Deep Learning Project Documentation

Project Title: Mental health detection from social media posts

Team Members

1. Aung Thu Phyo
2. Su Mon Hlaing
3. Moe Thet Hmue
4. Nang Mon Kham
5. Phyo Ko Ko

Model Refinement

Model Refinement

1. Overview

The model refinement phase focused on optimizing the best-performing architecture identified during the initial experimentation phase. After evaluating **DistilBERT**, **RoBERTa**, and **Multilingual BERT**, **DistilBERT** was selected as the primary model due to its superior balance of accuracy (highest among the three) and inference speed. This phase aimed to maximize DistilBERT's ability to classify mental health statuses by verifying optimal training parameters and enhancing data quality through preprocessing.

2. Model Evaluation

Initial training was conducted on three distinct architectures to establish a baseline. The results on the hold-out test set were as follows:

- **DistilBERT (distilbert-base-uncased):** Achieved the highest performance with an accuracy of **93.16%** and an F1-score of **93.08%**.
- **RoBERTa (roberta-base):** Performed strongly with ~92% accuracy but was computationally heavier than DistilBERT.
- **Multilingual BERT (bert-base-multilingual-cased):** Achieved ~85% accuracy. While capable of handling multiple languages natively, it underperformed compared to the specialized English models.

3. Refinement Techniques

To ensure the model was performing at its peak potential, we employed the following refinement techniques:

- **Hyperparameter Optimization (Optuna):** We utilized automated hyperparameter search to explore different learning rates and training durations to confirm the optimal configuration.
- **Text Preprocessing:** We implemented a cleaning pipeline that removes stopwords and non-alphabetic characters to reduce noise in the input tokens.

4. Hyperparameter Tuning

We utilized the Optuna framework integrated with the Hugging Face Trainer to perform a Bayesian search of the hyperparameter space.

- **Search Space:**
    - Learning Rate: 1×10^{-5} to 5×10^{-5}.
    - Num Train Epochs: 2 to 4.
    - Batch Size: 32.
- **Outcome:** The search ran for 10 trials. The results showed that the optimized hyperparameters yielded accuracy scores nearly identical to the baseline (~93.1%).
- **Insight:** This "null" result was significant—it validated that the standard transfer learning parameters were already highly efficient for this specific dataset. Consequently, we proceeded with the verified baseline settings, avoiding the need for computationally expensive training schedules.

5. Feature Selection

Feature selection in this NLP context was achieved through text preprocessing before tokenization. By filtering out **NLTK stopwords** and removing special characters/numbers, we reduced the vocabulary size. This allowed the self-attention mechanism of DistilBERT to focus on semantically meaningful emotional words rather than syntactic noise.

Test Submission

1. Overview

The test submission phase focused on building a robust inference pipeline capable of handling real-world, unstructured text. A key requirement addressed in this phase was **language agnosticism**—ensuring the model could handle non-English inputs despite being trained on English data.

2. Data Preparation for Testing

- **Input Source:** A new dataset, scraped_data.csv, was used to simulate a batch production environment.
- **Translation Layer:** A preprocessing step was added using langdetect and deep_translator. The system automatically detects if the input text is non-English (e.g., Burmese) and translates it to English *before* tokenization. This allows the model to serve a global audience without retraining on multilingual datasets.

3. Model Application

The fine-tuned DistilBERT model and its tokenizer were loaded from the my_saved_bert_model directory. A custom wrapper function, predict_sentiment, was created to encapsulate the pipeline:

1. **Preprocessing:** Detect language $\rightarrow$ Translate (if needed).
2. **Tokenization:** Truncation and padding to 200 tokens.

3. **Inference:** Forward pass through the model on the GPU.
4. **Output Mapping:** Decoding the logits to specific class labels (e.g., "Anxiety").

4. Test Metrics

The model was evaluated on the 20% held-out test set, demonstrating high reliability:

- **Accuracy:** 93.16%
- **Precision (Weighted):** 93.13%
- **Recall (Weighted):** 93.16%
- **F1-Score:** 93.08%
- **ROC AUC:** >0.98 for distinct classes, indicating excellent separability.

5. Model Deployment

For deployment, the trained model artifacts (model.safetensors, config.json, vocab.txt) were saved to a persistent directory. A standalone script was developed to load these artifacts, ingest a CSV file, generate predictions row-by-row, and export the results to scraped_data_with_predictions.csv.

6. Code Implementation

Inference Logic with Translation:

```
Python

def predict_sentiment(input_text):

   try:

      # 1. Language Detection & Translation

      if detect(input_text) != 'en':

         # Translate non-English text to English

         input_text = translator.translate(input_text)


      # 2. Tokenization

      inputs = tokenizer(input_text, return_tensors='pt', padding=True, truncation=True,
max_length=200)

      inputs = {k: v.to(device) for k, v in inputs.items()}


      # 3. Prediction

      with torch.no_grad():

         outputs = model(**inputs)
```

```
        # 4. Label Decoding
        prediction_idx = torch.argmax(outputs.logits, dim=1).item()
        return label_encoder.inverse_transform([prediction_idx])[0]

    except Exception as e:
        return f"Error: {str(e)}"
```

Hyperparameter Search with Optuna:

```
Python
def my_hp_space(trial):
    return {
        "learning_rate": trial.suggest_float("learning_rate", 1e-5, 5e-5, log=True),
        "num_train_epochs": trial.suggest_int("num_train_epochs", 2, 4),
        "per_device_train_batch_size": trial.suggest_categorical("per_device_train_batch_size", [32]),
    }

# Validating model parameters
best_run = trainer.hyperparameter_search(
    direction="maximize",
    backend="optuna",
    hp_space=my_hp_space,
    n_trials=10
)
```

Inference Pipeline with Translation:

```
Python
def predict_sentiment(input_text):
    try:
        # Detect and Translate
        if detect(input_text) != 'en':
            input_text = translator.translate(input_text)

        # Tokenize and Predict
```

```
        inputs = tokenizer(input_text, return_tensors='pt', padding=True, truncation=True,
max_length=200)
        inputs = {k: v.to(device) for k, v in inputs.items()}


        with torch.no_grad():
            outputs = model(**inputs)


        prediction_idx = torch.argmax(outputs.logits, dim=1).item()
        return label_encoder.inverse_transform([prediction_idx])[0]


    except Exception as e:
        return f"Error: {str(e)}"
```

Conclusion

The project successfully identified **DistilBERT** as the optimal model for mental health text classification, achieving a robust **93% accuracy**. The refinement phase utilized **Optuna** to validate the model's hyperparameter stability. Finally, the test submission addressed real-world usability by integrating a translation layer, ensuring the system can interpret distress signals regardless of the user's native language.

References

1. **Hugging Face Transformers:** For DistilBERT, RoBERTa, and BERT Multilingual models.
2. **Optuna:** For automated hyperparameter optimization.
3. **Deep Translator & Langdetect:** For the multi-lingual preprocessing layer.
4. **Scikit-Learn:** For metrics (Confusion Matrix, ROC Curve) and evaluation utilities.