ML Apprentice – Take Home Assignment

Task 1: Sentence Transformer Implementation

I used all-MiniLM-L6-v2 model from Hugging Face's sentence-transformers library because of the following:

1. This model is fine-tuned for sentence embeddings, unlike other models which are created on token-level embeddings.
2. It fares well on the memory-performance tradeoff, i.e., it is lightweight but powerful enough.
3. The model has been fine-tuned on sentence similarity and classification datasets, making it directly applicable to our tasks with minimal fine-tuning required.

To ensure proper implementation:

- I loaded the pre-trained all-MiniLM-L6-v2 model and used it to generate 384-dimensional sentence embeddings.
- I verified the embedding output using sample sentences, ensuring they were properly structured for downstream tasks.

Task 2: Multi – Task Learning Expansion

To adapt the sentence transformer for **multi-task learning**, I introduced two classification tasks:

Task A: Sentiment Analysis:- Classify a sentence as Positive (1) or Negative (0).
Application: Useful for analyzing customer feedback, product reviews, and chatbot interactions.

Task B: Intent Detection:- Classify a sentence into one of four intent categories: Question (0), Statement (1), Command (2), or Exclamation (3).
Application: Helps chatbots and virtual assistants understand user input more effectively.

Changes to Support Multi-Task Learning:

1. Two separate classifier heads were introduced:
    - A binary classification head for Sentiment Analysis.
    - A multi-class classification head for Intent Detection.

2. Shared Embeddings:- all-MiniLM-L6-v2 generates sentence embeddings, which are used for both tasks.

3. Ensured Correct Output Handling:- Sentiment predictions were checked to be in [0,1], and intent predictions were mapped to human-readable labels for better interpretability.

Explanation of Model Output:

1. Sentiment Analysis Output:

   The model was tested with two sentences:
   "I love this product!" (Positive)
   "I hate the weather today." (Negative)

   Output Predictions:

   [1, 1]

   Since the model is not trained yet, the predictions are biased, which is expected before fine-tuning.

2. Intent Detection Output:

   The model was tested with four sentences:
   "Can you tell me the time?" (Question)
   "This is the best day of my life!" (Exclamation)
   "Turn off the lights." (Command)
   "The sky is blue today." (Statement)

   Predicted Intent Categories:

   [1, 1, 1, 1] → Mapped to: [Statement, Statement, Statement, Statement]
   Here, the model incorrectly classified all the four sentences as "Statement", which is expected because the classifier has not been trained yet. Once fine-tuned on labeled data, the model should correctly assign intent categories.

Task 3: Multi – Task Learning Expansion

Freezing vs. Unfreezing Layers:

When fine-tuning pre-trained models, we have three main options:

1. Freezing the Entire Model: Neither the transformer backbone nor the classifier heads are updated.

   This approach is basically used when a pretrained model is directly used for inference without any fine-tuning. This makes it extremely cost-effective. However, the model may

exhibit substandard performance due to the generalization gap between the pretrained and the target data distributions.

This is not useful for our task since the model performance would essentially rely on the embedding model, which was not trained for our specific tasks and datasets.

2. Freezing the Transformer, Training Only Classifier Heads: We freeze the transformer backbone (all-MiniLM-L6-v2) and only train the new classifier heads (sentiment & intent classifiers) to learn task-specific patterns.

   Advantages:

   - Fast & Efficient: Training is much faster since we update only a few layers.
   - Avoids Overfitting: Pre-trained transformers have already learned rich general language representations, so they don't need to be modified unless absolutely necessary.

   This is the go-to approach because it fares well in both cost-effectiveness and performance. However, if the pre-trained sentence embeddings aren't capturing the nuances of our tasks, we might need to unfreeze some transformer layers later.

3. Unfreezing the Transformer Gradually (Fine-Tuning the Whole Model): We gradually unfreeze layers of the transformer during training. This allows the model to learn task-specific language patterns in addition to what it already knows.

   Advantages
   - If frozen embeddings don't work well, unfreezing helps adapt them to sentiment/intent detection.
   - Lets the transformer learn domain-specific nuances (e.g., customer reviews, chatbot conversations).

   Downsides
   - Computationally expensive (since we're updating the entire transformer).
   - More risk of overfitting, especially with a small dataset.

Transfer Learning Strategy:

Step 1: Start with a Frozen Transformer Backbone
- Freeze all layers of all-MiniLM-L6-v2.
- Train only the sentiment and intent classification heads.
- Monitor performance.

Step 2: Check Performance

- If accuracy is good, keep the transformer frozen.
- If accuracy is bad, start unfreezing the last few layers of the transformer.

Step 3: Gradually Unfreeze the Transformer (If Needed)

- Unfreeze the last 2-4 layers while keeping earlier layers frozen.
- Continue training with a lower learning rate.
- Fine-tune until we get the best results.

How Do We Handle Multi-Task Learning?

1. Use Task-Specific Loss Functions
    - Sentiment Analysis (Binary Classification):- Binary Cross-Entropy Loss
    - Intent Detection (Multi-Class Classification):- Categorical Cross-Entropy Loss

2. Combine Losses
    - Our final loss = loss_sentiment + loss_intent
    - This ensures that the model learns both tasks at the same time.

3. Optimization Strategy
    - Use AdamW optimizer (widely used for NLP fine-tuning).
    - Set a lower learning rate for the transformer (if unfreezing) and a higher learning rate for classifier heads.

Task 4: Training the Multi – Task Model

In this task, we trained the Multi-Task Sentence Transformer model, which we built in Task 2. The training involved two NLP tasks: Sentiment Analysis and Intent Detection

1. Data Preparation

    We used a synthetic dataset (100 sentences only for both datasets) to fine-tune the model and optimize its performance. Before training, we needed to ensure that the dataset was properly preprocessed. This involved:

    - Reading the CSV files containing sentiment and intent classification data.
    - Encoding labels:
        - Sentiment labels were mapped as 0 = Negative, 1 = Positive.
        - Intent labels were mapped into four categories using LabelEncoder.
    - Generating sentence embeddings using all-MiniLM-L6-v2 from sentence-transformers.
    - Splitting the dataset into training (80%) and validation (20%).

2. Model Training Process

   We trained the model using a multi-task learning approach, where both sentiment analysis and intent detection were trained simultaneously.

   Training Details:
   - Optimizer: AdamW (learning rate=5e-5)
   - Loss Functions:
     - Binary Cross-Entropy (BCE) for Sentiment Analysis
     - Cross-Entropy (CE) for Intent Detection
   - Batch Size: 16
   - Epochs: 10
   - Device: Used GPU (if available), else CPU

   Each batch contained both:
   - A sentiment classification sample (sentence + label)
   - An intent classification sample (sentence + label)

   For each batch:
   - Pass sentence embeddings through the transformer backbone.
   - Use the sentiment head for binary classification.
   - Use the intent head for multi-class classification.
   - Compute the losses for both tasks and backpropagate together.

3. Training Results & Observations

   During training, we tracked the combined loss (Sentiment Loss + Intent Loss). The loss remained almost constant, indicating that the model was not learning effectively. This is because the dataset is too small for the model to learn sufficient information for this task, i.e., the model is underfitting. Fine-tuning with more data could further improve performance. Moreover, using a benchmark dataset instead of creating a synthetic one may also improve performance and keep the results closer to the ground truth.