TASK 1: Implement a sentence transformer model using any deep learning framework of your choice. This model should be able to encode input sentences into fixed-length embeddings. Test your implementation with a few sample sentences and showcase the obtained embeddings. Describe any choices you had to make regarding the model architecture *outside of the transformer backbone.*

Explanation of Choices

1. Model Choice:

- Model: sentence-transformers/paraphrase-mpnet-base-v2 is chosen for its efficiency and accuracy in generating sentence embeddings. This model is fine-tuned for semantic textual similarity tasks, making it suitable for encoding sentences into meaningful embeddings.
- Library: Hugging Face Transformers library provides an easy-to-use interface and a variety of pre-trained models, allowing for quick and efficient implementation.

• **Tokenization:**

- Sentences are tokenized using the model's tokenizer with padding and truncation to ensure all input sequences are of equal length, necessary for batch processing.

• **Embedding Extraction:**

- **Mean Pooling:** The last hidden state of the model's output is averaged across the sequence length to obtain a fixed length embedding for each sentence. This approach is simple and effective for generating sentence-level embeddings.

Task 2

Expand the sentence transformer to handle a multi-task learning setting.

1. **Task A:** Sentence Classification – Classify sentences into classes.
2. **Task B:** Perform Sentiment Analysis

Architectural Changes for Multi-Task Learning

1. **Shared Encoder:**
   - Used a pre-trained transformer model as a shared encoder to generate embeddings for input sentences.
2. **Task-Specific Heads:**
   - Add a classification head for Task A.
   - Add a sentiment analysis head for Task B.
3. **Multi-Task Loss:**
   - Combine the losses from both tasks to perform joint training.

Explanation of the Changes

1. **Custom Model Definition:**
    - **MultiTaskModel:** Inherits from nn.Module.
    - **Encoder:** Shared transformer encoder (AutoModel).
    - **Task-Specific Heads:** Separate linear layers (classifier_task_a and classifier_task_b) for each task.
2. **Forward Method:**
    - Encodes the input sentences using the shared transformer encoder.
    - Applies mean pooling on the output to get fixed-length embeddings.
    - Feeds the embeddings to the task-specific heads to get logits for each task.
3. **Loss Calculation:**
    - Uses cross-entropy loss for both tasks.
    - Combines the losses from both tasks to compute the total loss.
4. **Training Step:**
    - A simple training step is demonstrated, including forward pass, loss computation, backward pass, and optimization.

Task 3: Training Considerations

Discuss the implications and advantages of each scenario and explain your rationale as to how the model should be trained given the following:

1. If the entire network should be frozen.
2. If only the transformer backbone should be frozen.
3. If only one of the task-specific heads (either for Task A or Task B) should be frozen.

Training Considerations for Multi-Task Learning

When deciding how to train a multi-task model, there are several considerations to take into account regarding which parts of the model to freeze or unfreeze.

1. **If the entire network should be frozen:**

    Implications:
    - The model will not learn or adapt to the specific tasks.
    - The pre-trained model's embeddings will be used as fixed features.

Advantages:

- Reduces computational requirements since no gradients need to be calculated.
- Prevents overfitting on small datasets.
- Useful when the pre-trained embeddings are expected to be highly relevant.

2. **If only the transformer backbone should be frozen:**

Implications:

- The pre-trained embeddings will be used as fixed features.
- The task-specific heads will adapt to the new tasks.

Advantages:

- Leverages the rich pre-trained features while allowing customization for specific tasks.
- Reduces the risk of overfitting compared to training the entire network.
- Efficient in terms of computational resources.

Scenario:

- This is useful when the downstream tasks are somewhat related to the pre-training tasks but still require some adaptation, and the dataset size is moderate.

3. **If only one of the task-specific heads should be frozen:**

Implications:

- The model can adapt to one task while keeping the learned parameters for the other task fixed.

Advantages:

- Allows focusing training resources on the task that requires more adaptation.
- Prevents catastrophic forgetting for the frozen task.

Scenario:

- This approach is beneficial when one task is already well learned and needs to be preserved while the other task requires further tuning.

Consider a scenario where transfer learning can be beneficial. Explain how you would approach the transfer learning process, including:

1. The choice of a pre-trained model.
2. The layers you would freeze/unfreeze.
3. The rationale behind these choices.

Transfer Learning Approach

Scenario:

- **Tasks:**
    - Task A: Sentence Classification
    - Task B: Sentiment Analysis
- **Pre-trained Model:**
    - A transformer model like sentence-transformers/paraphrase-mpnet-base-v2 which is pre-trained on a diverse set of sentence pairs for tasks such as paraphrase identification, sentence similarity, etc.

Steps:

1. **Load Pre-trained Model:**
    - Use a pre-trained transformer model that is relevant to the downstream tasks.
2. **Freeze/Unfreeze Layers:**
    - **Initial Phase:**
        - Freeze the transformer backbone to leverage the pre-trained features.
        - Train only the task-specific heads.
    - **Fine-Tuning Phase:**
        - Gradually unfreeze the transformer layers, starting from the top (closer to the output).
        - Use a lower learning rate for the transformer backbone compared to the task-specific heads.

Rationale:

- **Choice of Pre-trained Model:**
    - Models like sentence-transformers/paraphrase-mpnet-base-v2 are chosen because they are pre-trained on tasks that involve understanding sentence-level semantics, making them suitable for tasks like sentence classification and sentiment analysis.
- **Freezing Strategy:**
    - **Initial Phase:**
        - Freezing the backbone allows the model to leverage the robust, generalized features learned during pre-training, ensuring that the model does not deviate drastically from these useful representations.

- **Fine-Tuning Phase:**
    - Gradually unfreezing layers allows the model to adapt these pre-trained features to the specific nuances of the new tasks without losing the valuable knowledge embedded in the earlier layers.
    - Using a lower learning rate for the backbone helps to make fine-tuned adjustments without overfitting or destabilizing the pre-trained weights.

Summary

- **Freezing the entire network:** Useful when leveraging pre-trained embeddings as fixed features and preventing overfitting on small datasets.
- **Freezing only the transformer backbone:** Allows leveraging pre-trained features while adapting to specific tasks, balancing between overfitting and learning new task-specific information.
- **Freezing one task-specific head:** Useful for preserving performance on one task while adapting to another, preventing catastrophic forgetting.
- **Transfer Learning Strategy:**
    - Initial freezing of the transformer backbone to retain pre-trained features.
    - Gradual unfreezing with differential learning rates to adapt pre-trained features to new tasks effectively, minimizing the risk of overfitting and ensuring stable training.

TASK 4

Rationale for Specific Learning Rates

- **Base Learning Rate:** The base learning rate (base_lr) is set to a value like 1e-4, which is typical for fine-tuning pre-trained models.
- **Learning Rate Decay:** The decay factor (lr_decay) is set to 0.95. This means that each successive layer has a learning rate that is 95% of the previous layer's learning rate. This gradual decay ensures that earlier layers (which capture more general features) are updated more conservatively, while later layers (which capture more task-specific features) are updated more aggressively.

Benefits of Layer-Wise Learning Rates

1. **Better Fine-Tuning:**
    - Allows the model to retain the general features learned in earlier layers while adapting more specific features in later layers to the new tasks.
2. **Reduced Overfitting:**
    - By updating earlier layers less aggressively, the model is less likely to overfit to the specific dataset, particularly if the dataset is small.
3. **Improved Convergence:**

   - Different learning rates for different layers can lead to faster and more stable convergence during training.

Multi-Task Setting Benefits

- **Shared Features:** In a multi-task setting, layer-wise learning rates help ensure that shared features across tasks (captured in earlier layers) are preserved, while task-specific adaptations (captured in later layers) are fine-tuned more aggressively.
- **Balanced Training:** This approach balances the need to learn task-specific features while maintaining the integrity of the general features useful for multiple tasks, leading to better overall performance on both tasks.

By implementing layer-wise learning rates, the model can achieve more effective and efficient fine-tuning, particularly in complex scenarios like multi-task learning.