WEEK 31 ASSIGNMENT SUMMARY

RAM SUNDAR

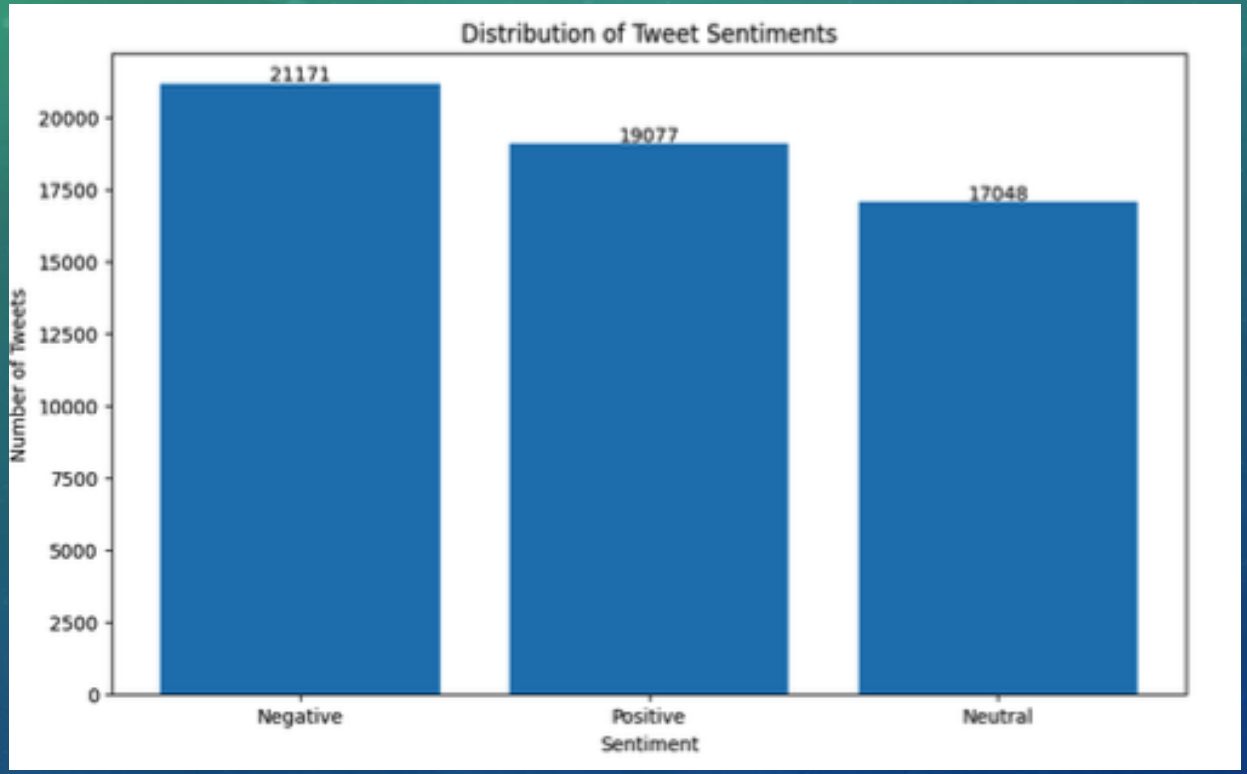
OBJECTIVES

- Understand the structure of the Twitter dataset
- Perform data preprocessing and text cleaning
- Conduct exploratory data analysis to uncover insights
- Build a Recurrent Neural Network (RNN) model for sentiment classification
- Evaluate and improve the performance of the model
- Present findings and recommendations

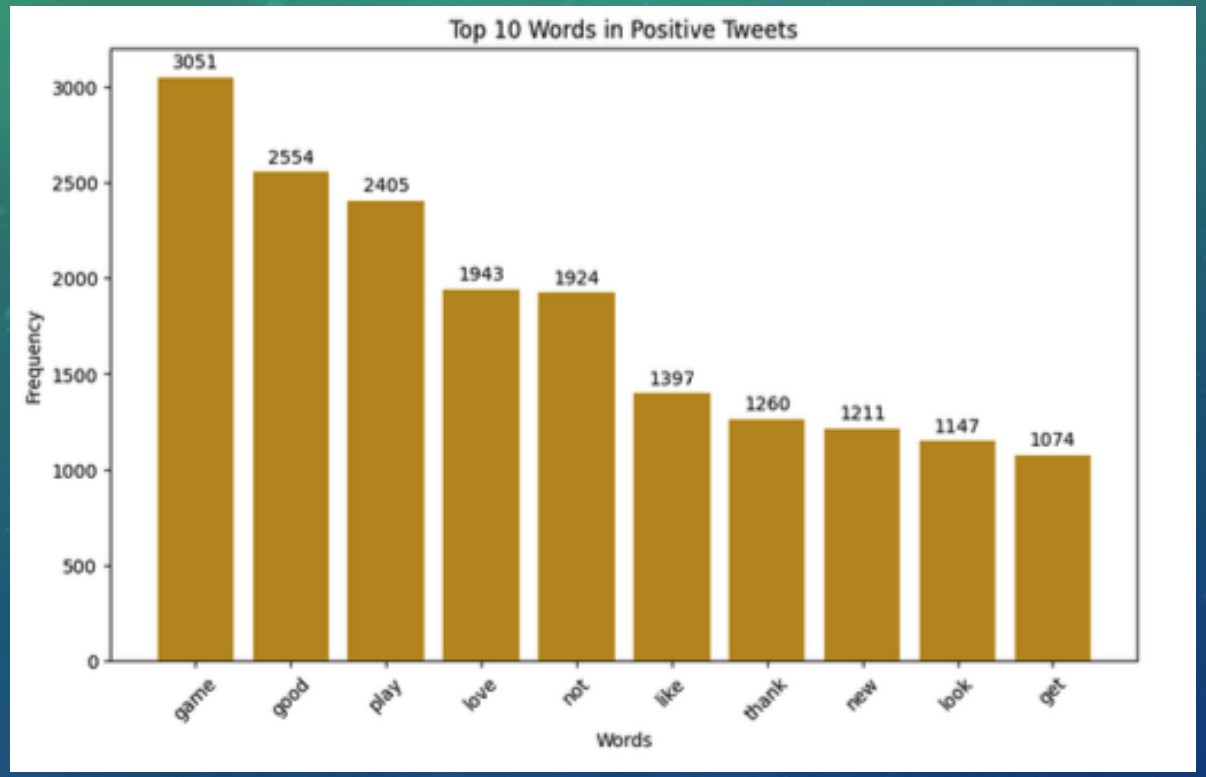
DATASET OVERVIEW

- Standard CSV without headers
- Appears to be from multiple sources
- Dataset observations
  - Total Rows: 74681
  - Total Columns: 4
  - Duplicate Rows: 2700 (3.62%)
  - Total Missing Values: 686
  - Completely Empty Rows: 0
- Dataset after post processing
  - Total Rows: 57296
  - Total Columns: 2
  - Duplicate Rows: 0 (0.00%)
  - Total Missing Values: 0
  - Completely Empty Rows: 0

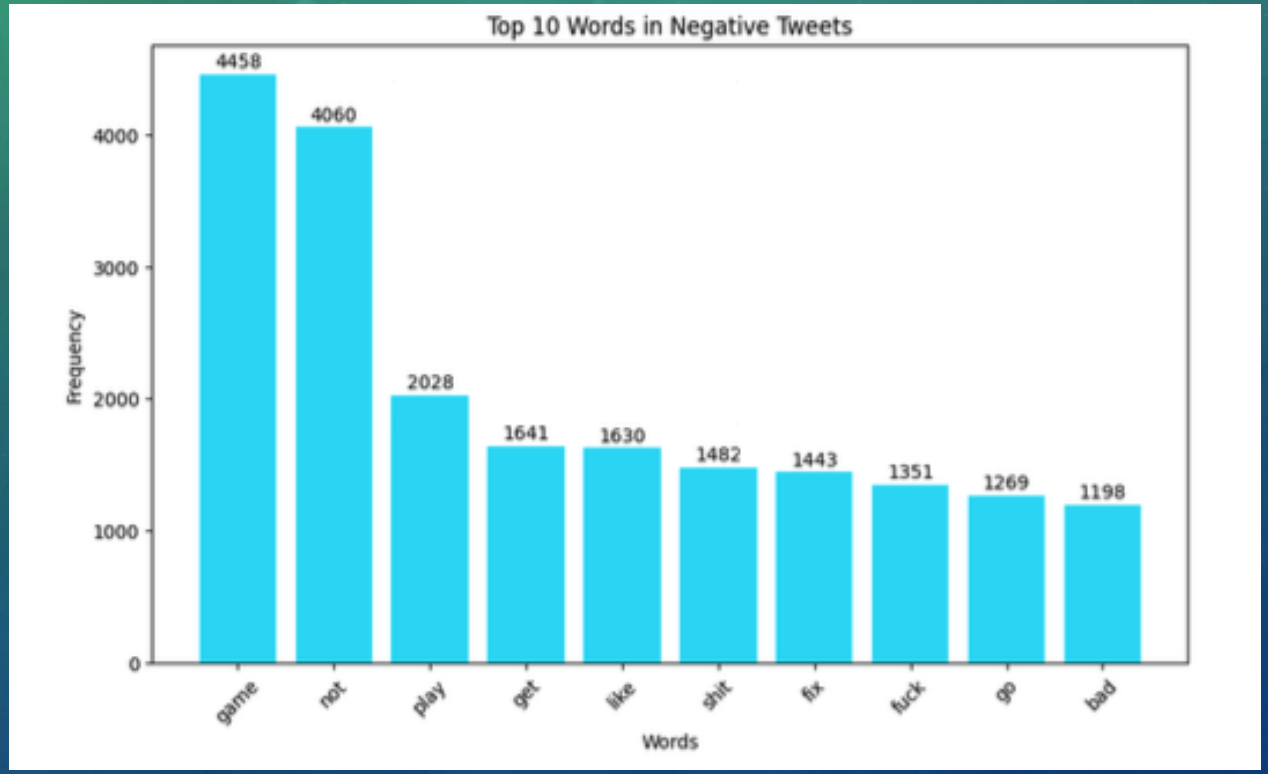
EDA – KEY FINDINGS – SENTIMENT DISTRIBUTION



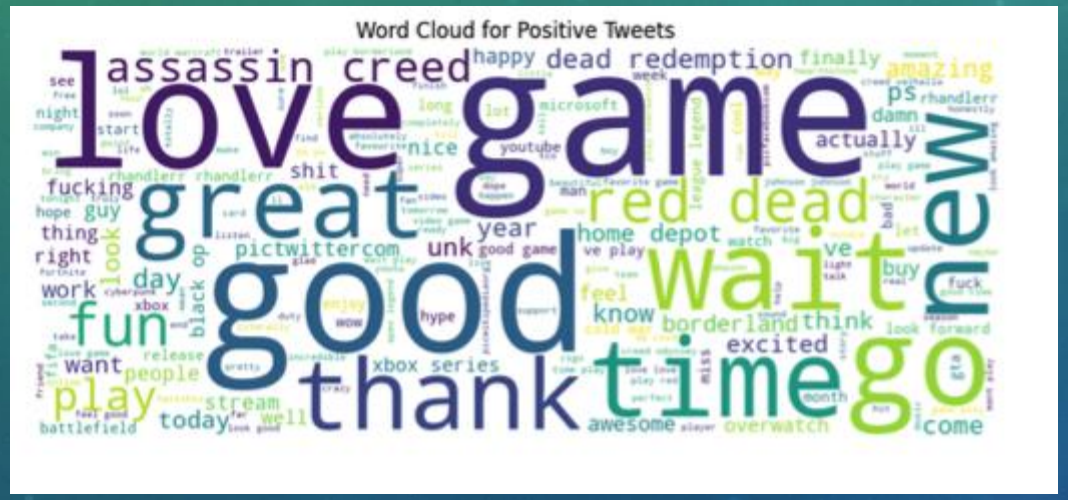
EDA – KEY FINDINGS – TOP 10 POSITIVE KEYWORDS



EDA – KEY FINDINGS – TOP 10 NEGATIVE KEYWORDS



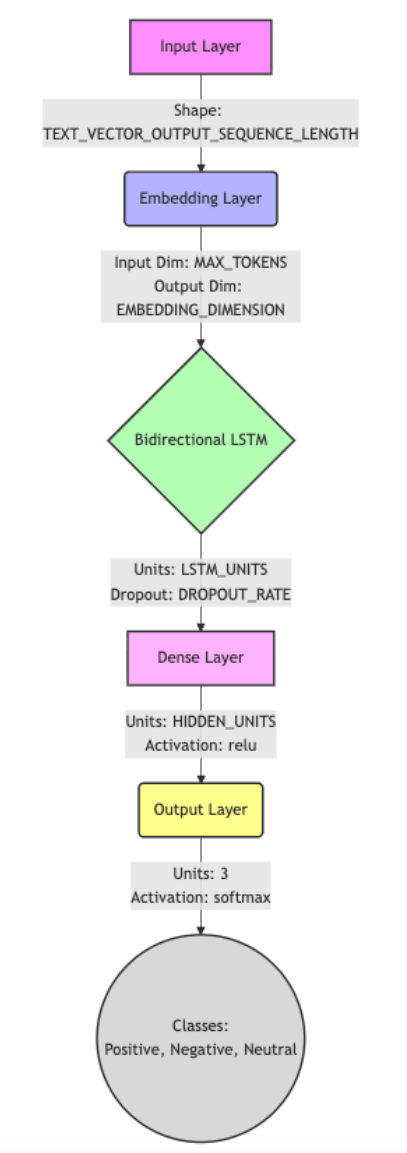
EDA – KEY FINDINGS – WORD CLOUDS



RNN MODEL BUILD - METHODOLOGY

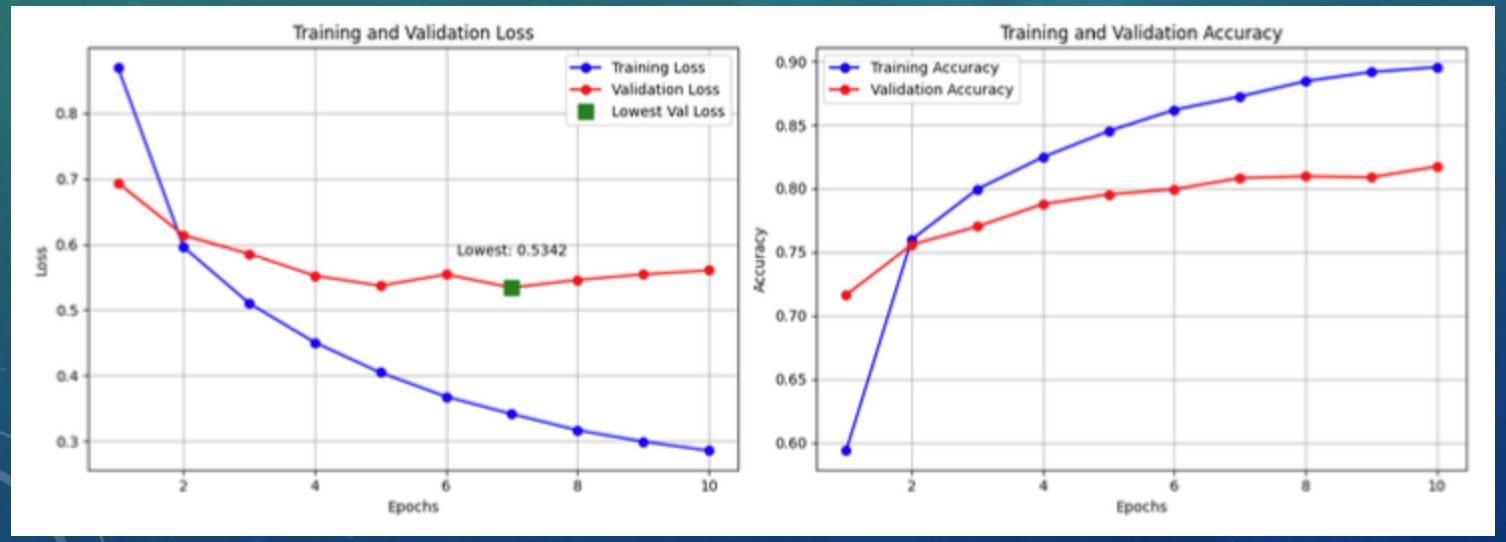
- Deep NN was chosen with multiple layers
- Embedding layer to begin with
- Bi Directional LSTM
- Followed by Dense Relu
- Lastly softmax with 3 classes for prediction probabilities

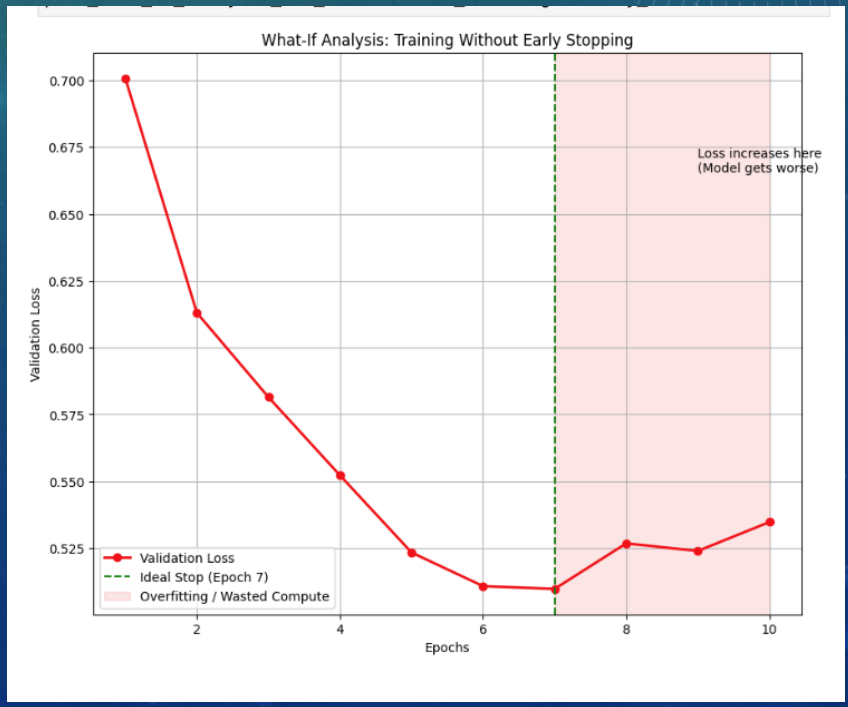
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 64)           320000

 bidirectional (Bidirection  (None, 128)               66048
 al)

 dense (Dense)               (None, 32)                4128

 dense_1 (Dense)             (None, 3)                 99

=================================================================
Total params: 390275 (1.49 MB)
Trainable params: 390275 (1.49 MB)
Non-trainable params: 0 (0.00 Byte)
```



EVAL RESULTS AND PERFORMANCE METRICS

- 10 Epochs were chosen – as per industry standard recommendation for tweet sentiment analysis
- Obtained ~82% Accuracy
- Performed what-if analysis to find out impact of not stopping early





CHALLENGES FACED & MODEL IMPROVEMENT

- Cross validation was performed with 5 folds
- Model performance was consistent with an average validation accuracy of 83.5%
- Grid search with various dropout rates were performed as well
- Our RNN model and hyperparameter choice was proven solid
- External benchmarking was performed with NLTK twitter dataset
- Results were moderate at best, indicating the need to use a global vocabulary for embeddings generation

```
Fold 1 - Validation Accuracy: 82.08%
Fold 2 - Validation Accuracy: 83.73%
Fold 3 - Validation Accuracy: 83.04%
Fold 4 - Validation Accuracy: 83.02%
Fold 5 - Validation Accuracy: 83.37%
Average Validation Accuracy across folds: 83.05% ± 0.55%
```

```
Training model with 64 lstm units, 64 hidden units, and 0.0 dropout...
Test Accuracy: 79.41%
Training model with 64 lstm units, 64 hidden units, and 0.2 dropout...
Test Accuracy: 81.18%
```

```
Downloading NLTK Twitter dataset...
Vectorizing 2000 tweets...
Running predictions...

========================================
 EXTERNAL DATASET BENCHMARK RESULTS
========================================
              precision    recall  f1-score   support

    Negative       0.54      0.33      0.41      1000
     Neutral       0.00      0.00      0.00         0
    Positive       0.52      0.72      0.60      1000

    accuracy                           0.53      2000
   macro avg       0.35      0.35      0.34      2000
weighted avg       0.53      0.53      0.51      2000
```

SAMPLE TWEET DEMONSTRATION

- Model performed very well when tested with sample tweets – accurately predicting all inputs given

```
def manual_test_samples() -> pd.DataFrame:
    sample_tweets = [
        "I absolutely love the new features!",
        "The service was terrible and I am very disappointed.",
        "It was okay, nothing special but not bad either.",
        "Can anyone help me with this issue? I'm stuck."
    ]
    label_mapping = {"Negative": 0, "Neutral": 1, "Positive": 2}
    index_to_label = {v: k for k, v in label_mapping.items()}
    processed_samples = GlobalConfig.VECTORIZER(sample_tweets)
    predictions = model.predict(processed_samples)
    predicted_indices = np.argmax(predictions, axis=1)
    predicted_labels = [index_to_label[idx] for idx in predicted_indices]
    results_df = pd.DataFrame({
        "Tweet": sample_tweets,
        "Prediction": predicted_labels,
        "Confidence": np.max(predictions, axis=1)
    })
```

```
    return results_df

results_df = manual_test_samples()
display(results_df)
```

```
1/1 [==============================] - 0s 21ms/step
```

	Tweet	Prediction	Confidence
0	I absolutely love the new features!	Positive	0.837939
1	The service was terrible and I am very disappo...	Negative	0.963436
2	It was okay, nothing special but not bad either.	Neutral	0.723399
3	Can anyone help me with this issue? I'm stuck.	Negative	0.918900