MIDDLE TENNESSEE STATE UNIVERSITY

Comparing BERT Transformer Results To Naive Bayes & Random Forest Models for Emotion Detection

CSCI 6350

Selected Topics in AI: Natural Language Processing (NLP)

By Richard Hoehn MTSU – April 2024

Welcome

Richard Hoehn

•

Living Franklin

•

MTSU Grad 2005, Vanderbilt Grad 2009

•

PhD Student

Computational & Data Science

•

Web & Mobile Applications, Databases, API / PubSub Integrations, and Product Development

Introduction & Agenda

Introduction

- Explore effectiveness of BERT Transformer for Emotion Detection with small datasets
- Compare to previous work on Naïve Bayes & Random Forest Models
- Visual Plots indicating Training & Validation and comparing Test with past model results
- and Finally compare to current BERT ED work on larger datasets.

Agenda

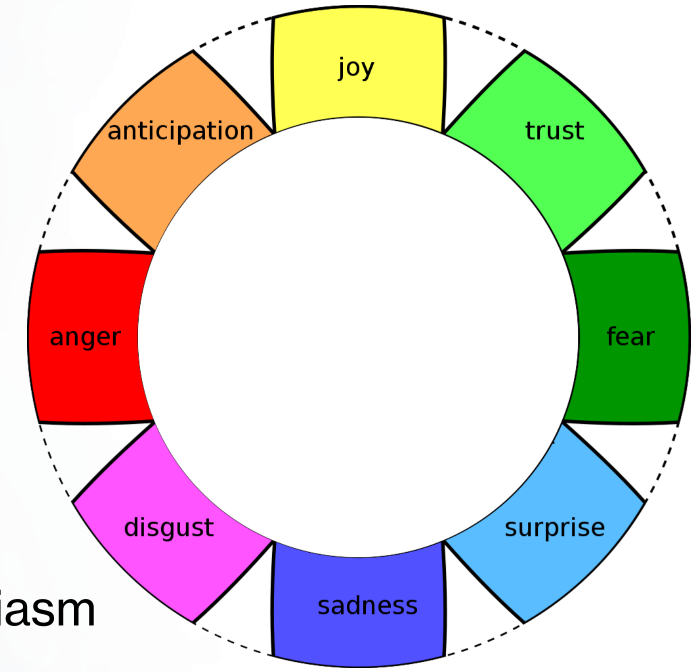
- Significance of Emotion Detection
- Challenges, Motivation and Scope of Research
- Methodology – Including Code Review and Demonstration!
- *and finally,* Analysis of Results, Comparisons, Conclusion, and Future Work

Significance of Emotion Detection

Emotions can vary depending on the framework or model being considered.

One well-known model is the Plutchik's Wheel of Emotions, which proposes eight (8) primary emotions.

- Joy / Happiness
- Sadness
- Anger
- Worry / Fear
- Surprise
- Disgust / Hate
- Trust / Love
- Anticipation / Enthusiasm



<u>Accurately Identifying</u> emotions can improve:

- User Experiences (chat-bots),
- Decision-Making Processes
- and Overall human-machine interactions in a positive manner[4, 5]

With most of these interactions being processed in <u>real-time</u>.

Challenges: Data Scarcity & Language Fragmentation

- Emotion detection data requires primarily supervised learning data! Meaning human curated and labeled data.
- Unlike Sentiment Analysis (SA) the availability of large datasets for training purposes of ML models is much smaller[3].
- Many datasets that are available are in many <u>different languages</u>
- Emotions in text are contextual in nature; meaning they conform in many cases based on the language and culture using them.

Motivation for Research? Evaluation of Models for Prediction?

- ED is still a growing field in Text, Video, and Image reading.
- The market for ED software and services is estimated to reach **$3.8billion**[2] by 2025.
- What model can be used for ED to improve on my past work? Random Forest, Naïve Bayes, **or BERT?**
- A recent paper[1] states that "***Emotions hold a paramount role in the conversation, as it expresses context to the conversation.***", this means that emotions are a part of a conversation and with that are needed to ensure valid analysis of a conversation.

MIDDLE TENNESSEE STATE UNIVERSITY

Objectives & Scope of Research

This Graduate Project's objectives were:

1. **Build, Train, and Validate a Transformer for Emotion Detection, using PyTorch[6].**
2. **Display the learning rates and validation results in graphical forms using plots.**
3. **Evaluate the Transformer's results against those from Hoehn[11] et al.'s paper.**
   In specific the <u>Naive Bayes & Random Forest</u> results.
4. **Compare the transformer with Lee et al.'s transfer learning-based emotion detection model[12] to ensure comparative results in ED have been achieved with the fine-tuned model.**

In summary, this research project **<u>investigated BERT Tranformers</u>** to enhance the predictability of Emotion Detection and compare to previous work by Hoehn[11].

Methodology

1) Data Preparation & Pre-Processing

2) Feature Engineering & Label Encoding

3) Sentence / Word Tokenization

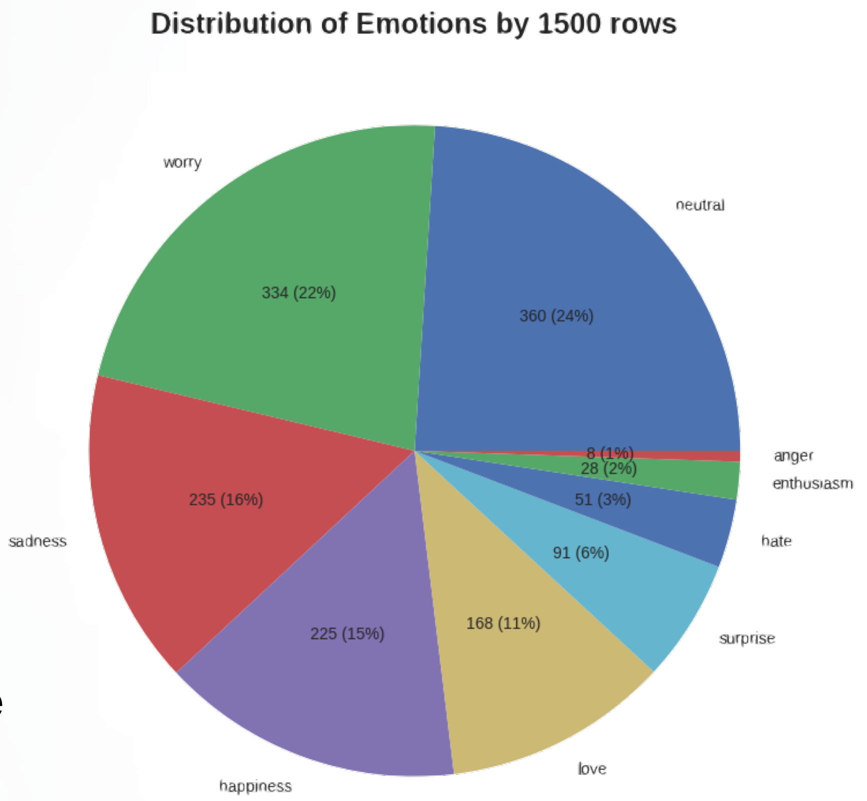
4) Model Training & Validation

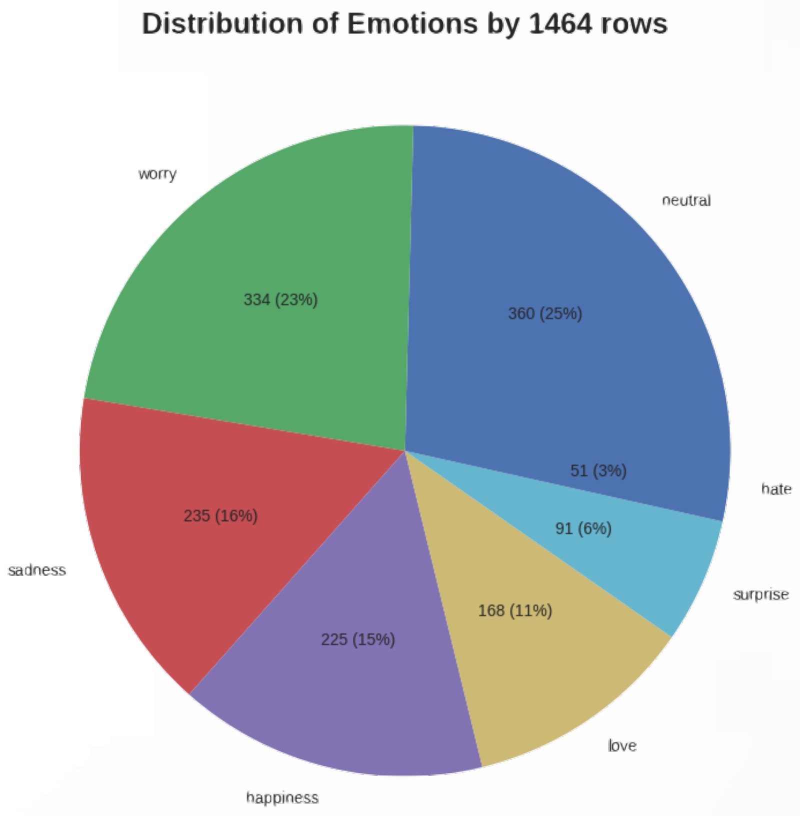
5) Test & Comparison

MIDDLE TENNESSEE STATE UNIVERSITY

Methodology
Data Preparation & Pre-Processing

- Loaded data into Pandas for review
- Visualized the distribution of the emotions.
- Removed:
  - Anger
  - Enthusiasm
- Emotion Count: **7**
- Only **1,464** rows of sentence and labeled emotions





Methodology
Feature Engineering & Label Encoding

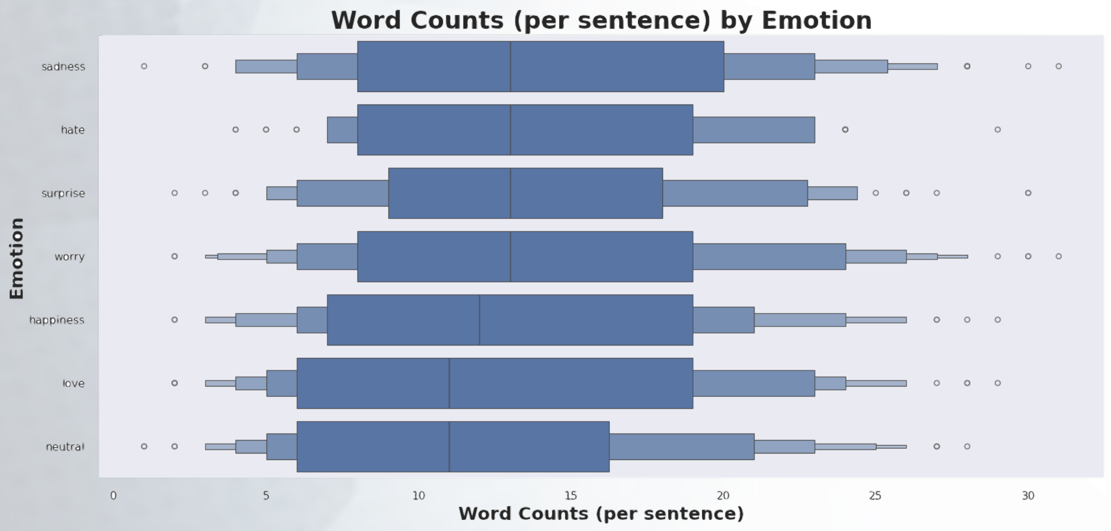
- By use of Label Encoder converted named emotion to numerical values
- Indexes:
  - 0 - 'Happiness'
  - 1 - 'Hate'
  - 2 - 'Love'
  - 3 - 'Neutral'
  - 4 - 'Sadness'
  - 5 - 'Surprise'
  - 6 - 'Worry'

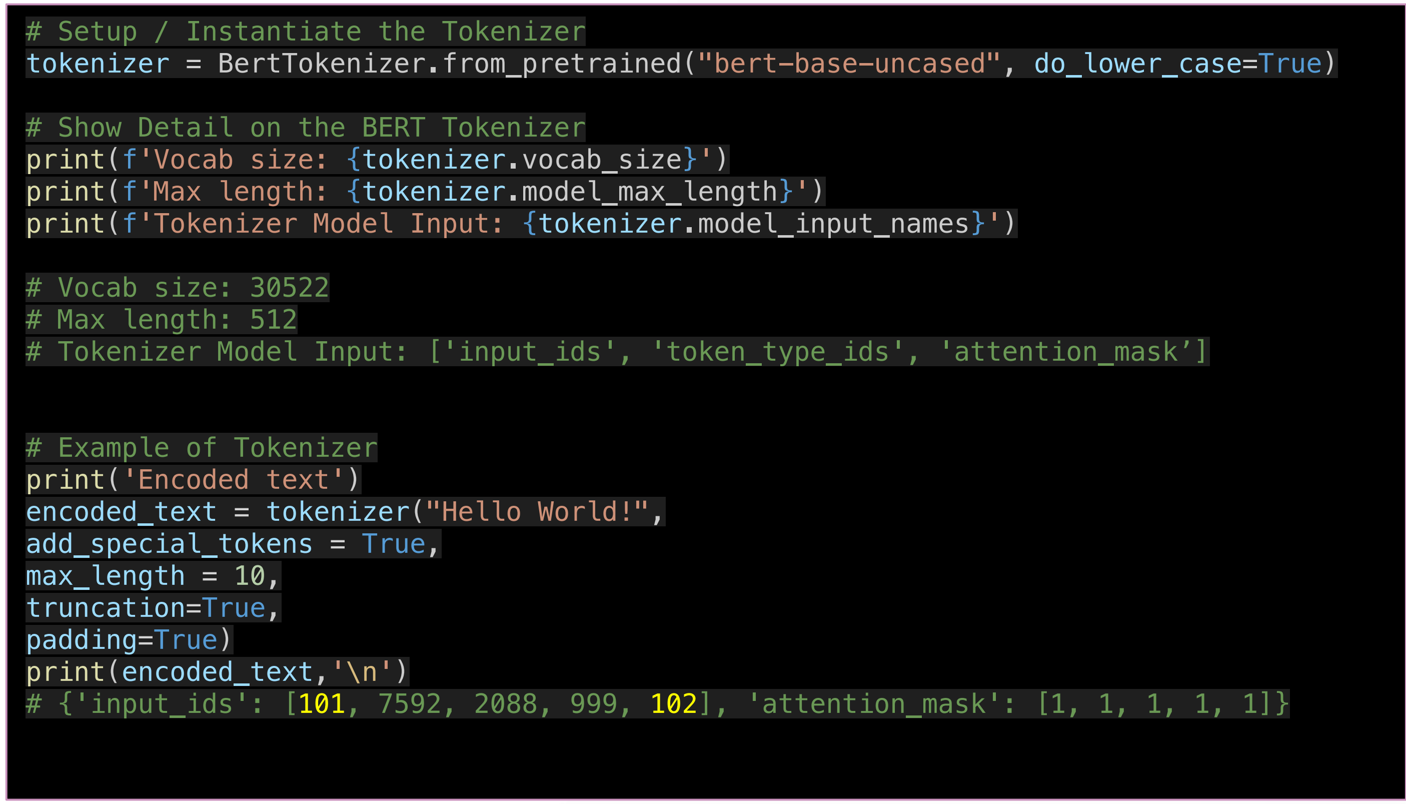
```
# SKlearn
from sklearn.preprocessing import LabelEncoder

# Label Encoding the Emotions
label_encoder = LabelEncoder()
# Encode labels in column 'emotion_en'
df['label'] = label_encoder.fit_transform(df['emotion_en'])

# Show Updated Dataframe
display(df.head(10))
```

Methodology

Sentence / Word Tokenization

- Tokenizer can encode up to 512
- Checked the Longest Sentence / Tokens
- Decided to tokenize on **64 length**
- Shortening the token length generally improves training efficiency by reducing the computational load


Word Counts (per sentence) by Emotion


```
# Setup / Instantiate the Tokenizer
tokenizer = BertTokenizer.from_pretrained("bert-base-uncased", do_lower_case=True)

# Show Detail on the BERT Tokenizer
print(f'Vocab size: {tokenizer.vocab_size}')
print(f'Max length: {tokenizer.model_max_length}')
print(f'Tokenizer Model Input: {tokenizer.model_input_names}')

# Vocab size: 30522
# Max length: 512
# Tokenizer Model Input: ['input_ids', 'token_type_ids', 'attention_mask']


# Example of Tokenizer
print('Encoded text')
encoded_text = tokenizer("Hello World!",
add_special_tokens = True,
max_length = 10,
truncation=True,
padding=True)
print(encoded_text,'\n')
# {'input_ids': [101, 7592, 2088, 999, 102], 'attention_mask': [1, 1, 1, 1, 1]}
```

Methodology

Sentence / Word Tokenization

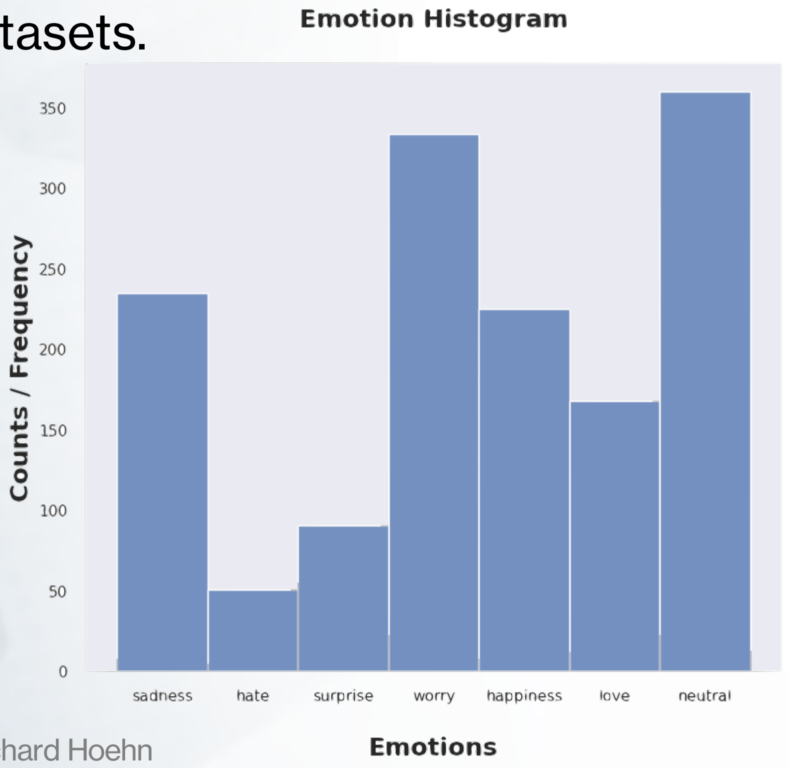
- Example:
  - Sentence that was tokenized at 64 length
  - Attention mask used
  - Start and Stop Tokens 101 & 102

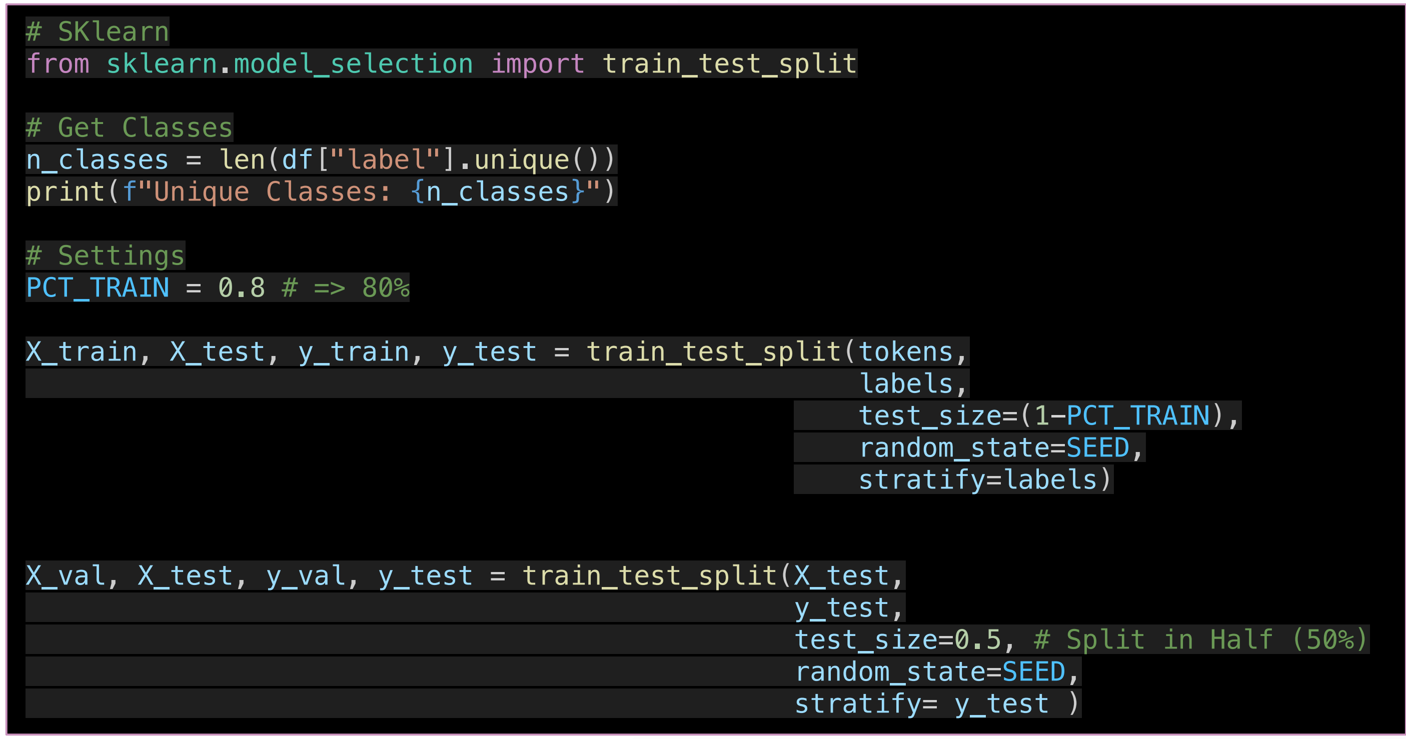
```
Sentence: ->
@Desert_Star95 oh so you know how I feel then Damn representative for bank of america tried to make it sound like I did it.
What a b ...

Tokens: ->
'input_ids':
[101, 1030, 5532, 1035, 2732, 2683, 2629, 2821, 2061, 2017, 2113, 2129, 1045, 2514, 2059, 4365, 4387, 2005, 2924, 1997, 2637,
2699, 2000, 2191, 2009, 2614, 2066, 1045, 2106, 2009, 1012, 2054, 1037, 1038, 1012, 1012, 1012, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
'attention_mask':
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Label: -> 6 Emotion: -> worry
```

MIDDLE TENNESSEE STATE UNIVERSITY

Methodology
Model Training & Validation

- Create Training, Validation, and Testing Datasets 80%, 10% and 10%
- Use **Stratification** for randomization!
  - This maintains the distribution of labels across datasets.



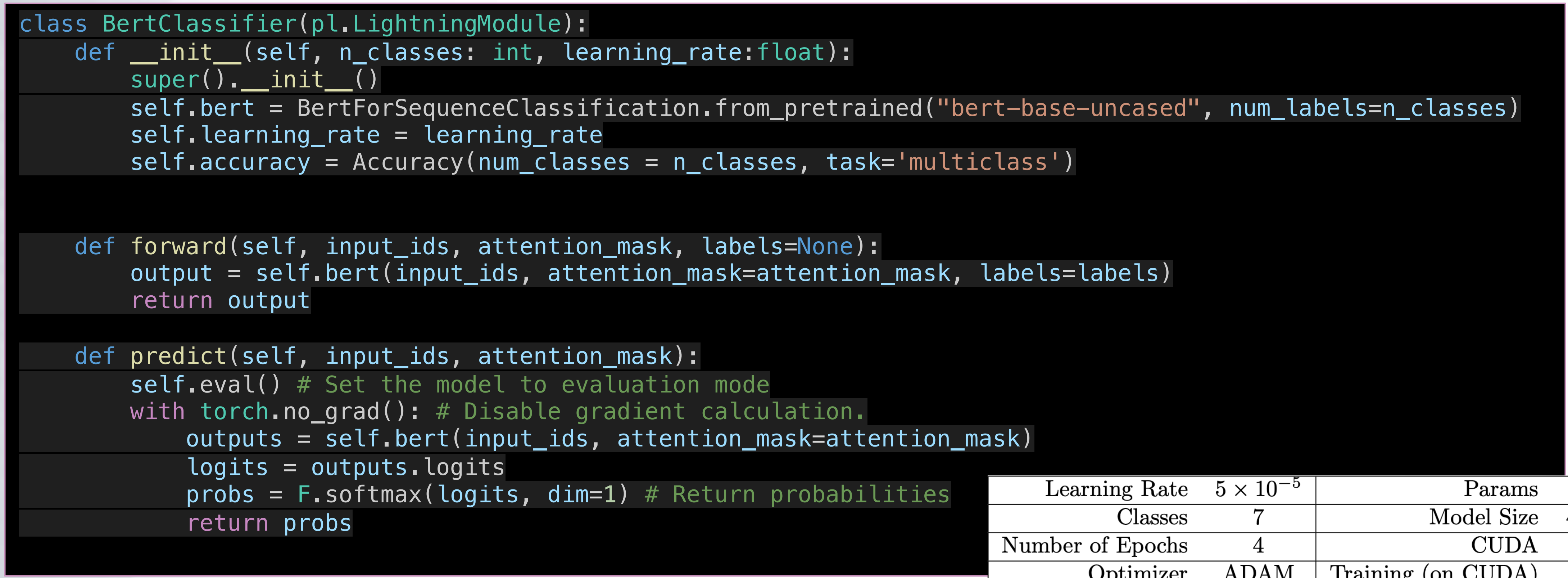


```
# SKlearn
from sklearn.model_selection import train_test_split

# Get Classes
n_classes = len(df["label"].unique())
print(f"Unique Classes: {n_classes}")

# Settings
PCT_TRAIN = 0.8 # => 80%

X_train, X_test, y_train, y_test = train_test_split(tokens,
                                                     labels,
                                                     test_size=(1-PCT_TRAIN),
                                                     random_state=SEED,
                                                     stratify=labels)


X_val, X_test, y_val, y_test = train_test_split(X_test,
                                                y_test,
                                                test_size=0.5, # Split in Half (50%)
                                                random_state=SEED,
                                                stratify= y_test )
```

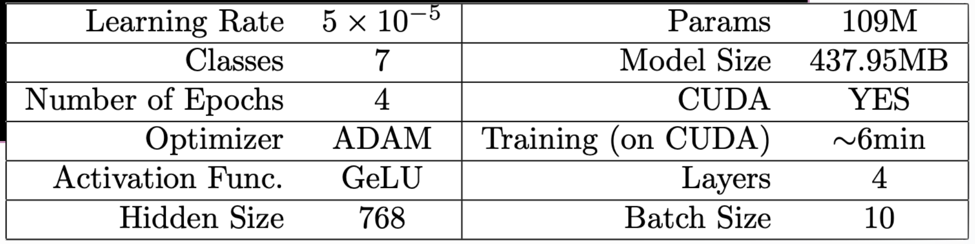
MIDDLE TENNESSEE STATE UNIVERSITY

Methodology
Model Training & Validation

BERT Model – Init, Forward, & Predict



```python
class BertClassifier(pl.LightningModule):
    def __init__(self, n_classes: int, learning_rate:float):
        super().__init__()
        self.bert = BertForSequenceClassification.from_pretrained("bert-base-uncased", num_labels=n_classes)
        self.learning_rate = learning_rate
        self.accuracy = Accuracy(num_classes = n_classes, task='multiclass')


    def forward(self, input_ids, attention_mask, labels=None):
        output = self.bert(input_ids, attention_mask=attention_mask, labels=labels)
        return output

    def predict(self, input_ids, attention_mask):
        self.eval() # Set the model to evaluation mode
        with torch.no_grad(): # Disable gradient calculation.
            outputs = self.bert(input_ids, attention_mask=attention_mask)
            logits = outputs.logits
            probs = F.softmax(logits, dim=1) # Return probabilities
            return probs
```



Learning Rate	5×10^{-5}	Params	109M
Classes	7	Model Size	437.95MB
Number of Epochs	4	CUDA	YES
Optimizer	ADAM	Training (on CUDA)	~6min
Activation Func.	GeLU	Layers	4
Hidden Size	768	Batch Size	10

MIDDLE TENNESSEE STATE UNIVERSITY

Methodology
Model Training & Validation

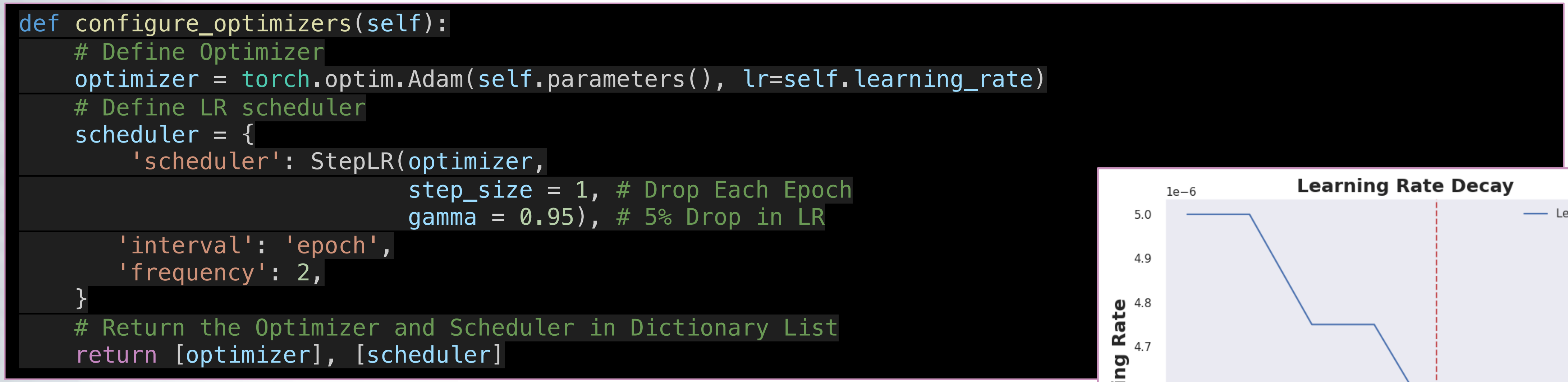
<u>BERT Model – Training Step</u>

```python
def training_step(self, batch, batch_idx):
    input_ids, attention_mask, labels = batch
    output = self.forward(input_ids, attention_mask, labels)
    # Get Loss
    loss = output.loss
    self.log('train_loss', loss, on_step=False, on_epoch=True)

    # Get Acc
    logits = output.logits
    preds = torch.argmax(logits, dim=1)
    acc = self.accuracy(preds, labels)
    self.log('train_acc', acc, on_step=False, on_epoch=True)

    # Get Learning Rate (for Logging)
    lr = self.trainer.optimizers[0].param_groups[0]['lr’]
    self.log('lr', lr, on_step=False, on_epoch=True)

    return loss
```

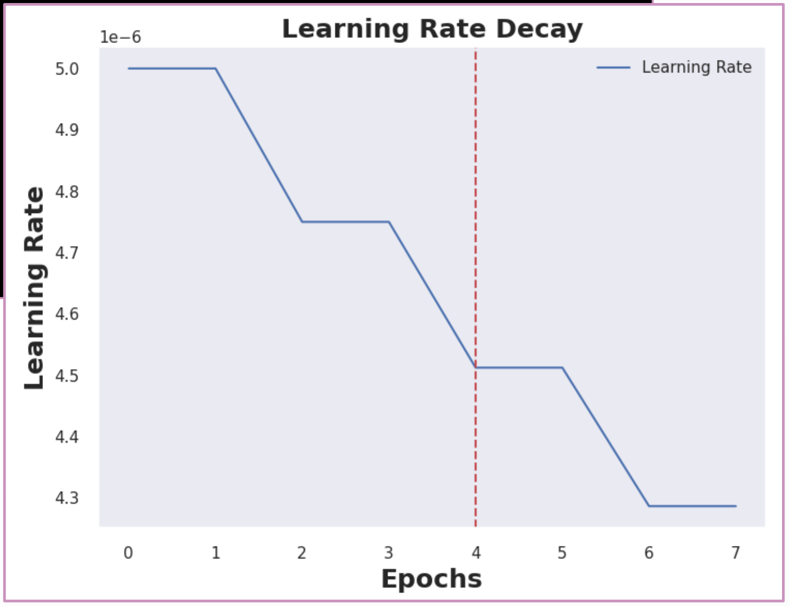
Methodology

Model Training & Validation

BERT Model – Optimizer



```python
def configure_optimizers(self):
    # Define Optimizer
    optimizer = torch.optim.Adam(self.parameters(), lr=self.learning_rate)
    # Define LR scheduler
    scheduler = {
        'scheduler': StepLR(optimizer,
                            step_size = 1, # Drop Each Epoch
                            gamma = 0.95), # 5% Drop in LR
       'interval': 'epoch',
       'frequency': 2,
    }
    # Return the Optimizer and Scheduler in Dictionary List
    return [optimizer], [scheduler]
```

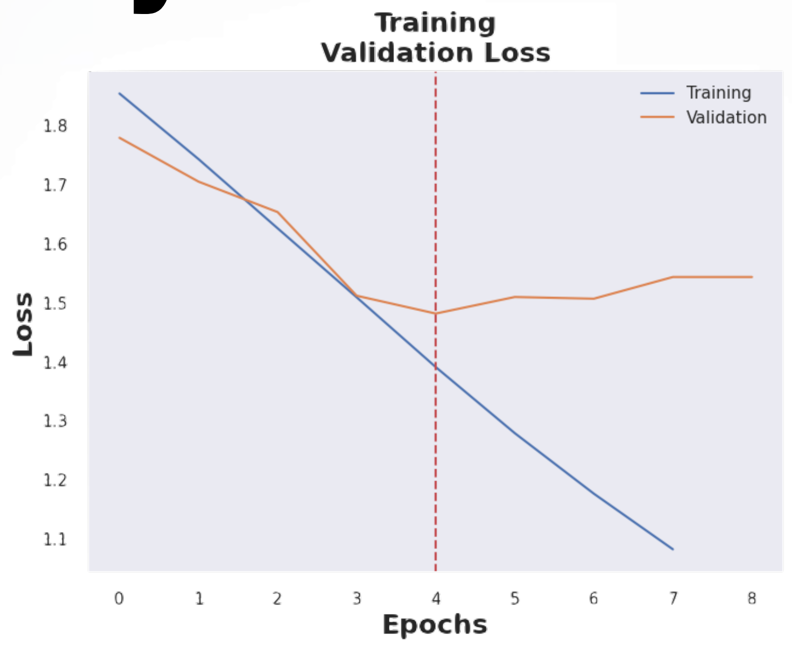


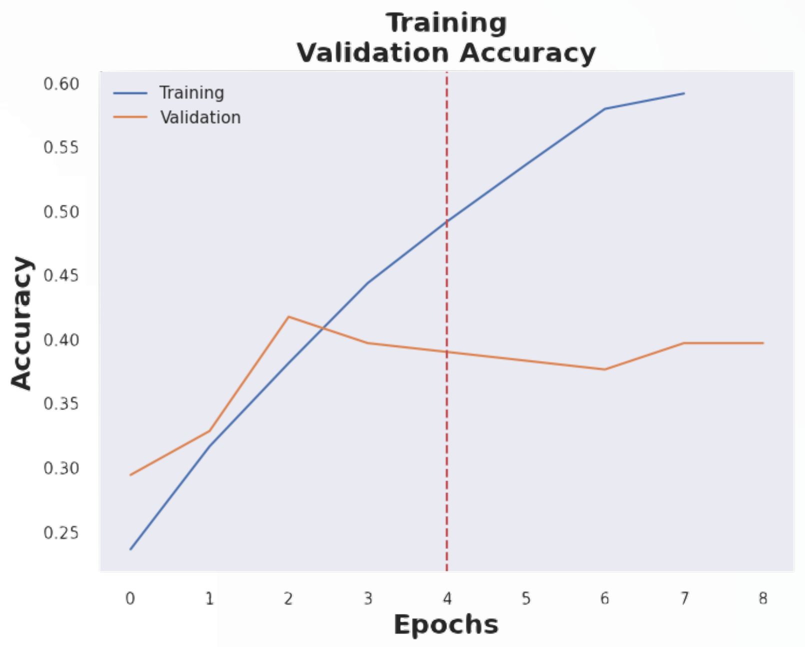
MIDDLE TENNESSEE STATE UNIVERSITY

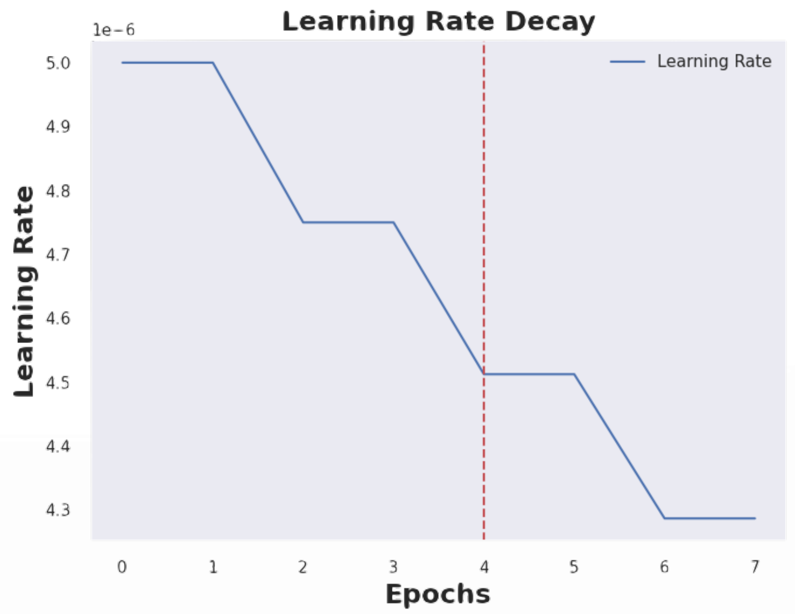
Demo & Code Review

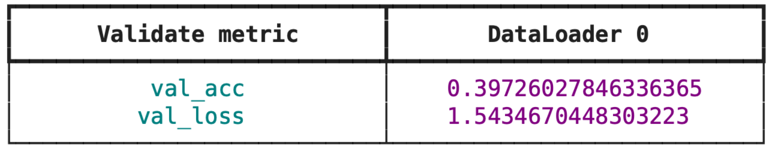
Results & Analysis

- Best convergence achieved with only 4 Epochs
- Early stopping allowed for quicker training time
- Validation Acc: 39%
- Test Acc: 38%
- Model was not – overfitted!

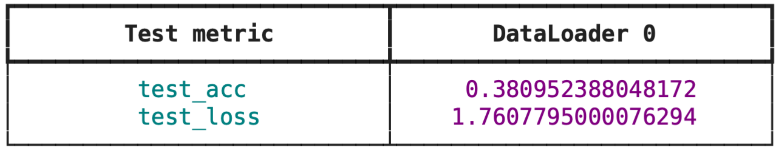








Validate metric	DataLoader 0
val_acc	0.39726027846336365
val_loss	1.5434670448303223



Test metric	DataLoader 0
test_acc	0.380952388048172
test_loss	1.7607795000076294

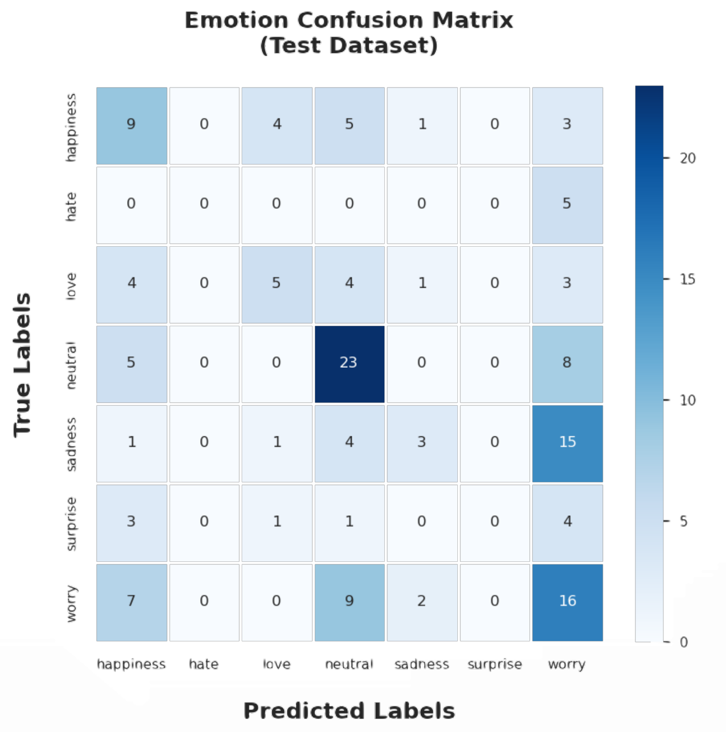
Results & Analysis

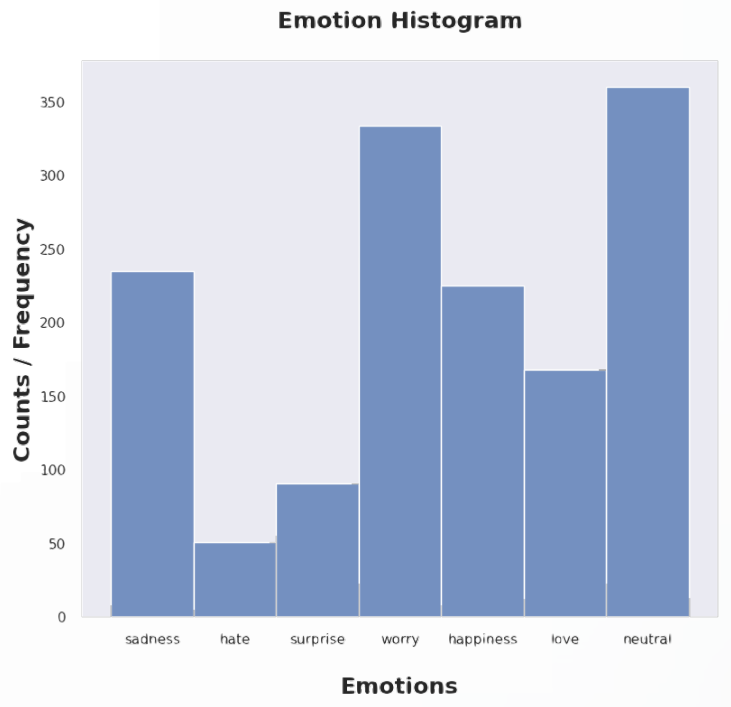
Prediction Results

- Improved over Naïve Bays and Random Forest
- Training Time Improvements
- Confusion Matrix indicated Neutral to good predictor
- Miss Classifications:
  - ’Happiness’ and ’Neutral’
  - ’Love’ and ’Neutral’
  - ’Sadness’ and ’Worry’
- **’Neutral’** and **’Happiness’** are the most prevalent emotions, while **’Hate’** has the least occurrence.

Comparisons - Test Accuracy & Model Training Time

Model	Accuracy	Training Time	Epochs
Naive Bayes	32.08%	~30 min	150
Random Forest	29.72%	~32 min	150
BERT	**38.09%**	**~6 min**	**4**





Conclusions & Future Work

Concluding

The model was verging on over-fitting. This is to be expected since there where only 1,500 sentences to train on.

The prediction rate is still not great compared to BERT models with lots of data! Those achieve up to 76%[12] in accuracy.

BERT outperformed the Naive Bayes and Random Forest models by **over 6 - 10%,** with far fewer epochs and hence less training time required. that it .

Areas of Future Work

- **Regularization Techniques**
  Implementing more sophisticated regularization techniques could help mitigate the overfitting observed in the training process. Techniques such as dropout variations, L2 regularization (weight decay) might help.
- **Exploration of Learning Rate Schedules**
  The learning rate schedule has a big impact on the training dynamics of BERT. Experimenting with different schedules, might lead to better optimization pathways and quicker convergence without falling into local minima or overfitting

Thank you

MIDDLE TENNESSEE STATE UNIVERSITY

Citations

1. Andry Chowanda et al. “Exploring Text-based Emotions Recognition Machine Learning Techniques on Social Media Conversation”. In: Procedia Computer Science 179 (2021). 5th International Conference on Computer Science and Computational Intelligence 2020, pp. 821–828. issn: 1877-0509. doi: https : / / doi . org / 10 . 1016 / j . procs . 2021 . 01 . 099. url: https://www.sciencedirect.com/science/article/pii/S1877050921001320.

2. *Jay Stanley. ”THE DAWN OF ROBOT SURVEILLANCE”. 2019. url: https://www.aclu. org/report/dawn-robot-surveillance*

3. Sajani Ranasinghe et al. “An Artificial Intelligence Framework for the Detection of Emotion Transitions in Telehealth Services”. In: July 2022, pp. 1–5. doi: 10.1109/HSI55341.2022.9869503.

4. Laura Mascarell et al. “Stance Detection in German News Articles”. In: Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER). Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 66–77. doi: 10.18653/v1/2021.fever-1.8. url: https://aclanthology.org/2021.fever-1.8.

5. Valentina Colonnello, Katia Mattarozzi, and Paolo M Russo. “Emotion recognition in medical students: effects of facial appearance and care schema activation”. In: Medical Education 53.2 (2019), pp. 195–205. doi: https://doi.org/10.1111/medu.13760. eprint: https:// onlinelibrary.wiley.com/doi/pdf/10.1111/medu.13760. url: https://onlinelibrary.wiley.com/doi/abs/10.1111/medu.13760.

6. Shalini Kapoor and Tarun Kumar. “Detecting emotion change instant in speech signal using spectral patterns in pitch coherent single frequency filtering spectrogram”. In: Expert Systems with Applications 232 (2023), p. 120882. issn: 0957-4174. doi: https://doi.org/10.1016/j.eswa.2023.120882. url: https://www.sciencedirect.com/science/article/pii/S0957417423013842.

7. Sheetal Kusal et al. “AI Based Emotion Detection for Textual Big Data: Techniques and Contribution”. In: Big Data and Cognitive Computing 5.3 (2021). issn: 2504-2289. doi: 10.3390/bdcc5030043 . url: https://www.mdpi.com/2504-2289/5/3/43.

8. Kaggle. Url: https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text

9. Victor Sanh et al. “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: ArXiv abs/1910.01108 (2019).

10. Xue Ying. “An Overview of Overfitting and its Solutions”. In: Journal of Physics: Conference Series 1168 (Feb. 2019), p. 022022. doi: 10.1088/1742-6596/1168/2/022022.

11. Richard Hoehn and Jaishree Ranganathan. Improving Emotion Detection through Translation of Text to ML Models Trained in Different Languages. Version 1.0.1. https://github.com/richardhoehn/ mtsu.coms.qualifying-exam. Aug. 2023. url: https://github.com/richardhoehn/mtsu.coms. qualifying-exam.

12. Sanghyub John Lee et al. “Transformer transfer learning emotion detection model: synchronizing socially agreed and self-reported emotions in big data”. In: Neural Computing and Applications 35.15 (2023), pp. 10945–10956. doi: 10.1007/s00521-023-08276-8. url: https://doi.org/10.1007/s00521- 023-08276-8.