Twitter Sentiment Mini Project
Eric Chang and Naif Ganadily

Contributions: Both members worked on all parts of the project and would share details about each part with each other to maximize learning and get the 'best' model for the project.

1. Basics and Visualization

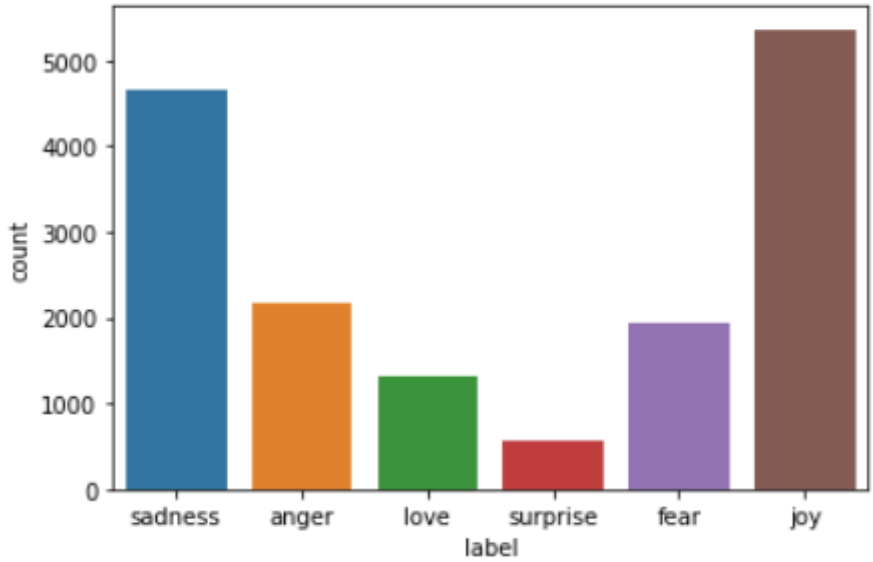

Figure 1: Emotion Class Distribution

The data is a dataframe of two columns. Each row contains a sentence and its respective sentiment based on six different emotions: sadness, joy, love, anger, fear, and surprise. Each of these emotions are represented by a number value. There are 16,000 sentences in the training set, 2,000 in the second set, and another 2,000 in the kaggle test set.

2. Logistic Regression

There are many ways to process the data. The first thing to do is to tokenize each of the sentences. We used the Tokenizer() function from the tensorflow preprocessing text library to create tokens of each word. Then, we changed each token to a value with the texts_to_sequences function. Finally, we padded the sequences using pad_sequences() to make sure our inputs are the same length across the board. The max length for the padded sequence is 50 since none of our sentences are larger than 50 tokens.
Another way we tried was to tokenize the sentences, and then also remove stopwords, punctuation, numbers, etc. However, this method proved to be incompatible with a lot of pretrained models. It also showed lower results from the data set.

Running the logistic regression was pretty easy to do. We used sklearn's library for its logistic regression function. Even splitting this training set into a training and validation set did not change the accuracy of the model.

Accuracy: 0.8815625

	precision	recall	f1-score	support
0	0.92	0.93	0.92	946
1	0.88	0.93	0.90	1021
2	0.81	0.74	0.77	296
3	0.90	0.86	0.88	427
4	0.84	0.83	0.83	397
5	0.82	0.67	0.74	113
accuracy			0.88	3200
macro avg	0.86	0.83	0.84	3200
weighted avg	0.88	0.88	0.88	3200

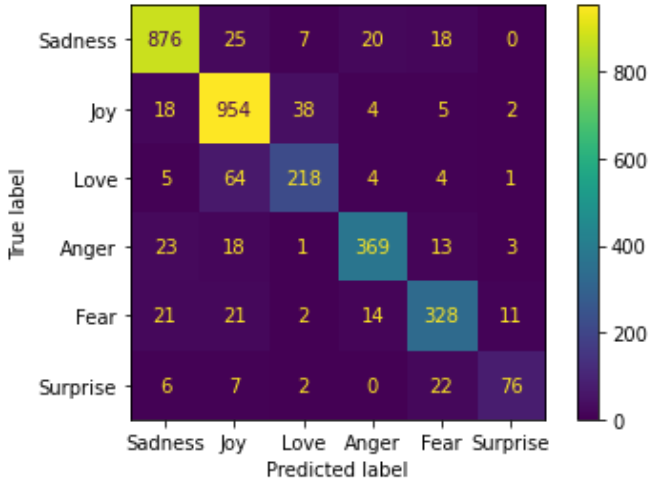

Figure 2: Logistic Regression for Local set (aka the training set) results (top) with its confusion matrix (bottom)

3. LSTM

The LSTM implemented is shown below. Preprocessing is the same as was for the logistic regression model.

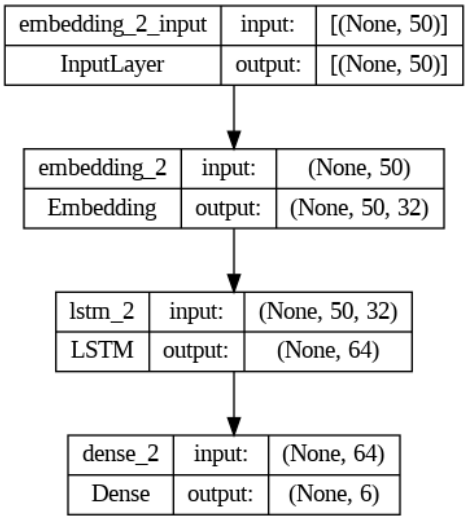

Figure 3: LSTM for local set (aka the training set)

A bi-directional LSTM was also tested, but was unable to produce any better results than this one.

4. Transformer-based

The transformer used is taken from a pretrained model from Hugging Face from the user 'bhadresh-savani'. This model is trained on the same dataset so we expected the results to be very good. This model is a reduced size (around 40% reduced) of the BERT model and was finetuned on the emotion dataset. It uses a learning rate of 0.00002, batch size 64, epochs 8 as hyperparameters for fine tuning. The full name of the model is DistilBERT-base-uncased-emotion.

Training Classification Report:

	precision	recall	f1-score	support
anger	1.00	1.00	1.00	2161
fear	1.00	1.00	1.00	1977
joy	1.00	1.00	1.00	5310
love	1.00	1.00	1.00	1353
sadness	1.00	1.00	1.00	4657
surprise	1.00	1.00	1.00	542
accuracy			1.00	16000
macro avg	1.00	1.00	1.00	16000
weighted avg	1.00	1.00	1.00	16000

Figure 4: Transformer results on the training set.

5. Insights

Model	Pros	Cons
LR	Quick, easy, applicable for all cases even for text learning	Not always accurate, not great at retaining context
LSTM	Can preserve context, Applicable for series/sequence data	Takes resources and time compared to LR
Transformer	The best at retaining context, has many pretrained models to use, can be applied to time series data too.	Takes a long time to train, requires a huge amount of data to train (needs a large corpus)

We can tell that there is overfitting. One of the reasons for this is the class imbalance that occurs between the different emotions. To counteract this, we can either undersample the majority or oversample the minority. We chose to undersample because it would allow for quicker training times. We took 500 random samples from each class and made a dataset of that. Then all preprocessing was done on this smaller data set for all the models. However, after running this result, we learned that undersampling causes a lower performance. This is because we do not have enough data when we undersample the data. As a result, we should look to oversample as a method of dealing with this class imbalance.

6. Tabular Results

Table I: Results of our models based on the test set.

Model	F1-Score	Precision	Recall	Accuracy
LR	0.84	0.86	0.82	0.88
LR (undersampled)	0.75	0.75	0.76	0.75
LSTM	0.78	0.79	0.78	0.84
LSTM (undersampled)	0.62	0.63	0.61	0.61
Transformer	1.00	1.00	1.00	1.00
Transformer (undersampled)	0.99	0.99	0.99	0.99

7. Scalability

A big bottleneck for logistic regression for the case of understanding text is the fact that LR does not retain contextual information. As a result, this model does not perform as well as LSTM and transformers. However, a huge bottleneck for both LSTM and Transformers is the hardware problem, where running these models cost a lot of time and resources. LSTMs are faster than transformers, but they are still slower than simple models like LR. Since our transformer was already pre-trained, running it was fine. However, creating a new one is definitely costly.
As a result, if we are to scale up to one million data points, we can definitely see that LR might be the way to go for our case. However, if we were to invest in more money for hardware, LSTM or even transformers would be much better.
In order for LSTM and transformers to scale with this amount of data points, we may need to increase the number of layers within the architecture.

8. Plot

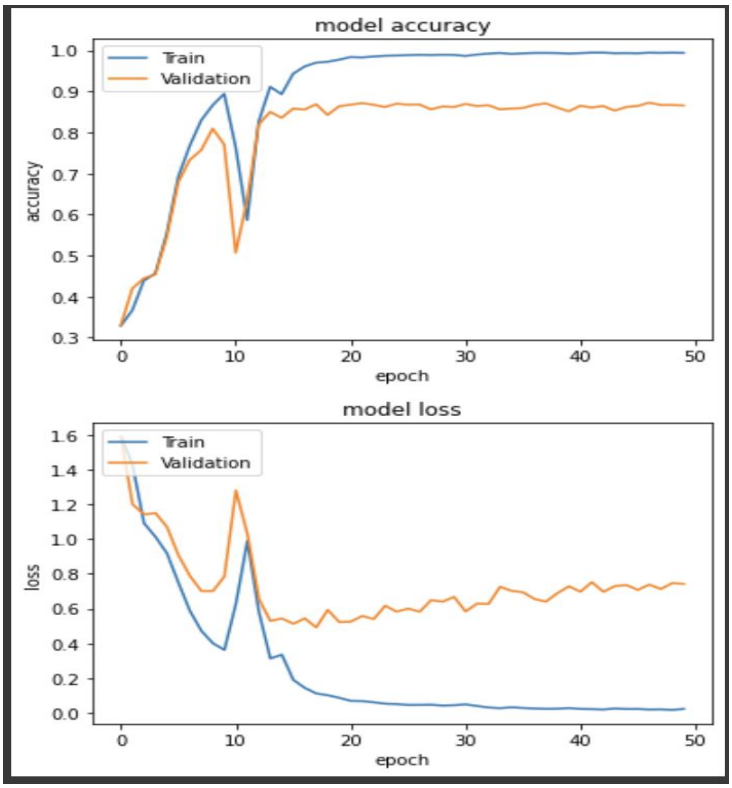

Figure __: LSTM training and validation accuracy plots per epoch. (We use our test split in the table of results.) (Unable to show transformer plots because we did not train it ourselves)

Table II: Results of our models based on the test set.

Model	F1-Score	Precision	Recall	Accuracy
LR	0.84	0.86	0.82	0.88
LR (undersampled)	0.75	0.75	0.76	0.75
LSTM	0.78	0.79	0.78	0.84
LSTM (undersampled)	0.62	0.63	0.61	0.61
Transformer	1.00	1.00	1.00	1.00
Transformer (undersampled)	0.99	0.99	0.99	0.99

9. Interpretability

Transformer model misclassifications

```
i honestly feel at heart we should be faithful to each other if its yo girl
Truth: joy
Prediction: love

i resorted to yesterday the post peak day of illness when i was still housebound but feeling agitated and peckish for brew a href http pics
Truth: anger
Prediction: fear

i feel for you my beloved master time will tell you this is true
Truth: joy
Prediction: love

i knows is the boy makes her feel weird and yuuki doesnt know what to tell her
Truth: surprise
Prediction: fear
```

Some of these misclassifications occur because the two emotions are very closely related. For example, joy and love are very similar in that they are both positive emotions and they tend to be used in very similar situations. In a way, some people will consider these predictions ‘correct.’ Another reason for some errors could come from the corrupted data. In the Transformer model, second example, ‘href http pics’ is not a valid part of the sentence. The model could have interpreted this incorrectly and caused the misclassification.

Another problem we noticed was that removing stop words and punctuation might actually cause problems in the model. Sometimes, sentences would end up being a bunch of adjectives and nouns that make no sense without the context.

10. Kaggle 2 Method

In order to implement this, research was done on the zero-shot classification. The goal of this is to see if our model can predict other related emotions based on the classes it has received from training. In our case, our models know sadness, joy, love, anger, fear, and surprise. However, it does not know words like morose, jubilant, and ecstatic. Here, we know that we can use cosine-similarity to allow for these relationships between the new words and the trained ones.

One method was more of a brute force method where we would use the glove embeddings (glove.6B.50d) to produce word vectors. We found the vectors that represent the 6 different known emotions. The data set is a set of sentences with its corresponding list of possible emotions to pick from. We found word vectors to each of these possible emotions and would run the cosine-similarity between the predicted emotion from one of our models with each of these possible emotions. The maximum value would be the chosen emotion for this method. One problem that came up with this method is that not all of the words found in the possible emotions are in the glove embeddings. One very apparent word that is missing is ‘flabbergast.’ There is a possibility that other tenses like ‘flabbergasted’ exist in the vector but it is not found. Some ways to solve this problem would be to 1) find synonyms 2) create our own word vector. One problem with this method is that GloVe embeddings do not keep the context that the transformer, or LSTM would keep as the model runs. This poses a problem for understanding specific words since opposite words might show high similarity.

Another method is to find a zero-shot classification model. We found roberta-large-mnli which is a pretrained model from Hugging Face. This model was fine-tuned on the Multi-Genre Natural Language Inference corpus and is from the model RoBERTa. RoBERTa was trained on 5 different datasets. Similar to before, we input the sentence and the possible emotions. Once we run the classifier from roberta-large-mnli, we get an output which gives a value of each possible emotion. The maximum value of this would give the final emotion we pick.