Email Subject Line Generation

Model Selection

I picked **T5-small** for this task instead of larger models like LLaMa for a few key reasons:

1. **Speed and resources**: No need for heavy computation like fine tuning a large model when a small model should be able to do just fine.
2. **Task simplicity**: Creating email subject from body is basically a simple summarization task. Big models would be overkill.
3. **Clear improvements**: Using a smaller base model lets us easily see how much fine-tuning helps as bigger billion param models are already good at such tasks.

This decision paid off. The original T5-small had zero exact matches on test data, which made our improvements crystal clear.

Data Preprocessing

I worked with the Enron Email Dataset (this one seemed more appropriate) and processed it like this:

1. Extracted subject-body pairs from raw emails
2. Stripped out "Re:", "RE:", "FW:", and "Fwd:" prefixes from subjects
3. Kept only emails with subjects between 20-60 characters and bodies over 50 characters
    - Too short or too long examples may confuse the LLM, better keep the behavior slightly predictable
    - This filtering kept about 53% of the original data
4. Added a "summarize: " prefix to each email body as input

For faster testing, I used a small subset (4,099 emails) split into train/validation/test sets. Two line uncomment will use all of the filtered data.

Training

My fine-tuning setup was straightforward:

- 5 epochs with batch size of 16
- Learning rate of $5e^{-5}$
- Max input length: 512 tokens (this can deteriorate performance a bit), max output: 32 tokens
- FP16 precision for speed
- ROUGE-L and exact match metrics for evaluation

Results

The fine-tuning made a huge difference:

Metric	Base T5	Fine-tuned T5	Improvement
ROUGE-L	0.10	0.54	+0.44
Exact Match	0.00	0.32	+0.32

The pre-trained model couldn't generate a single matching subject line. After fine-tuning,we see a great improvement.

Challenges & Observations

The trickiest part was getting the evaluation metrics right. Hugging Face's Trainer passes predictions in a weird format that needs careful handling before computing scores (Spoiler: used Claude 3.7 Sonnet for designing the compute_metrics function).

I also noticed that results converged around epoch 3. While 32% exact matches is good, there's room to do better with more data training

This project shows we don't need huge models for practical tasks. A well-tuned small model can be surprisingly effective for specialized jobs.

The notebook was created and ran using Kaggle and its P100 GPU. The total notebook takes **around 17 minutes** to run.