SFWRENG 4NL3 Assignment 4: Pretrained Transformers

Sumanya Gulati

1 April 2025

Contents

1 Dataset **2**
- 1.1 Data Collection . . . 2
- 1.2 Dataset Structure . . . 2
- 1.3 Evaluation Metrics . . . 3
- 1.4 Data Splits . . . 4
- 1.5 Additional Features . . . 5

2 Fine-tuned Models **5**
- 2.1 Model 1: RoBERTa-base . . . 5
  - 2.1.1 Pretraining Dataset . . . 5
  - 2.1.2 Compute Requirements for Pretraining . . . 6
- 2.2 Model 2: DistilBERT-base-uncased . . . 6
  - 2.2.1 Pretraining Dataset . . . 6
  - 2.2.2 Compute Requirements for Pretraining . . . 6
- 2.3 Fine-Tuning Steps . . . 6
- 2.4 Results . . . 7
  - 2.4.1 Model 1: RoBERTa-base . . . 7
  - 2.4.2 Model 2: DistilBERT-base-uncased . . . 8

3 Zero-shot Classification **8**
- 3.1 Models Description . . . 8
  - 3.1.1 AquilaChat2 34B - GGUF . . . 8
  - 3.1.2 Deepseek LLM 7B Chat - GGUF . . . 8
- 3.2 Model Prompting . . . 9
  - 3.2.1 Prompt 1 . . . 9
  - 3.2.2 Prompt 2 . . . 9
  - 3.2.3 Prompt 3 . . . 9
  - 3.2.4 Prompt 4 . . . 10
  - 3.2.5 Prompt 5 . . . 10
- 3.3 Results . . . 10

4 Reflection **10**

5 Use of Generative AI **11**
- 5.1 How it was used . . . 11
- 5.2 Associated Carbon Footprint . . . 12

List of Tables

1 Label Mapping . . . 3
2 Data Split Overview . . . 4
3 Zero-shot Classification Results . . . 10

List of Figures

1 Class Distribution in Training and Test Splits . . . 4
2 Class Distribution in Training and Test Splits . . . 5

1 Dataset

For this assignment, I am using the GoEmotions dataset extracted from the HuggingFace can be accessed here. The task at hand involves emotion classification, aiming to identify and categorize the emotions expressed in textual data. This is essential for applications such as sentiment analysis, mental health assessment, enhancing human-computer interactions and more.

The GoEmotions dataset comprises of about 58,000 English Reddit comments, each annotated for 27 distinct emotion categories or marked as neutral. The simplified version of the dataset with predefined train, val and test splits has been used for this assignment.

1.1 Data Collection

The dataset has been constructed by selecting English-language comments from Reddit by researchers at Amazon Alexa, Google Research and Stanford Linguistics. A complete list of authors can be found here. The comments were extracted from Reddit using a variety of automated methods along with data curation techniques such as reducing profanity, length filtering, sentiment and emotion balancing, masking and more. Further information about the data collection process can be found in section 3.1 of this paper.

1.2 Dataset Structure

Each instance of the dataset corresponds to a reddit comment with an ID and one or more emotion annotations including neutral. The simplified configuration of the dataset which has been used for this assignment, includes:

- `text`: the Reddit comment
- `labels`: the emotional annotations
- `comment_id`: a unique identifier for the comment

The input for the task is a Reddit comment in English and the corresponding output is a set of one or more labels corresponding to the 27 emotion categories or neutral, reflecting the emotional content of the comment.

The labels are stored as a list of integers ranging from 0 to 27 where each integer represents an emotion category or neutral. The label mapping is as follows:

Label Number	Label Category
0	admiration
1	amusement
2	anger
3	annoyance
4	approval
5	caring
6	confusion
7	curiosity
8	desire
9	disappointment
10	disapproval
11	disgust
12	embarrassment
13	excitement
14	fear
15	gratitude
16	grief
17	joy
18	love
19	nervousness
20	optimism
21	pride
22	realization
23	relief
24	remorse
25	sadness
26	surprise
27	neutral

Table 1: Label Mapping

1.3 Evaluation Metrics

The following evaluation metrics have been used to assess the performance of the BERT-based model:

- Model Performance:
  - Emotion-level Precision, Recall, F1: Measured per each emotion in the GoEmotions taxonomy.
  - Transfer Learning: F1 score on data transferred between domain X and GoEmotions.
- Decision thresholds: No thresholds are used. The data is presented in full granularity.
- Uncertainty and variability: Repeated experiments have yielded results with similar taxonomical rankings.

The model has been evaluated on 10 publicly available datasets including 9 benchmark datasets presented in compilation by Bostan and Klinger and the GoEmotions eval set. Full details about the evaludation results can be found in the paper.

To summarize, based on the information derived from the dataset, the following evaluation metrics will be used for this assignment:

1. **Loss**: The loss value on the evaluation dataset.
2. **F1 Score**: The model's accuracy on a scale of 0 to 1 with 1 being perfect accuracy.

3. **Runtime**: Time taken to complete the evaluation.
4. **Samples per Second**: The processing speed of the model, as in, how many data samples are evaluated by the model per second.
5. **Steps per Second**: How many evaluation steps or batches were processed per second.
6. **Epoch**: A measure of how far the training process of the evaluation was taken. One epoch refers to one complete pass through the training dataset.

1.4 Data Splits

The dataset is divided into training, validating and test splits as follows:

Data Split	Number of Instances
Training	43,410
Validation	5,426
Test	5,427

Table 2: Data Split Overview

Class distributions for the training and test splits are shown in the figure 1.4.

	Emotion	Train Count	Test Count
0	admiration	4130	504
1	amusement	2328	264
2	anger	1567	198
3	annoyance	2470	320
4	approval	2939	351
5	caring	1087	135
6	confusion	1368	153
7	curiosity	2191	284
8	desire	641	83
9	disappointment	1269	151
10	disapproval	2022	267
11	disgust	793	123
12	embarrassment	303	37
13	excitement	853	103
14	fear	596	78
15	gratitude	2662	352
16	grief	77	6
17	joy	1452	161
18	love	2086	238
19	nervousness	164	23
20	optimism	1581	186
21	pride	111	16
22	realization	1110	145
23	relief	153	11
24	remorse	545	56
25	sadness	1326	156
26	surprise	1060	141
27	neutral	14219	1787

Figure 1: Class Distribution in Training and Test Splits

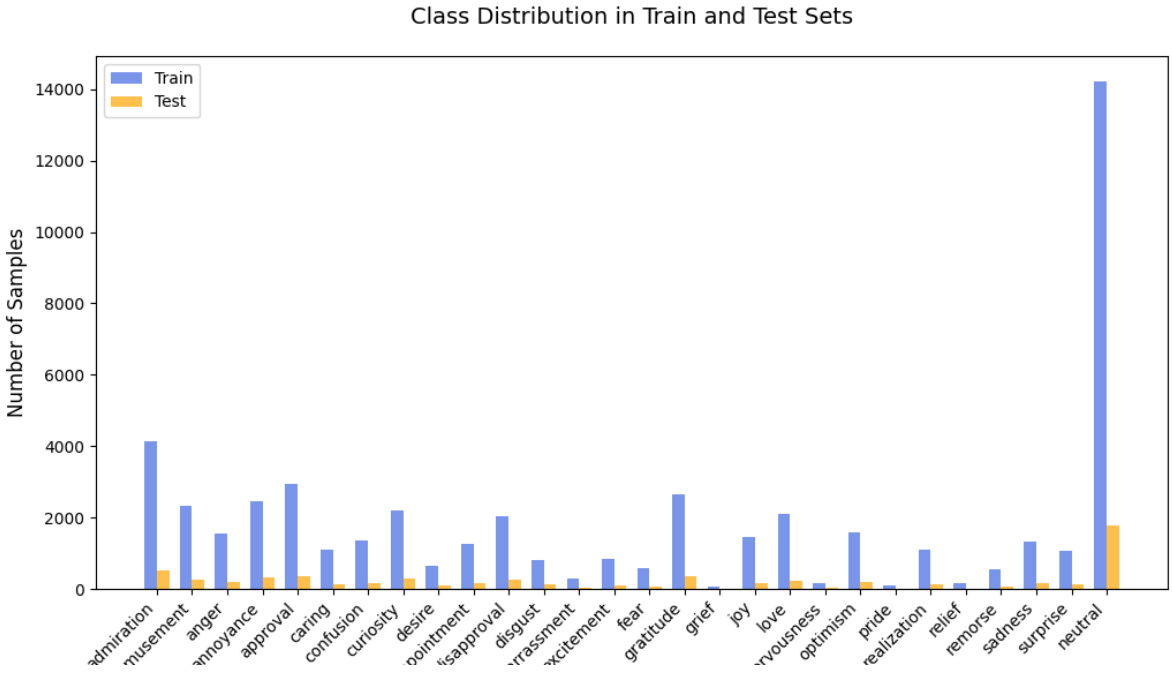

Figure 2: Class Distribution in Training and Test Splits

1.5 Additional Features

Although the comments vary in length, the maximum sequence length has been capped at 30 tokens in the training and evaludation datasets to ensure concise and focused emotional expressions. Furthermore, linguistic context consistency has been ensured by only choosing comments that are in English.

2 Fine-tuned Models

For this task, the RoBERTa-base model and the DistilBERT-base-uncased model have been used.

2.1 Model 1: RoBERTa-base

RoBERTa or "A Robustly Optimized BERT Pretraining Approach" is a transformer-based model that builds on the BERT architecture. It improves upon BERT's pretraining methodology by training longer with larger batches over more data, removing the next sentence prediction objective, and dynamically changing the masking pattern applied to the training data. The model has approximately 125 million total parameters.

2.1.1 Pretraining Dataset

RoBERTa was pretrained on a diverse and extensive corpus totalling around 160GB of text, including:

- BookCorpus which is a dataset of 11,038 unpublished books
- English Wikipedia excluding lists, tables and headers
- CommonCrawl News with over 63 million English news articles from 2016-2019
- OpenWebText which is an open source recreation the WebText dataset used to train GPT-2

- Stories which is a dataset containing a subset of CommonCrawl data filtered to match story-like style of Winograd schemas

2.1.2 Compute Requirements for Pretraining

The pretraining of RoBERTa-base involved substantial computational resources, including:

- **Hardware**: 1,024 V100 GPUs
- **Training Duration**: 500,000 steps
- **Batch Size**: 8,000
- **Sequence Length**: 512 tokens
- **Optimizer**: Adam with a learning rate of 6e-4

2.2 Model 2: DistilBERT-base-uncased

DistilBERT is a distilled version of BERT, designed to be smaller, faster, and more efficient while retaining 97% of BERT's language understanding capabilities. It achieves this through knowledge distillation during the pretraining phase, effectively reducing the model size by 40%. The model has approximately 66 million total parameters.

2.2.1 Pretraining Dataset

DistilBERT was pretrained on the same corpus as BERT, which includes:

- BookCorpus which is a dataset of 11,038 unpublished books
- English Wikipedia excluding lists, tables and headers

2.2.2 Compute Requirements for Pretraining

The pretraining of DistilBERT-base-uncased utilized:

- **Hardware**: 8 16GB V100 GPUs
- **Training Duration**: 90 hours

2.3 Fine-Tuning Steps

The RoBERT-a and DistilBERT-base-uncased models were fine-tuned on the GoEmotions dataset using the following process:

1. Dataset Preparation
    - Loaded the GoEmotions dataset with simplified labels (27 categories in total).
    - Used the predefined train, validation and test splits.
    - Processed multi-label classification format where each text can have multiple emotions.
2. Preprocessing
    - Tokenized the data using RoBERTa's byte-level Byte-Pair Encoding (BPE) tokenizer and DistilBERT's WordPiece tokenizer.
    - Applied truncation and padding to a maximum sequence length of 40 tokens.
    - Converted labels to multi-hot encoded vectors where each of 27 emotion categories is represented as a 0 or 1.

- Saved preprocessed datasets to disk to avoid redundant processing in future runs.

3. Hyperparameter Optimization
   - Used Optuna framework to optimize for weighted F1 score on the validation set.
   - Tuned parameters such as learning rate (1e-5 to 5e-5), weight decay (0.01 to 0.1) and batch size (4 or 8).
   - Performed 2 trials per model to balance optimization quality with computational resources.
   - Implemented pruning with MedianPruner to terminate underperforming trials early.
4. Model Configuration
   - Configured both models with classification heads for multi-label classification.
   - Implemented early stopping with a patience of 2 epochs during optimization and 3 epochs for final training.
   - Applied gradient checkpointing to reduce memory storage.
   - Used gradient accumulation (steps=2 for trials and steps=4 for final models) to stimulate larger batch sizes.
5. Fine-tuning Process
   - Trained both models using a custom MultiLabelTrainer extending HuggingFace's Trainer API.
   - Applied BCEWithLogitsLoss for multi-label classification.
   - Applied the best hyperparameters found during optimization for final model training.
   - Trained for up to 5 epochs with early stopping.
   - Envaluated performance using weighted F1 score on validation set during training.
   - Saved the best checkpoint based on validation performance.
   - Performed final evaluation on the test set and saved results in JSON format.

2.4 Results

Based on the evaluation metrics outlined above, the results of each model are as reported below.

2.4.1 Model 1: RoBERTa-base

The best parameters based on hyperparameter optimization were found to be:

- `learning_rate`: 2e-05
- `weight_decay`: 0.05
- `batch_size`: 8

The evaluation test results are:

- `eval_loss`: 0.0861
- `eval_f1`: 0.551
- `eval_runtime`: 43.1728
- `eval_samples_per_second`: 125.688
- `eval_steps_per_second`: 7.874
- `epoch`: 4.867

To summarize the results, the model has reasonable accuracy and based on the F1 score, is correctly predicting answers about 55% of the time. It also has significantly lower loss, possibly due to nearly 5 complete epochs.

2.4.2 Model 2: DistilBERT-base-uncased

The best parameters based on hyperparameter optimization were found to be:

- `learning_rate`: 1.799e-05
- `weight_decay`: 0.071
- `batch_size`: 4

The evaluation test results are:

- `eval_loss`: 0.494
- `eval_f1`: 0.0031
- `eval_runtime`: 32.7287
- `eval_samples_per_second`: 165.818
- `eval_steps_per_second`: 20.746
- `epoch`: 0.295

To summarize the results, a moderate loss value suggests that the model is learning but has room for improvement. With an extremely poor F1 score of just 0.0031, it is observed that the model has very poor performance and is barely predicting any correct answers. A low epoch score suggests that the evaluation was done when about 29% of the first epoch was complete.

3 Zero-shot Classification

Models from Llama.cpp were used for this portion of the assignment.

3.1 Models Description

Detailed descriptions of each model can be found below.

3.1.1 AquilaChat2 34B - GGUF

Originally developed by the Beijing Academy og Artificial Intelligence, the Q4_K_M quant method of the model, has a size of approximately 20.33 GB. The model is medium-sized and of balanced quantity. While exact data about its pretraining is not publicly available, it is known that this model was built upon an earlier model and fine-tuned on a dataset comprising of roughly 200,000 high-quality, long-text conversations. For comparison, pretraining a 34B parameter model can involve anywhere from tens of thousands to over a hundred thousand GPU hours.

3.1.2 Deepseek LLM 7B Chat - GGUF

Originally developed by DeepSeek, this model, specifically the Q4_K_M quantization, is of size 4.22 GB. This version is the 7 billion parameter variant and is specifically fine-tuned for conversational tasks. Exact information about its pretraining data or computer required for pretraining has not been publicly disclosed but it is known that the model is built upon the Llama architecture and is designed to handle both English and Chinese languages. For comparison, training a 7B parameter model such as the Llama 2 reportedly took roughly 184,320 GPU hours.

3.2 Model Prompting

Each model was prompted with with 5 distinct templates designed to guide it into classifying emotions in the input text based on predefined emotion indices, similar to the GoEmotions dataset. Each template had a different style:

- **Prompt 1**: Simply asks the model to classify the text by emotion indices, listing the indices and their corresponding labels.
- **Prompt 2**: Provides example texts and asks the model to classify a new text based on the same emotion indices.
- **Prompt 3**: Directly asks model to select all applicable emotion indices for the given text, again providing the emotional index mapping.
- **Prompt 4**: Frames the task as a content moderation task where the model is asked to tag emotions from the text.
- **Prompt 5**: Instructs the model to analyze the emotional tone of the text step-by-step and return the emotion indices.

The exact text used for each prompt is as follows:

3.2.1 Prompt 1

> You are an emotion classifier. Analyze the text and respond ONLY with the corresponding emotion numbers (0-27) between square brackets.
>
> Available emotions: 0=admiration 1=amusement 2=anger 3=annoyance 4=approval 5=caring 6=confusion 7=curiosity 8=desire 9=disappointment 10=disapproval 11=disgust 12=embarrassment 13=excitement 14=fear 15=gratitude 16=grief 17=joy 18=love 19=nervousness 20=optimism 21=pride 22=realization 23=relief 24=remorse 25=sadness 26=surprise 27=neutral
>
> Text: "text"

3.2.2 Prompt 2

> Examples: 1. "I love you" → [18] 2. "This is terrible" → [2, 10] Now classify: Text: "text". Analyze the text and respond ONLY with the corresponding emotion numbers (0-27) between square brackets. Available emotions: 0=admiration 1=amusement 2=anger 3=annoyance 4=approval 5=caring 6=confusion 7=curiosity 8=desire 9=disappointment 10=disapproval 11=disgust 12=embarrassment 13=excitement 14=fear 15=gratitude 16=grief 17=joy 18=love 19=nervousness 20=optimism 21=pride 22=realization 23=relief 24=remorse 25=sadness 26=surprise 27=neutral.

3.2.3 Prompt 3

> Select all applicable emotion numbers (0-27) for this text and respond ONLY with the corresponding emotion numbers (0-27) between square brackets. Text: "text".
>
> Available emotions: 0=admiration 1=amusement 2=anger 3=annoyance 4=approval 5=caring 6=confusion 7=curiosity 8=desire 9=disappointment 10=disapproval 11=disgust 12=embarrassment 13=excitement 14=fear 15=gratitude 16=grief 17=joy 18=love 19=nervousness 20=optimism 21=pride 22=realization 23=relief 24=remorse 25=sadness 26=surprise 27=neutral.

3.2.4 Prompt 4

> As a content moderator, Analyze the text and respond ONLY with the corresponding emotion numbers (0-27) between square brackets by tagging emotions (0-27) where 0=admiration 1=amusement 2=anger 3=annoyance 4=approval 5=caring 6=confusion 7=curiosity 8=desire 9=disappointment 10=disapproval 11=disgust 12=embarrassment 13=excitement 14=fear 15=gratitude 16=grief 17=joy 18=love 19=nervousness 20=optimism 21=pride 22=realization 23=relief 24=remorse 25=sadness 26=surprise 27=neutral. Text: "text".

3.2.5 Prompt 5

> Analyze this text's emotional tone step-by-step. Text: "text" Then provide emotion indices (0-27 ONLY) ONLY with the corresponding emotion numbers (0-27) between square brackets. Available emotions: 0=admiration 1=amusement 2=anger 3=annoyance 4=approval 5=caring 6=confusion 7=curiosity 8=desire 9=disappointment 10=disapproval 11=disgust 12=embarrassment 13=excitement 14=fear 15=gratitude 16=grief 17=joy 18=love 19=nervousness 20=optimism 21=pride 22=realization 23=relief 24=remorse 25=sadness 26=surprise 27=neutral.

3.3 Results

For each model, all prompts were tested on 5 trial examples. For each trial example, a *best prompt* was chosen and out of the final 5 *best prompts*, the one with the highest occurrence was selected as the overall best prompt to be used for running zero-shot classification on the entire test dataset. The results of each model are as follows:

Model	Best Prompt	Best Prompt Occurrences
AquilaChat2 34B - GGUF	None	
Deepseek LLM 7B - GGUF	Prompt 4	2

Table 3: Zero-shot Classification Results

For the AquilaChat2 34B model, none of the prompts were able to classify any of the trial examples correctly. Thus, prompt 1 was chosen as the default prompt for this model. Apart from this, a few interesting observations for both models are:

- The AquilaChat2 34B model incorrectly labelled 'neutral' as 'amusement' for all prompts.
- The Deepseek LLM 7B model incorrectly labelled 'annoyance' as 'anger' for 3 out of 5 prompts.
- The Deepseek LLM 7B model was able to correctly classify 'anger' for almost all prompts.

4 Reflection

During the completion of this assignment, I learned just how much impact a seemingly trivial parameter can have on the time it takes to train a model when fine-tuning it. I also learned how to use the Optuna framework to optimize hyperparameters and how hyperparameters can drastically change the performance of a model. I also learned just how useful MPS actually is when compared to a CPU. The most unexpected and possibly frustrating part of this assignment was how it took 10 hours to train the RoBERTa model with my unoptimized script the first time I ran it. Building on to this, I also realized how despite training the odel for 5 whole epochs, the accuracy rate still remained around 55% which was disappointing.

I also spent a considerable amount of time trying to figure out why my fine-tuned models, especially the DistilBERT model had such extremely poor performance. I have narrowed it down to the following possible reasons:

- **Early Stopping Issue**: The DistilBERT model only completed 0.29 epochs before stopping which likely happened because the F1 score likely did not improve for 2 consecutive evaluations (where `eval_steps=100` and `early_stopping_patience=2`) triggering early stopping.
- **Multi-Label Classification**: The `compute_metrics` function is using a threshold of 0.5 for all labels which is not optimal as it does not account for imbalanced classes. This can lead to poor performance because some emotions in the dataset are pretty rare in terms of occurrence.
- **Loss Function**: The use of the `BCEWithLogitsLoss` without class weights did not account for an imbalanced dataset.

5 Use of Generative AI

The use of Generative AI for this assignment has been detailed below in accordance with the course policy.

5.1 How it was used

Claude by Anthropic was used for a portion of this assignment to optimize the python script for fine-tuning models. The inital script provided in the prompt by me had absurdly ambitious parameters and was extremely slow, to the point where it took approximately 10 hours to optimize and train one trial of a model. Along with the entire script, a prompt describing the issue and the hardware configurations of my device was provided to ensure optimal use.

This resulted in Claude providing me an optimized version of the original script with the following key changes:

- Preserving existing trials by adding code to detect trial directories and setting up a mechanism to use parameters from completed trials.
- Using Python's Pickle library and adding preprocessing caching with pickle to avoid repeated tokenization.
- Increasing dataloader workers and gradient accumulation steps.
- Changing evaluation from every epoch to every 500 steps.
- Adding warmup and gradient norm clipping for stability.
- Reducing the hyperparameter search space around successful values if trial results exist.
- Using Metal Performance Shaders (MPS) to leverage Apple's Metal API for GPU acceleration on M2 chips.

In addition to these changes suggested by Claude, the following changes were also implemented to further optimize the script:

- Reducing maximum length from 128 to 40 since a feature of the dataset dictates that maximum length of each comment has been capped at 30 tokens. An additional 10 tokens have been added as a buffer for any potential discrepancies.
- Reducing the number of trials to 2.
- Reducing batch sizes from 8-16 to 4-8.
- Preventing the script from using more than 4 cores to avoid causing thermal throttling.

The original script and subsequent changes can be tracked using the GitHub repository.

5.2 Associated Carbon Footprint

To use the Machine Learning Emissions Calculator, the following options were selected:

- **Name of Model**: Claude
- **Hardware Type**: T4
- **Time Used**: 1 Hour
- **Provider**: Amazon Web Services (AWS)
- **Region of Computer**: Canada (Central)
- **How the Values were Determined**: Since the website does not provide any options that durectly match Apple's GPU, T4 was selected as the best match since it is a lower-power GPU that is somewhat closer to the M2 GPU in terms of power efficiency.