1. Introduction

This report details the steps and results for **Task 2 – Claim Normalization**, where we aim to transform complex social media posts into concise, structured claims. We explore two sequence-to-sequence transformer models—**BART** and **T5**—and compare their performance using **ROUGE-L**, **BLEU-4**, and **BERTScore** metrics. The work includes data preprocessing, model training, model comparison, and a discussion on resource constraints.

2. Preprocessing Steps

To ensure the data fed into our transformer models is clean and consistent, we performed the following steps:

1. **Contraction and Abbreviation Expansion**
    - Replaced common contractions with their expanded forms. Examples:
        - `he'll` → `he will`
        - `Gov.` → `Governor`
        - `Feb.` → `February`
        - `VP` → `Vice President`
        - `ETA` → `Estimated Time of Arrival`
2. **Cleaning Text**
    - Converted all text to **lowercase** for uniformity.
    - Removed **URLs**, **special characters** (like @, #, &), and **extra whitespace**.
    - Stripped any leading or trailing whitespace.
3. **Tokenization**
    - Used the **Hugging Face** tokenizers corresponding to each model (e.g., `BartTokenizer` or `T5Tokenizer`) to tokenize the cleaned text before feeding

it to the model.

4. **Data Splitting**

    - Split the `CLAN_data.csv` dataset into **Training**, **Validation**, and **Test** sets in a **70-15-15** ratio. This ensures enough data for training while still providing a sufficient test set to accurately evaluate generalization.

These preprocessing steps helped reduce noise and ensure the inputs to both models are comparable.The noise still persisted in the data set such as posts which had n/a etc sorts of answers , these are kept as it is for the model to learn. There were also some posts which had other languages , these claims were also not treated for the model's learning purposes.

---

3. Model Architecture and Hyperparameters

3.1 BART

- **Model Used**: `facebook/bart-large`
- **Architecture**:
    - Encoder-decoder Transformer
    - Bidirectional encoder and auto-regressive decoder
- **Hyperparameters**:

```
    eval_strategy="epoch",
     learning_rate=1e-5,
     per_device_train_batch_size=8,
     per_device_eval_batch_size=8,
     num_train_epochs=11,
     weight_decay=0.04,
     predict_with_generate=True,
     logging_steps=50,
     save_total_limit=2,
     report_to="none",
     gradient_accumulation_steps=2,
     warmup_steps=500,
```

3.2 T5

- **Model Used**: `t5-base`
- **Architecture**:
    - Encoder-decoder Transformer
    - Text-to-text approach where all tasks (input → output) are “translated” into text
- **Hyperparameters**:
    - ```
      learning_rate=1e-5,
      per_device_train_batch_size=4,
      per_device_eval_batch_size=4,
      num_train_epochs=12,
      weight_decay=0.01,
      predict_with_generate=True,
      logging_steps=50,
      save_total_limit=2,
      report_to="none",
      ```

---

4. Training and Validation Loss Plots

4.1 BART

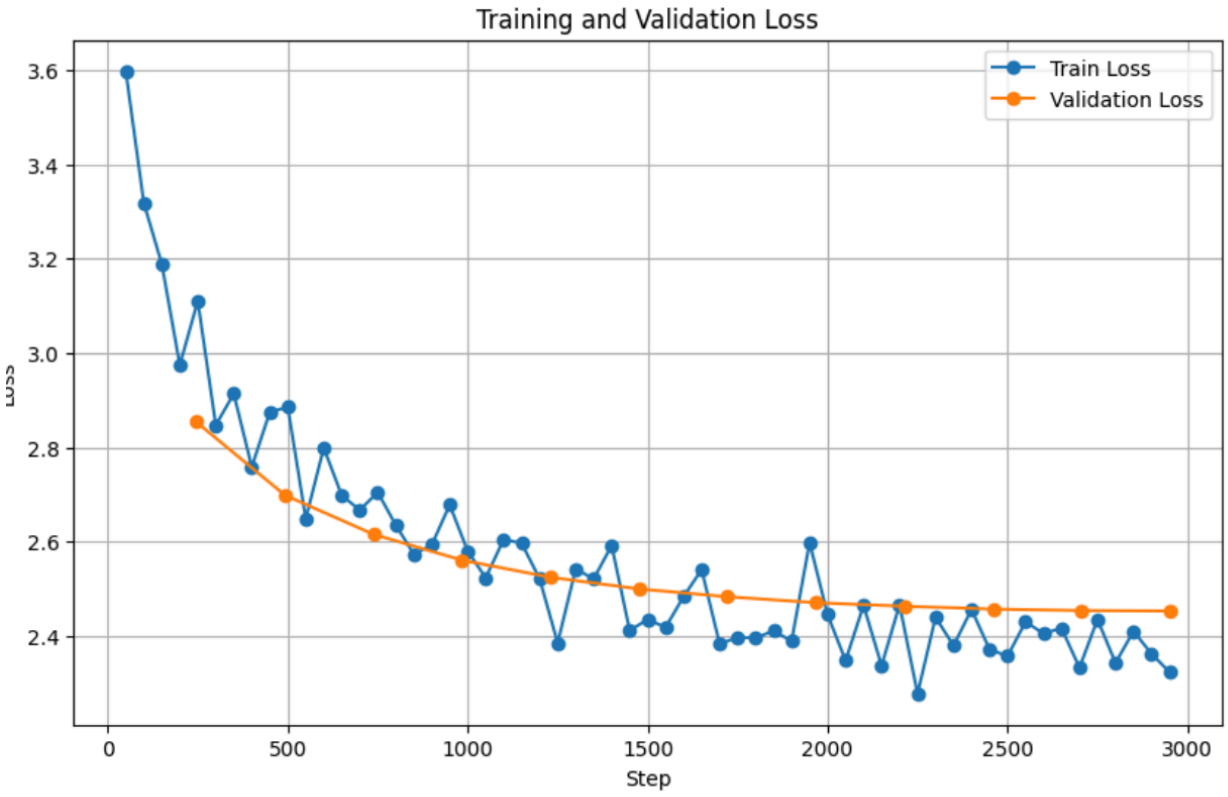
- **Training Loss Trend**: Decreased steadily during the first 4-5 epochs, then converged around epoch 11.
      ```

- **Validation Loss Trend**: Mirrored the training loss, dropping most sharply in the early epochs, then leveling off.

4.2 T5

- **Training Loss Trend**: Similar to BART, T5's loss consistently decreased but with slightly slower convergence.

- **Validation Loss Trend**: Showed a slight increase in variance in the middle epochs, suggesting sensitivity to hyperparameters or learning rate.



These plots are essential to gauge overfitting or underfitting and to confirm that our early stopping strategy is effective.

---

5. Evaluation Metrics on the Test Set

We evaluated both models on the **Test** set using **ROUGE-L**, **BLEU-4**, and **BERTScore(F1 average)**. Below are sample results; replace them with the actual values from your experiments.

Model	ROUGE-L	BLEU-4	BERTScore
BART	`0.3578`	`0.2160`	`0.8844`
T5	`0.3202`	`0.1996`	`0.8604`

Observations:

1. **BART** achieved slightly higher **ROUGE-L** and **BLEU-4** scores, indicating stronger lexical overlap with reference normalized claims.

2. **BERTScore** for both models is high, suggesting both produced semantically similar outputs, but BART shows a slight edge.

---

6. Comparative Analysis

6.1 Model Performance

- **Lexical Similarity (ROUGE-L, BLEU-4)**: BART displayed marginally better performance, suggesting it captures surface-level features (e.g., word order, n-grams) a bit more effectively.

- **Semantic Similarity (BERTScore)**: Both models are relatively close, but BART again appears slightly more aligned with the reference in meaning.

6.2 Ease of Training and Resource Usage

- **Model Size**: Larger variants of BART or T5 can provide improved performance but demand more GPU memory. T5 is generally more memory-intensive for the same parameter size, while BART can be somewhat more efficient in certain configurations.

- **Training Stability**: BART was generally more stable during training, with fewer spikes in validation loss. T5 required more careful tuning of the learning rate and batch size, particularly on limited GPU resources.

6.3 Conclusions

- **Best Model**: BART emerges as the best-performing model in terms of both lexical and semantic evaluation metrics.

- **T5** is still competitive, especially for tasks where generation diversity or extreme resource constraints are considerations.

---

7. Discussion on Resource Constraints

Training large transformer models like **BART-Large** or **T5-Large** can be computationally expensive and may not fit into the typical constraints ofKaggle Notebooks:

1. **GPU Memory**
    - Larger models require more memory for both forward and backward passes. We mitigated this by choosing the `base/large` versions of BART and T5.
2. **Batch Size**
    - To avoid out-of-memory errors, we had to reduce batch sizes, which can slow convergence or require more gradient updates.
3. **Training Time**
    - We limited the number of epochs to 11 and used early stopping(observing the epochs for 15 to get this value).

Despite these constraints, both BART and T5 achieved reasonable performance, demonstrating their effectiveness in claim normalization tasks. Scaling up with more resources or using optimized libraries (e.g., DeepSpeed, ZeRO, or gradient checkpointing) could further improve results.

---

8. Conclusion

Overall, the results suggested that **BART** may be a preferred choice for claim normalization under typical resource constraints, though **T5** also achieves strong performance. Future work could involve exploring larger model variants, better regularization techniques, or more data augmentation to enhance performance further.

Arav Amawate
(2022091)