Project Overview

The tasks are divided as follows:

1. **Sentence Transformer Implementation** - Encoding sentences into fixed-length embeddings.
2. **Multi-Task Learning Expansion** - Supporting both classification and sentiment analysis tasks.
3. **Training Considerations** - Exploring different training strategies, including freezing layers and transfer learning.
4. **Layer-wise Learning Rate Implementation (BONUS)** - Setting specific learning rates for each layer in the transformer.

Task 1: Sentence Transformer Implementation

Objective

To implement a sentence transformer model that encodes input sentences into fixed-length embeddings.

Model Selection and Design Choices

For the backbone model, **MPNet** (pre-trained as all-mpnet-base-v2 from Hugging Face's Transformers library) was selected because of its high performance on sentence embedding tasks. MPNet excels at generating high-quality embeddings that capture semantic meanings effectively, making it a great choice for tasks that require nuanced sentence representations.

Design Choices

1. **Embedding Size**: The embedding size was set to 768 dimensions, aligning with MPNet's output dimensions, ensuring compatibility with downstream tasks.
2. **Pooling Strategy**: To obtain a fixed-length embedding for each sentence, a mean pooling layer was applied over the token embeddings generated by MPNet. Mean pooling captures the contextual representation of the sentence effectively without losing too much information.
3. **Framework**: The model was implemented using the **Transformers** library by Hugging Face, which simplifies loading pre-trained models and integrating them into custom architectures.

Testing the Model

After implementing the encoder, the model was tested with sample sentences, and embeddings were printed to verify correct functionality. This test confirmed that the embeddings were produced in the expected format and could be used for downstream tasks.

Task 2: Multi-Task Learning Expansion

Objective

Expand the sentence transformer model to support multi-task learning, with:

Task A: Sentence Classification.
Task B: Sentiment Analysis.

Architecture Adjustments

To support multi-task learning, the following changes were made to the architecture:

1. **Separate Classification Heads**: Two separate heads were added on top of the transformer backbone:
   a. **Head for Task A**: A classification layer with softmax activation, tailored to the number of classes in the custom classification task.
   b. **Head for Task B**: A binary classification layer for sentiment analysis (positive vs. negative).
2. **Shared Backbone, Task-Specific Heads**: The MPNet transformer backbone is shared by both tasks, while each head is trained on task-specific data. This setup enables the model to leverage the shared semantic knowledge from the backbone, while specializing for each task through the separate heads.
3. **Loss Calculation**: During training, task-specific losses were computed independently and then combined for optimization. This ensured that each task contributed to the gradient updates without interference from the other.

Challenges and Solutions

- **Balancing Task-Specific Learning**: Task-specific heads allowed the model to specialize without requiring separate transformer instances for each task, reducing computational overhead.
- **Data Format and Collation**: Ensuring consistent data collation and batching for multi-task input required custom handling for tokenization and output processing.

Testing the Multi-Task Model

The multi-task model was tested by feeding in sample sentences and observing the predictions for each task. Both classification and sentiment tasks yielded results, verifying that the model's architecture correctly handled multi-task inputs and outputs.

Evaluation Results

Task A (Sentence Classification)

- **Accuracy**: 78.00%
- **Detailed Report**:
- **Positive Sentiment**: Precision 0.95, Recall 0.97, F1-score 0.96
- **Negative Sentiment**: Precision 0.60, Recall 0.92, F1-score 0.73
- **Technical Issues**: Precision 0.00, Recall 0.00, F1-score 0.00
- **Purchase or Value Concerns**: Precision 0.00, Recall 0.00, F1-score 0.00

These scores highlight a strong performance in classifying **Positive Sentiment** but significant difficulties in accurately predicting **Technical Issues** and **Purchase or Value Concerns** due to low support and potential overlap with other classes.

- **Macro and Weighted Averages**:
- **Macro Average**: Precision 0.39, Recall 0.47, F1-score 0.42 (affected by poor performance in minority classes)
- **Weighted Average**: Precision 0.67, Recall 0.78, F1-score 0.71 (boosted by the model's performance on majority classes)

Task B (Sentiment Analysis)

- **Accuracy**: 95.50%
- **Detailed Report**:
- **Positive Sentiment**: Precision 0.95, Recall 0.96, F1-score 0.96
- **Negative Sentiment**: Precision 0.96, Recall 0.95, F1-score 0.95

Task B demonstrates a well-balanced and robust performance in sentiment analysis, achieving high precision, recall, and F1-scores for both positive and negative sentiment classes.

Suggestions for Improvement

1. **Increase Data for Minor Classes**: The performance for "Technical Issues" and "Purchase or Value Concerns" could be improved by gathering more data samples for these classes, as they are underrepresented in the dataset.
2. **Apply Class Weighting**: Using class weights in the loss function can help the model pay more attention to underrepresented classes, improving recall and precision for these categories.
3. **Multi-Stage Classification**: For Task A, consider a multi-stage classification approach. First, classify sentences into "Positive" or "Negative" categories, and then further classify "Negative" sentences into subcategories like "Technical Issues" or "Purchase or Value Concerns."
4. **Error Analysis**: Perform a detailed error analysis on misclassified instances to better understand where the model struggles, which can reveal patterns that could guide future feature engineering or model adjustments.

Task 3: Training Considerations

Objective

To evaluate different training strategies based on which layers or heads to freeze/unfreeze, and to propose a transfer learning strategy.

Scenario Evaluations

1. **Entire Network Frozen**:
    i. **Description**: Freezing the entire network would mean only using the pre-trained embeddings without updating any weights during training.
    ii. **Implication**: This approach significantly reduces computational costs but limits the model's ability to adapt to task-specific requirements, likely leading to suboptimal performance.
    iii. **Recommendation**: Suitable only for cases where computational resources are highly constrained, or the downstream tasks are very similar to the tasks on which MPNet was originally trained.
2. **Transformer Backbone Frozen**:
    i. **Description**: Freezing the MPNet backbone while allowing the task-specific heads to train.
    ii. **Implication**: This allows task-specific heads to adapt while preserving the generalized knowledge in the transformer. This approach is useful for tasks that only need minor adjustments to the classification heads without overhauling the backbone.

iii. **Recommendation**: Recommended when there is limited labeled data for fine-tuning, as it minimizes overfitting while leveraging pre-trained semantic knowledge.

3. **Only One Task-Specific Head Frozen**:
   i. **Description**: Freezing one of the task-specific heads while fine-tuning the other.
   ii. **Implication**: This scenario is beneficial in situations where one task (e.g., sentiment analysis) is well-represented in the pre-trained model, while the other (e.g., custom classification) requires more domain adaptation.
   iii. **Recommendation**: Useful for incrementally updating the model for new tasks without retraining all parts of the network, especially in cases where computational resources are limited.

Transfer Learning Strategy

Chosen Pre-trained Model: all-mpnet-base-v2

MPNet was chosen due to its high-quality embeddings for sentence-level tasks.

Freezing and Unfreezing Layers:

- **Unfreeze Classification Heads**: Always leave the task-specific heads unfrozen for adapting to new tasks.
- **Layer-Wise Unfreezing**: Gradually unfreeze the last few layers of the transformer backbone during fine-tuning to enable adaptation to new domain-specific language patterns while retaining generalized knowledge.

Rationale:
Gradual unfreezing and fine-tuning of the last layers allow the model to learn task-specific patterns without overfitting. This approach strikes a balance between adapting to new tasks and retaining the semantic capabilities of the pre-trained model.

Task 4: Layer-Wise Learning Rate Implementation (BONUS)
Objective

Implement layer-wise learning rates for the multi-task sentence transformer.

Implementation

- **Layer-Wise Learning Rates**: Different learning rates were assigned to different parts of the model. The task-specific heads received a higher learning rate to encourage faster adaptation, while the lower layers of the transformer received a smaller learning rate to prevent overwriting pre-trained knowledge.
- **Learning Rate Schedule**:
- **Classification Heads**: Set at a higher learning rate (1e-4) to adapt quickly.
- **Upper Transformer Layers**: Set at a moderate learning rate (5e-5) for fine-tuning.
- **Lower Transformer Layers**: Set at a low learning rate (1e-5) to retain pre-trained