functioning, the generated midis were absolutely horrific. We suspect it might be an issue with encoding the text back into a midi sound file.

Still, despite the inability to reproduce the results of this paper, we present a generated positive piece, as well as a negative piece, courtesy of the author.

Method	Test Accuracy
Gen. mLSTM-4096 + Log. Reg.	89.83 ± 3.14
Sup. mLSTM-4096	60.35 ± 3.52

Table 2: Average (10-fold cross validation) sentiment classification accuracy of both generative (with logistic regression) and supervised mLSTMs.

The paper provides an analysis of the sentiment classification accuracy, the GA mLSTM (with LR) being far superior and learning sentiment through unsupervised associations between neuronal activation patterns (and associated weights)

Melody Generator - Attempt 3 (Multi-Class Generator)

Data Collection and Engineering

The EMOPIA project utilises a dataset of pop piano pieces, annotated with V/A values by four independent experts. The unique aspect of their approach is that they break up songs into song clips and annotate the respective clips, which are then used to train their model. The innovation with regards to this dataset is that it is the third of its kind to provide VA annotated MIDI files, the key difference to the previous ones (VGMIDI, discussed earlier and MOODetector) being the sheer sample number (n = 1078).

The difference to the previous model is that rather than building a binary generator, EMOPIA utilises Russel's (1980) model of emotion, which is composed of a 2D Cartesian space, with 4 quadrants indicating a combination between high/low valence and arousal.

Given that the dataset contains exclusively piano pieces, in order to encode the midis, a transcription function built by Kong et al., (2021) was used and then the midi files were encoded as tokens belonging to families (see figure 11.c and 11.d below, as well as 11.a and 11.b for comparison). This was done building on previous object-oriented generation research which has shown the benefits this provides (Keskar et al., 2019).

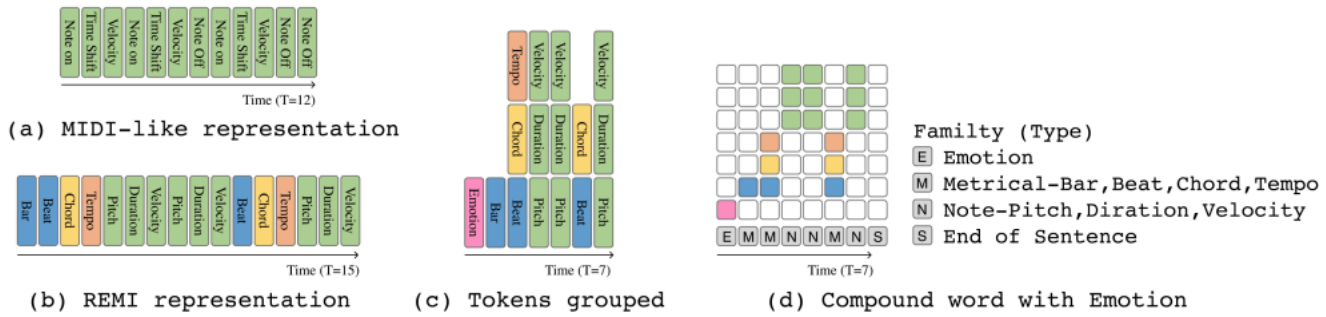


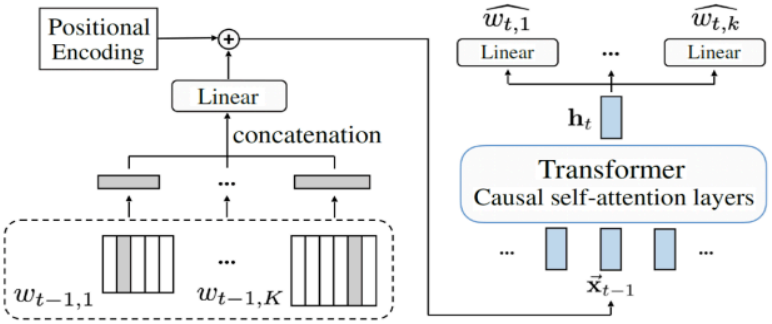
Figure 1. Illustration of different token-based representation for symbolic music: (a) MIDI-like [39], (b) REMI [34], and (d) CP [35] plus emotion token. Sub-figure (c) is an intermediate representation of the CP one.

Figure 14. Data Encoding of MIDI files for the purpose of model generation

Model Building

EMOPIA adapted the transformer model built by Hsiao et al., (2021) and improving it by utilising the aforementioned data representation tactic. They also compared results with Ferreira's previous state-of-the-art sentimental music generator and showed how this new method improved both objective and subjective methods (see below for model analysis)

The model architecture itself is quite complex, but it essentially combines the token embeddings from different families at each timestep to feedforward to the causal self-attention layers which predict the output at the next timestep. This ensures that the transformer considers macro factors (families) and micro factors (the tokens themselves), a great improvement to event-based generation (which was done with previous midi encodings)..



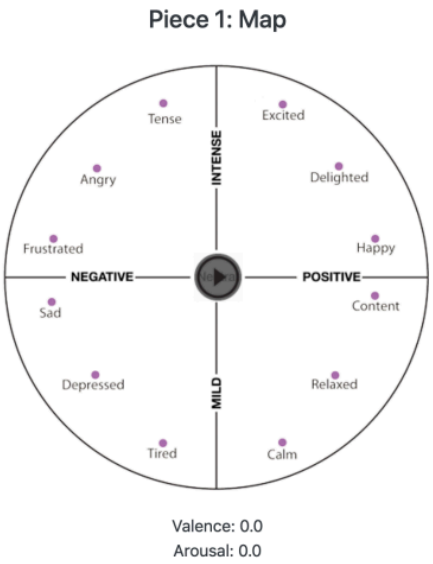
Model Analysis

Model	Objective metrics						Subjective metrics		
	PR	NPC	POLY	4Q	A	V	Humanness	Richness	Overall
EMOPIA (i.e., real data)	51.0	8.48	5.90	—	—	—	—	—	—
LSTM+GA [10]	59.1	9.27	3.39	.238	.500	.498	2.59±1.16	2.74±1.12	2.60±1.07
CP Transformer [35]	53.4	9.20	3.48	**.418**	**.690**	.583	2.61±1.03	2.81±1.03	2.78±1.03
CP Transformer w/ pre-training	**49.6**	**8.54**	**4.40**	.403	.643	**.590**	**3.31±1.18**	**3.22±1.23**	**3.26±1.15**

Table 6. Performance comparison of the evaluated models for emotion-conditioned symbolic music generation in *surface-level objective metrics* (Pitch Range, Number of Pitch Classes used, and POLYphony; the closer to that of the real data the better), *emotion-related objective metrics* (4Q classification, Arousal classification, Valence classification; the higher the better), and *subjective* metrics (all in 1–5; the higher the better); bold font highlights the best result per metric.

Figure 15. Model Accuracy Comparisons (from both subjective and objective measures)

The metrics presented by their paper speak quite well for the accuracy and efficiency of this generative model, but what is the strongest argument in favour of the EMOPIA generator is simply listening to the different outputs it is capable of generating. Here are samples from each category: Q1, Q2, Q3, Q4.



For reference, here is the 2D Cartesian space of V/A values and how they correspond to different emotions. Q1 represents High Valence and High Arousal, Q2 Low Valence and High Arousal, Q3 Low Valence and Low Arousal, Q4 High Valence and Low Arousal.

Limitations

Sentiment analysis, both for categorical, and continuous data, have major limitations of accurate datasets, which additionally challenges NPL problems. The main reason for accurate datasets limitations is the subjectivity of emotions themselves and the difficulty to identify the ground truth. We had only 10 000 data points for text-VAD pairs which seem to be too little to get very accurate predictions. For the multiclass categorization, we were able to make assumptions that emotion-related hashtags in the tweets will reflect the true emotion of the post, thus a larger unique dataset was created. However, this process also had many limitations and the assumptions themselves hold errors. Furthermore, while constructing the dataset on a specific day or several days the sentiments of the tweets are influenced highly by the news of that day. For example, when collecting tweets on the Oscars awards day, the vast majority of tweets with “disgust” sentiment included “Will Smith” and “Chris Rock” because of the slapping incident. The model learns very specific words and thus, after a while, the “disgust” category is not accurately identified while predicting emotion for tweets.

With regards to the latter part of our project, the earliest available paper on the topic of object-oriented generation (in our case, sentimental music), comes from 2019, with more refined models being produced in 2021. When we first delved into this topic, we had not realised how difficult the problem of continuous V/A/D generation would prove to be, yet as we worked along, we slowly acknowledged that we would have to tune down the scope of our goals. Still, we are very proud to be able to generate sound given a multi-class emotional input, derived from the sentimental analysis of tweets.

A possible limitation with regards to sound generation comes from the hyper tuning of our final model to piano pieces. This has pros, in that it makes it easier to transcribe the data and less difficult for the model to produce a convincing piece as a result (which we would claim passes the Turing Test!), but cons in that, well, the outcome contains exclusively one instrument.

We conclude by echoing the consideration that all papers we have analysed through our literature search have put forward. Future research should concern itself with utilising the existing datasets to build models that can

take continuous inputs, models that can analyse and produce songs with multiple instruments and, finally, models that can do both.

Bibliography

Buechel, Sven & Hahn, Udo. (2017). EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis. 10.18653/v1/E17-2092.

Bradley, Margaret M. and Peter J. Lang. “Affective Norms for English Words (ANEW): Instruction Manual and Affective Ratings.” (1999).

Ferreira, L. N., & Whitehead, J. (2021). Learning to generate music with sentiment. *arXiv preprint arXiv:2103.06125*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

Hsiao, W. Y., Liu, J. Y., Yeh, Y. C., & Yang, Y. H. (2021). Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs. *arXiv preprint arXiv:2101.02402*.

Hung, H. T., Ching, J., Doh, S., Kim, N., Nam, J., & Yang, Y. H. (2021). EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation. *arXiv preprint arXiv:2108.01374*.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Kong, Q., Li, B., Song, X., Wan, Y., & Wang, Y. (2021). High-resolution Piano Transcription with Pedals by Regressing Onset and Offset Times. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *29*, 3707-3717.

Panda, R. E. S., Malheiro, R., Rocha, B., Oliveira, A. P., & Paiva, R. P. (2013). Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis. In *10th International Symposium on Computer Music Multidisciplinary Research (CMMR 2013)* (pp. 570-582).

Pashupati Gupta. (2021). Emotion detection from text. Retrieved from Kaggle (CC0: Public Domain)

Russell, J. A. (1980). A circumplex model of affect. *Journal of personality and social psychology*, *39*(6), 1161.

Mehrabian, A., & Russell, J. A. (1974). An approach to environmental psychology. The MIT Press.

W. Wang, L. Chen, K. Thirunarayan and A. P. Sheth, "Harnessing Twitter "Big Data" for Automatic Emotion Identification," 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing, 2012, pp. 587-592, doi: 10.1109/SocialCom-PASSAT.2012.119.

Warriner, A.B., Kuperman, V. & Brysbaert, M. Norms of valence, arousal, and dominance for 13,915 English lemmas. Behav Res 45, 1191–1207 (2013). https://doi.org/10.3758/s13428-012-0314-x