EMOTION DETECTION IN TEXTUAL DATA

By: Peter Charles Bailey (Charlie) | peba2926

CSPB 4830 – Natural Language Processing

PROBLEM OVERVIEW

PROBLEM OVERVIEW

TASK

- Classify text into 6 emotions:
  - Sadness
  - Joy
  - Love
  - Anger
  - Fear
  - Surprise

WHY IT'S INTERESTING

- Written text lacks the full emotional context of spoken language.
- Improving detection accuracy can improve communication and reduce misunderstanding.

MOTIVATION & GOALS

MOTIVATION

- Improve online communication and reduce polarization.

GOAL

- Fine-tune a BERT-based classifier to detect emotion in text.

RESEARCH QUESTION

- Can a fine-tuned BERT architecture accurately capture the subtle human emotional cues in text?

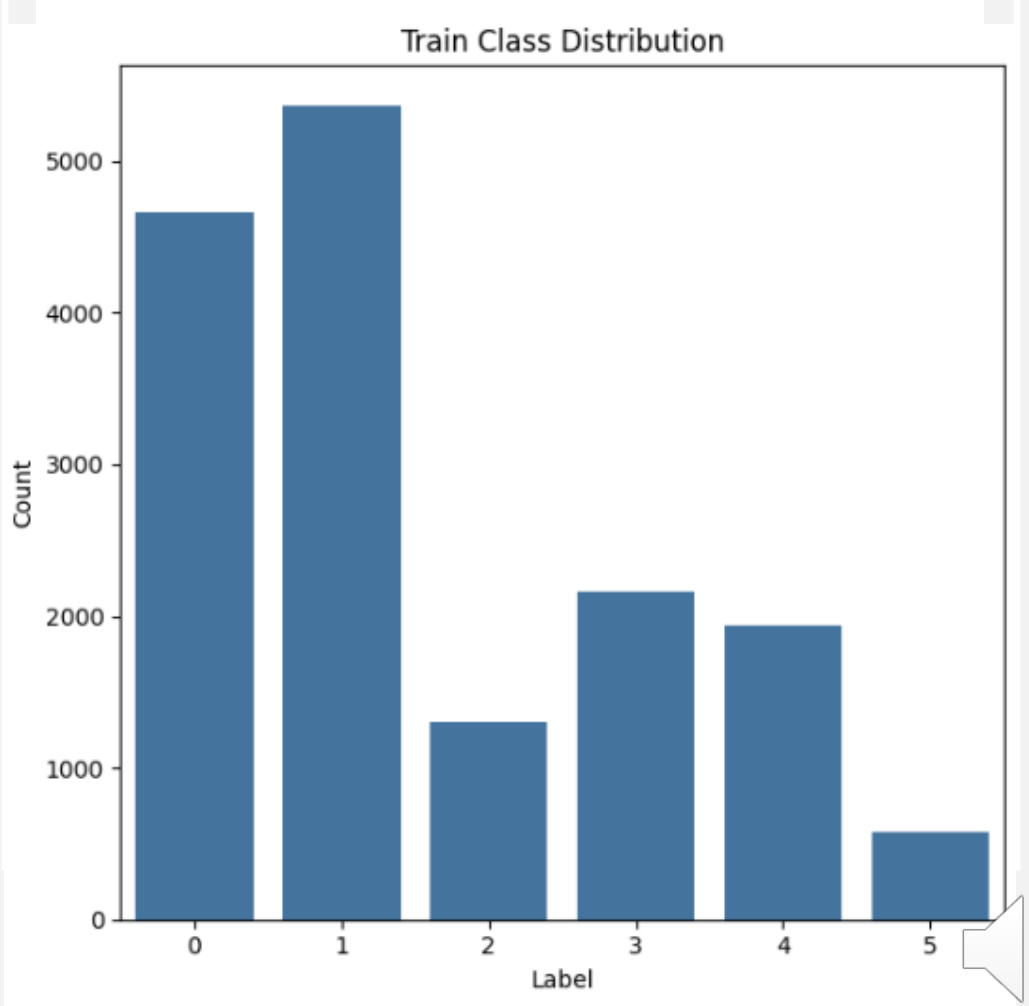
THE DATASET

	text	label
0	i didnt feel humiliated	0
1	i can go from feeling so hopeless to so damned...	0
2	im grabbing a minute to post i feel greedy wrong	3
3	i am ever feeling nostalgic about the fireplac...	2
4	i am feeling grouchy	3
5	ive been feeling a little burdened lately wasn...	0
6	ive been taking or milligrams or times recomme...	5
7	i feel as confused about life as a teenager or...	4
8	i have been with petronas for years i feel tha...	1
9	i feel romantic too	2

THE DATASET

- Dair-ai/emotion from Hugging Face
- Split version with:
  - 16k train instances
  - 2k validation instances
  - 2k test instances
- Features:
  - 0: text string
  - 1: numeric label with sadness (0), joy (1), love (2), anger (3), fear (4), surprise (5)

Name	Train	Validation	Test
split	16,000	2,000	2,000
unsplit	416,809	N/A	N/A



APPROACH OVERVIEW

PLAN

- 1: Exploratory Data Analysis
- 2: Fine-tune the "*bert-base-uncased*" model
- 3: Perform hyperparameter tuning
- 4: Scale up

RESULTS

- 1: Exploratory Data Analysis
- 2: Fine-tune the "bert-base-uncased" model
- 3: Perform hyperparameter tuning
- 4: Scale up

TOOLS AND TECHNIQUES

```
import pandas as pd
import torch
import numpy as np
from tqdm import tqdm
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from datasets import load_dataset
import evaluate
from transformers import (BertTokenizer,
                          BertForSequenceClassification,
                          DataCollatorWithPadding,
                          TrainingArguments,
                          Trainer,
                          )
```

THE EXPERIMENT: IMPLEMENTATION STEPS

1. Data loading and tokenization utilizing Hugging Face datasets and BertTokenizer
2. Model setup utilizing BertForSequenceClassification
3. Training with Trainer class using default hyperparameters
4. Evaluation with weighted F1 and Accuracy scores

```python
class EmotionDetection():
    def __init__(self):
        """
        Load data. Initialize tokenizer and model.
        """
        self.dataset = load_dataset('dair-ai/emotion', 'split')
        self.checkpoint = "bert-base-uncased"
        self.tokenizer = BertTokenizer.from_pretrained(self.checkpoint)
        self.data_collator = DataCollatorWithPadding(tokenizer=self.tokenizer)
        self.tokenized_datasets = self.dataset.map(
            self.tokenize_function,
            batched=True
        )
        self.model = BertForSequenceClassification.from_pretrained(
            self.checkpoint,
            num_labels = 6,
        )
        self.trainer = None
        self.train = self.tokenized_datasets['train']
        self.validation = self.tokenized_datasets['validation']
        self.test = self.tokenized_datasets['test']

    def tokenize_function(self, data):
        """
        Tokenize the input data.
        """
        return self.tokenizer(
            data['text'],
            truncation=True, # cutoff text longer than max_length
        )
```

RESULTS
ACCURACY: 0.93 WEIGHTED F1: 0.93

```
100%|████████████████████████████████████████| 6000/6000 [10:15<00:00,  9.75it/s]
100%|████████████████████████████████████████| 250/250 [00:04<00:00, 58.98it/s]
validation results::: {'eval_loss': 0.2083398401737213, 'eval_accuracy': 0.939, 'eval_f1':
 0.9385439274360571, 'eval_runtime': 4.2538, 'eval_samples_per_second': 470.172, 'eval_ste
ps_per_second': 58.771, 'epoch': 3.0}
100%|████████████████████████████████████████| 250/250 [00:04<00:00, 58.47it/s]
test results::: {'eval_loss': 0.22365400195121765, 'eval_accuracy': 0.931, 'eval_f1': 0.92
96616843630424, 'eval_runtime': 4.2901, 'eval_samples_per_second': 466.19, 'eval_steps_per
_second': 58.274, 'epoch': 3.0}
100%|████████████████████████████████████████| 250/250 [00:04<00:00, 59.02it/s]
test score:::
{'accuracy': 0.931, 'f1': 0.9296616843630424}
```

CHALLENGES AND LESSONS LEARNED

CHALLENGES

- Minor class imbalance
- Getting familiar with Hugging Face ecosystem
- Setting appropriate project expectations

LESSONS

- Weighted F1 can be used to evaluate class imbalances
- Hugging Faces is incredible but takes time to learn
- Multiply by 3 when forecasting project time!

CONCLUSION

FINAL RESULTS

- A fine-tuned BERT based classifier achieved ~93% accuracy in detecting 6 core human emotions in text data

FUTURE DIRECTIONS

- Test for improvements with hyperparameter tuning
- Implement class weighting for improved training
- Potentially integrate into a user facing application

THANK YOU FOR WATCHING!