Fetch Take Home Report

Task 1

Sentence Transformer Implementation

I used the all-MiniLM-L6-v2 transformer from HuggingFace's sentence-transformers. After tokenizing the sentences and passing the tokenized sentences through the model, I applied mean pooling to the model output to reduce the output to a sentence-level representation. It also helps ensure the whole sentence contributes equally to the final embedding, which will be helpful for sentiment analysis in Task 2.

Sample Data Embeddings

Text	Class	Sentiment	384 Dimension Embedding Stats
"The government's response to the crisis was utterly disappointing."	Politics	Negative	Min: -0.7380 Max: 0.7802 Mean: 0.0014 Median: 0.0157 Std: 0.2640
"The team's comeback victory was incredibly thrilling to watch."	Sports	Positive	Min: -0.8576 Max: 0.9114 Mean: 0.0033 Median: 0.0065 Std: 0.2612
"The new smartphone was released with standard features this year."	Technology	Neutral	Min: -0.6628 Max: 0.7185 Mean: 0.0007 Median: 0.0111 Std: 0.2461
"The side effects of the new medication were concerning."	Health	Negative	Min: -0.7840 Max: 0.7400 Mean: -0.0010 Median: 0.0035 Std: 0.2456
"That movie was a masterpiece from start to finish."	Entertainment	Positive	Min: -0.5712 Max: 0.6973 Mean: 0.0001 Median: -0.0049 Std: 0.2263

"The senator introduced a new bill in the house today."	Politics	Neutral	Min: -0.6292 Max: 0.7531 Mean: 0.0029 Median: -0.0031 Std: 0.2514
"The referee's decisions ruined the entire match experience."	Sports	Negative	Min: -0.8370 Max: 1.0321 Mean: 0.0084 Median: 0.0005 Std: 0.2876
"This app makes managing tasks so much easier and faster."	Technology	Positive	Min: -0.6784 Max: 0.6954 Mean: 0.0091 Median: 0.0000 Std: 0.2386
"The doctor explained the procedure clearly and professionally."	Health	Neutral	Min: -0.8404 Max: 0.9054 Mean: 0.0027 Median: 0.0041 Std: 0.2622
"The show's plot was predictable and lacked emotional depth."	Entertainment	Negative	Min: -0.9048 Max: 1.1217 Mean: 0.0008 Median: 0.0037 Std: 0.2775

Task 2

Sentence Classification (Task A) & Sentiment Analysis (Task B)

To expand the sentence transformer into a multi-task learning setting, I added a head for sentence classification and another for sentiment analysis, each with the same architecture as follows:

Layer	Reasoning
nn.Linear(self.hidden_size, 256)	This layer uses a weighted transformation to reduce the feature dimension size. Reducing the dimension size makes the network focus on more important features and acts as a bottleneck. This forces the model to learn more abstract concepts from the data instead of memorizing given features. It also reduces computational cost during training.

nn.LayerNorm(256)	This layer normalizes the features across each sentence's embedding. It ensures extreme values do not affect the gradient and prevents the chance of exploding or vanishing gradients. LayerNorm() is used instead of BatchNorm() to maintain consistency since the sentence transformer also uses layer normalization. In addition, the model input has varying lengths (different sentence lengths), which can be reflected in extreme values in the sample embeddings. So, the different sample lengths make normalizing each sample more suitable.
nn.ReLU()	ReLU introduces non-linearity, which allows the model to capture more complex, non-linear patterns in the data.
nn.Dropout(0.1)	This layer randomly deactivates 10% of the elements in the input tensor, forcing the network not to rely too heavily on any single dimension. This improves generalization through randomization.
nn.Linear(256, 64)	This layer reduces the feature dimension size to distill the most important patterns/features.
nn.LayerNorm(64)	This layer normalizes the features to prevent extreme gradient values.
nn.ReLU()	This layer introduces non-linearity to allow the network to capture more complex patterns in the reduced-size feature space.

nn.Linear(64, 5)
nn.Linear(64, 3)
	This layer transforms the features into the final output logits for sentence classification (5 classes: politics, sports, technology, health, and entertainment) and sentiment analysis (negative, neutral, positive).

Task 3

Training Considerations

1) If the entire network were frozen, training would not be possible as weights would not be updated. Instead, the model could be used for fast inference (faster due to the lack of backpropagation) and evaluating its effectiveness after the pre-training phase. Freezing the whole network would be useful for gauging zero-shot learning since the model won't learn from any new data. However, the model may perform poorly on data dissimilar to the training data since the model can not learn when the network is frozen.
2) If the transformer backbone were frozen, there would be fewer trainable parameters and training would be faster and consume less memory. The backbone would preserve the language representations from pretraining while the model trains the heads for sentence

classification (Task A) and sentiment analysis (Task B). Freezing the transformer backbone would be useful when you trust the performance of the transformer backbone.

3) If either the sentence classification or sentiment analysis head are frozen, only the other unfrozen head and the transformer backbone would be trained. The frozen head will mostly maintain its performance while the other will learn incrementally. However, the frozen head's performance can decrease if the shared backbone weights drastically change. Freezing one head is useful when the model performs well for one task and you want to fine-tune performance on the other task.

Transfer Learning

The pre-trained model should be trained on tasks related to the new task in order to use transfer learning. The tasks should require common features. For my model with a shared transformer backbone with heads for sentiment analysis and sentence classification, transfer learning could be applied to tasks like emotion classification and topic detection. Our model can leverage its pre-trained understanding of semantic structure and sentiment to classify the emotion of a sentence. The pre-trained model would also have a strong base for topic detection since sentence classification is like a general form of topic detection. The model already has features that classify sentences into classes, reducing the scope of topics and making topic detection easier. However, tasks that do not use sentence representations would not benefit much from transfer learning. Named entity recognition would be a poor choice for transfer learning since it requires token-level understanding when the model is pre-trained to have sentence-level understanding.

For tasks closely related to sentiment analysis and sentence classification (like emotion detection), the transformer backbone can be frozen while training in order to only fine-tune the emotion detection head. Since the tasks require similar levels of sentence understanding, you can save computation by preventing the backbone from learning. If the task requires more specific features (like emotion detection), we can unfreeze some of the higher layers of the backbone. Higher-level layers are responsible for more precise features, while lower layers are responsible for more general language understanding. Only unfreezing the higher layers preserves general linguistic understanding while allowing it to learn more specific features. If the task is completely unrelated, the entire model should remain unfrozen. Applying our model to Named Entity Recognition would require token-level features, which would require learning from lower layers.

Task 4

Training Loop Implementation

For our data, we assume there are labels "class" and "embedding". Each data entry is not guaranteed to have either label. Samples without a label do not affect their corresponding loss values. During each training epoch, the model iterates through the data in batches. The data goes through the model's forward pass, which calculates the embeddings and performs mean pooling to aggregate the features into a fixed-size sentence embedding. The shared embeddings are then passed through the sentence classification and sentiment analysis heads as described in Task 2. It conditionally computes the task-specific cross-entropy loss for samples when the corresponding labels are present. The task losses are summed together to get the overall batch loss, which is backpropagated to update the model parameters. After every epoch, it calculates the accuracy and F1 score for sentiment analysis and sentence classification separately. The F1 scores are macro-averaged for each task, meaning the F1 score is calculated for each class and averaged where each class has equal weight. It also outputs the average loss for each epoch.