Assignment3 Report

Name: 陳安楷 Kyle Chen

Institution (school): NTHU

Student ID: 111030034

Platform (Colab/Kaggle/Local): Local (lab server)

Python version: 3.12.11

Operating system: Ubuntu 20.04.6 LTS

CPU: Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz

GPU requirement: NVIDIA GeForce GTX 1080 Ti

AI disclaimer: Whole report and code uses AI to assist writing.

Remark: For ease of grading, you are encouraged to present data in textual form rather than as images.

1. Train a RoBERTa-base model on the same dataset and report the performance of both BERT-base and RoBERTa-base models. (10%)

Results:

- **Best BERT-base:** (lr=1e-4, wd=0.01, b=32, rw=10)
    - **Test Accuracy:** 0.827
    - **Test Pearson:** 0.813
- **Best RoBERTa-base:** (lr=2e-5, wd=0.005, b=16, rw=10)
    - **Test Accuracy:** 0.785
    - **Test Pearson:** 0.799

1.1 Discuss the main differences between BERT and RoBERTa, and explain how these differences may affect performance. (5%)

Answer: My results show **BERT outperforming RoBERTa**, which is the opposite of what is typically seen. And I think the reason is my architecture design: the **Siamese architecture**.

- A Siamese model forces the encoder to create a single, meaningful vector for an entire sentence (vec_a and vec_b).
- BERT was **explicitly pre-trained on NSP**, a task that forced it to learn how to create and compare these sentence-level representations.
- RoBERTa's creators **removed the NSP task**, meaning its pre-training *never* focused on creating single sentence-level embeddings for comparison.

This suggests that for a **cross-encoder** (where models can use cross-attention), RoBERTa's robust training would likely win. But for a **Siamese** architecture that *relies* on high-quality, pre-trained sentence-level vectors, BERT's (coincidentally helpful) NSP training gave it an unexpected and decisive edge on this specific task.

Train a GPT-2 model on the same dataset and evaluate its performance. (10%)

Results:

- **Best BERT-base:** (lr=1e-4, wd=0.01, b=32, rw=10)
  - **Test Accuracy:** 0.827
  - **Test Pearson:** 0.813
- **Best GPT-2:** (lr=1e-4, wd=0.005, b=32, rw=10)
  - **Test Accuracy:** 0.810
  - **Test Pearson:** 0.792

1.2 What are the main difference of BERT and GPT2? Discuss the main differences between BERT and GPT-2, and analyze how these differences influence their performance. (5%)

Answer: The main differences between BERT and GPT-2 are:

- **Architecture:** BERT is an **encoder-only** model; GPT-2 is a **decoder-only** model.
- **Attention:** BERT is **bi-directional** (sees all tokens at once). GPT-2 is **uni-directional** (auto-regressive, only sees past tokens).
- **Purpose:** BERT is for NLU (understanding); GPT-2 is for NLG (generation).

How these differences influence their performance:

By using a Siamese architecture for all models, the comparison between GPT-2 and Bert is fair by comparing the representation's robustness after viewing both sentences**.** Since GPT-2 doesn't have CLS token, so forcing BERT and RoBERTa to also use a Siamese structure ensures an fair comparison for this NLU task.

Even on this level playing field, **BERT still wins**. The reason lies in the *quality* of the sentence vectors (vec_a and vec_b) that each model produces.

- When **BERT** (bi-directional) creates the mean-pooled vector for the premise, it does so having seen *every word in the premise* in full context. This creates a rich, high-quality vector that represents the sentence's full meaning.
- When **GPT-2** (uni-directional) creates the mean-pooled vector for the premise, the vector for each token was created *without seeing any future tokens*. This results in a fundamentally weaker representation of the sentence's overall meaning.

For an NLU task that requires deep *understanding*, the bi-directional context is a fundamental advantage. My results confirm this: even when forced into the same architecture, BERT's bi-directional encoders produce superior sentence representations for this task, leading to better performance.

2. Compare the performance of a multi-output learning model with BERT models trained separately on each sub-task. Does multi-output learning improve performance? Explain why or why not. (5%)

Answer: This experiment revealed a classic case of **positive and negative transfer** in multi-task learning. Comparing my best multi-output model with models trained only on one task shows that **multi-output learning improved performance for the regression task but hurt performance for the classification task.**

Experimental Results:

- **Best Multi-Task (BERT):**
    - **Accuracy:** 0.827
    - **Pearson:** 0.813
- **Best Single-Task (Classification-Only):**

    - **Accuracy:** 0.844

- **Best Single-Task (Regression-Only):**
    - **Pearson:** 0.761

Analysis:

- **Negative Transfer (Classification):** The classification-only model (Acc: **0.844**) performed *better* than the multi-task model (Acc: **0.827**). This is **negative transfer**. The model was "distracted" by the regression task. It had to find a compromise for its shared BERT weights to be "good enough" for both tasks, rather than finding the *perfect* weights for classification.
- **Positive Transfer (Regression):** The multi-task model (Pearson: **0.813**) performed significantly *better* than the regression-only model (Pearson: **0.761**). This is **positive transfer**. The classification task acted as a powerful regularizer, forcing the shared BERT model to learn a much richer, more nuanced understanding of the text's logical and semantic relationship. This "smarter" shared representation was then hugely beneficial for the regression head, allowing it to perform far better than when it was trained on the simpler regression signal alone.

3. Why does your mod el fail to correctly predict some data points? Please provide an error analysis and introduce how do you improve your performance. (10%)

Answer: **1. Error Analysis**

My analysis of the model's errors points to three primary weaknesses:

1. **Confusing "Neutral" and "Entailment"**: The model struggles to distinguish between high semantic similarity (relatedness) and strict logical entailment.
2. **Inconsistent Negation Handling**: The model fails to reliably identify clear contradictions, especially when they rely on a single word or phrase (e.g., "missing" vs. "dunking").
3. **Over-Inference**: The model makes assumptions and "infers" details that are not explicitly stated in the premise.

Analysis from Confusion Matrix

The raw counts show a clear pattern:

- **True: 0 (NEUTRAL), Pred: 1 (ENTAILMENT)**: **433 errors** (The largest error source)
- **True: 1 (ENTAILMENT), Pred: 0 (NEUTRAL)**: **279 errors** (The second largest)

This "Neutral-Entailment Confusion" (433 + 279 = 712 errors) accounts for over 72% of all classification mistakes. The model has a strong tendency to see two related sentences and predict **Entailment**, even when there is no strict logical proof.

The model is much better at separating opposites, with only 32 errors (27 + 5) between **Entailment** and **Contradiction**. However, it still frequently confuses **Neutral** and **Contradiction** (180 + 59 = 239 errors).

Analysis from Error Samples

The provided samples perfectly illustrate these statistical findings:

1. **Over-Inference (Predicting Entailment, True is Neutral):**
    - **Premise:** "Children in red shirts are playing in the leaves"
    - **Hypothesis:** "Children covered by leaves are playing with red shirts"
    - **Model:** Pred: 1 (ENTAILMENT)
    - **Analysis:** The model sees "children," "red shirts," and "leaves" and assumes they are related enough to be an entailment. It **incorrectly infers** that "playing in" means "covered by," which is not a guaranteed logical step.
2. **Weak Negation Handling (Predicting Neutral, True is Contradiction):**
    - **Premise:** "The player is **missing** the basket and a crowd is in background"
    - **Hypothesis:** "The player is **dunking** the basketball into the net and a crowd is in background"
    - **Model:** Pred: 0 (NEUTRAL)
    - **Analysis:** This is a clear model failure. "Missing" and "dunking" are mutually exclusive actions. The model should have identified this as a **Contradiction (2)** but defaulted to Neutral. It failed to capture the semantic opposition. The same failure is seen in the "There is no man

dunking..." example.

3. **Missed Hypernyms (Predicting Neutral, True is Entailment):**
    - **Premise:** "Two people are **kickboxing** and spectators are watching"
    - **Hypothesis:** "Two people are **fighting** and spectators are watching"
    - **Model:** Pred: 0 (NEUTRAL)
    - **Analysis:** This is another clear failure. "Kickboxing" is a *type* of "fighting." The hypothesis is a more general, true statement based on the premise. This should be a clear **Entailment (1)**. The model failed to understand this "is-a" relationship.
4. **Data Ambiguity (Predicting Entailment, True is Neutral):**
    - **Premise:** "A little girl is looking at a **woman** in costume"
    - **Hypothesis:** "The little girl is looking at a **man** in costume"
    - **Model:** Pred: 1 (ENTAILMENT)
    - **Analysis:** This is a very difficult example. The model's prediction is wrong, but the *true label* of **Neutral (0)** is also debatable. "Woman" vs. "Man" is a logical contradiction. This points to potential noise and ambiguity in the dataset itself, which makes the model's task even harder.

Strategies for Improvement

Data Augmentation for Hard Negatives:

- **Problem:** The model is weak on specific logical cases (negation, hypernyms).
- **Solution:** I would **augment the training data** with more hard examples.
    - **Contradictions:** Automatically create new contradiction pairs by taking existing entailment pairs and swapping keywords (e.g., "woman" -> "man", "inside" -> "outside", "missing" -> "scoring").
    - **Entailment:** Use a thesaurus or word-hierarchy tool (like WordNet) to find hypernyms (e.g., "kickboxing" -> "fighting", "poodle" -> "dog") and create new entailment pairs.

Switch to a Cross-Encoder Architecture:

- **Problem:** My current Siamese (Bi-Encoder) model analyzes the premise and hypothesis separately, only comparing their final vectors. This makes it "blind" to the direct word-to-word interactions like "missing" vs. "dunking."
- **Solution:** I would switch to the **Cross-Encoder** (MultiLabelModel) architecture. This model feeds both sentences into BERT at the same time ([CLS] premise [SEP] hypothesis [SEP]). This allows BERT's attention mechanism to directly compare the tokens and learn these critical contradictory or entailment relationships. This architecture is slower but almost always more accurate.

4. Anything that can strengthen your report. (5%)

Answer: **Model Refinement and Architecture Justification**

During initial experiments, it became clear that the two sub-tasks were not of equal difficulty. The model consistently learned the classification task (Accuracy) quickly but struggled with or ignored the regression task (Pearson Correlation). To achieve our final results, two key refinements were necessary.

Loss Weighting

We observed that the regression task was significantly harder for the model to learn. When using a simple $total\_loss = loss\_cls + loss\_reg$, the model's performance on the regression task was extremely low (e.g., Pearson ~0.6). The classification loss, being larger or easier to optimize, dominated the training, and the model failed to learn the relatedness score.

To solve this, we introduced a hyperparameter, $regression\_weight$, to amplify the regression loss signal.

$$total\backslash_{loss} = loss_{cls} + loss_{reg} * regression\_weight$$

Our grid search confirmed this was critical. A small weight (rw=2) had little effect, while a larger weight (rw=10) was required to force the model to pay attention to both tasks. This suggests that fine-tuning a deep, 110M-parameter model like BERT requires a sufficiently strong gradient signal to effectively update its weights for a difficult regression task.

Justification for Siamese Architecture

A simple fine-tuning approach (like using the single [CLS] token from a cross-encoder) is often insufficient for complex regression tasks like semantic relatedness. Based on literature (e.g., Sentence-BERT), we implemented a **Siamese (bi-encoder) architecture** to create a more powerful representation.

- **Richer Features:** Instead of one [CLS] vector, this method creates two independent sentence vectors (vec_a, vec_b) using mean pooling. We then combine them in a way proven to be effective for sentence-pair tasks: torch.cat((vec_a, vec_b, |vec_a - vec_b|), dim=1). This hidden_size * 3 vector provides the final prediction heads with a much richer set of features, allowing it to directly compare the two sentence embeddings. This change was a key factor in achieving a high Pearson score.
- **Fair Model Comparison:** As a secondary benefit, this architecture provided a more level playing field for comparing BERT, RoBERTa, and GPT-2. Since GPT-2 (a decoder) cannot be used as a cross-encoder in the same way, forcing all models into a Siamese structure allowed us to fairly compare the *quality of their sentence representations* for this NLU task.

The screenshot of your testing logs and accuracy. (One Figure only) (10%)