Fetch Assessment
Discussion Questions Part 3 (Machine Learning)

Mahitha Sannala

1. Consider the scenario of training the multi-task sentence transformer that you implemented in Task 2. Specifically, discuss how you would decide which portions of the network to train and which parts to keep frozen.

For example,

- When would it make sense to freeze the transformer backbone and only train the task specific layers?
- When would it make sense to freeze one head while training the other?

Training Strategy for Network Portions:
Deciding which parts of a multi-task network to freeze or train depends on various factors, including the amount of available training data, the complexity of the tasks, and the pre-trained model's relevance to those tasks.

Freezing the Transformer Backbone
When to Freeze: If the transformer model (like BERT) is already well-tuned for similar types of data or tasks as those in your application, you might choose to freeze the transformer backbone. This is particularly practical when your training dataset is small, which might not be sufficient to fine-tune a large model without overfitting.
Advantages: Freezing the backbone reduces the computational load and avoids overfitting on smaller datasets. It also preserves the pre-trained features, which have been learned from a much broader dataset.

Freezing One Head While Training the Other
When to Freeze: If one task (say Task B) is well-optimized or less critical and you have gathered more data or insights for another task (Task A), you might choose to freeze Task B's head. This allows you to focus computational resources and model capacity on improving Task A without disturbances from the already satisfactory performance of Task B.
Use Case: This approach is suitable when tasks have differing levels of maturity or importance in the application, or when updates to the dataset affect one task more than the other.

2. Discuss how you would decide when to implement a multi-task model like the one in this assignment and when it would make more sense to use two completely separate models for each task.

Deciding Between Multi-task Model and Separate Models
Choosing between a multi-task model and separate models for each task depends on the relationship between the tasks, data availability, and the intended application.

Multi-task Model
When to Use: Multi-task models are beneficial when the tasks are related and can benefit from shared representations, leading to improved performance on both tasks. This approach is also more efficient in terms of memory and computational resources since it leverages a shared backbone.
Example: If Task A is Sentence Classification and Task B is Sentiment Analysis, both tasks can benefit from similar types of linguistic understanding and contextual embeddings.

Separate Models
When to Use: If the tasks are very distinct or require highly specialized processing that could interfere with each other, separate models might be more effective. This approach also allows each model to be optimized independently, which can be crucial if the tasks are of differing priorities or scales.
Example: If one task is about medical text analysis and the other is about financial news classification, the domain-specific features might be too different to benefit from shared representations.

3. When training the multi-task model, assume that Task A has abundant data, while Task B has limited data. Explain how you would handle this imbalance.

Handling Data Imbalance in Multi-task Learning
When dealing with a scenario where Task A has abundant data while Task B has limited data, special techniques can be applied to balance training and prevent the model from becoming biased towards Task A.

Techniques to Handle Data Imbalance
Data Augmentation for Task B: Increase the effective size of the smaller dataset through techniques such as synonym replacement, back translation, or generating synthetic data via techniques like GANs.

Balanced Sampling: During training, adjust the sampling of data so that batches contain a balanced mix of data from both tasks, preventing the model from seeing too much of one task.

Task Weighting in Loss Function: Modify the loss function to weight the contributions of each task's loss differently. Task B's loss could be weighted more heavily than Task A's, compensating for its smaller dataset and encouraging the model to pay more attention to it.

Curriculum Learning: Start training with more emphasis on the task with more data (Task A), and gradually increase the emphasis on Task B as training progresses. This approach can help in stabilizing the training process and leveraging the richer dataset to boost feature extraction capabilities before focusing on the under-represented task.