Training a Transformer on Canterbury and Victorian Play

The Build Fellowship

September 2, 2025

Sabarish Raja Ramesh Raja

Motivation and Goal

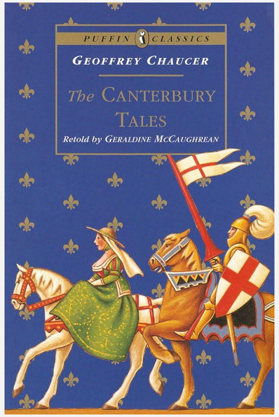
- Build a Transformer model from scratch.

- Deliverable:

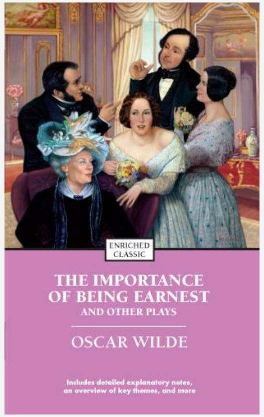
  Two distinct generations from the model

  1. Style 0: Canterbury Medieval Tone
  2. Style 1: Earnest witty Victorian Dialogue (from The Importance of Being Earnest)

Data Preparation

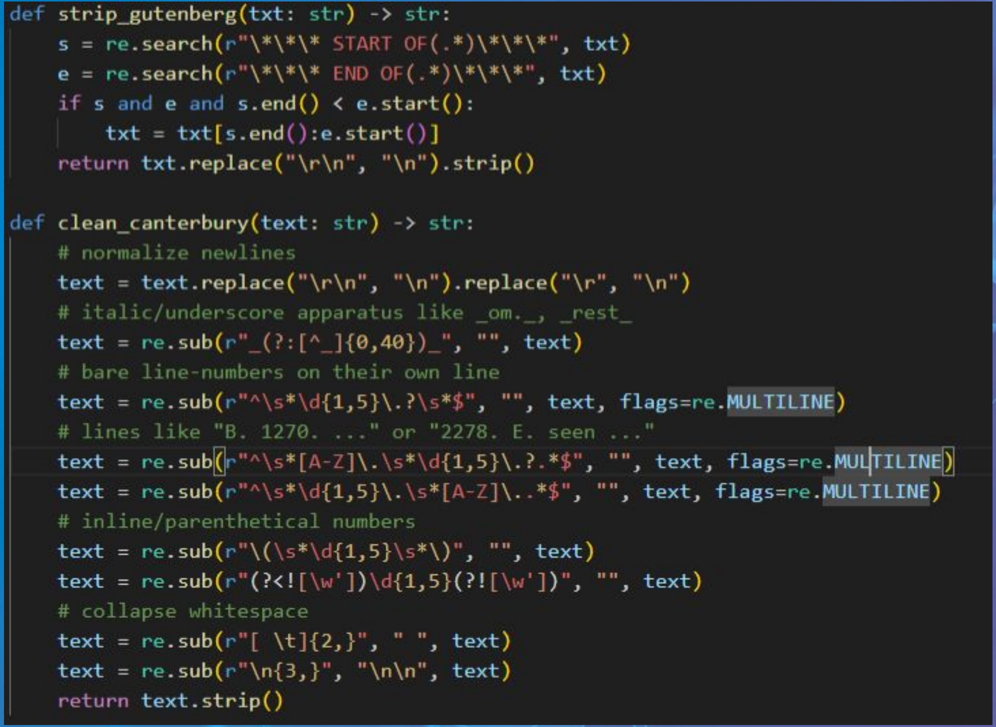


This povre widwe awaiteth al that night
After hir litel child, but he cam noght;
For which, as sone as it was dayes light,
With face pale of drede and bisy thoght,
She hath at scole and elles-wher him soght,
Til finally she gan so fer espye
That he last seyn was in the lewerye.


ENRICHED CLASSIC
THE IMPORTANCE OF BEING EARNEST
AND OTHER PLAYS
OSCAR WILDE
Includes detailed explanatory notes, an overview of key themes, and more

ALGERNON.
Did you hear what I was playing, Lane?
LANE.
I didn't think it polite to listen, sir.
ALGERNON.
I'm sorry for that, for your sake. I don't play accurately—any one can play accurately—but I play with wonderful expression.

Using Regex to pre-process



```python
def strip_gutenberg(txt: str) -> str:
    s = re.search(r"\*\*\* START OF(.*)\*\*\*", txt)
    e = re.search(r"\*\*\* END OF(.*)\*\*\*", txt)
    if s and e and s.end() < e.start():
        txt = txt[s.end():e.start()]
    return txt.replace("\r\n", "\n").strip()

def clean_canterbury(text: str) -> str:
    # normalize newlines
    text = text.replace("\r\n", "\n").replace("\r", "\n")
    # italic/underscore apparatus like _om._, _rest_
    text = re.sub(r"_(?:[^_]{0,40})_", "", text)
    # bare line-numbers on their own line
    text = re.sub(r"^\s*\d{1,5}\.?\s*$", "", text, flags=re.MULTILINE)
    # lines like "B. 1270. ..." or "2278. E. seen ..."
    text = re.sub(r"^\s*[A-Z]\.\s*\d{1,5}\.?.*$", "", text, flags=re.MULTILINE)
    text = re.sub(r"^\s*\d{1,5}\.\s*[A-Z]\..*$", "", text, flags=re.MULTILINE)
    # inline/parenthetical numbers
    text = re.sub(r"\(\s*\d{1,5}\s*\)", "", text)
    text = re.sub(r"(?<![\w'])\d{1,5}(?![\w'])", "", text)
    # collapse whitespace
    text = re.sub(r"[ \t]{2,}", " ", text)
    text = re.sub(r"\n{3,}", "\n\n", text)
    return text.strip()
```

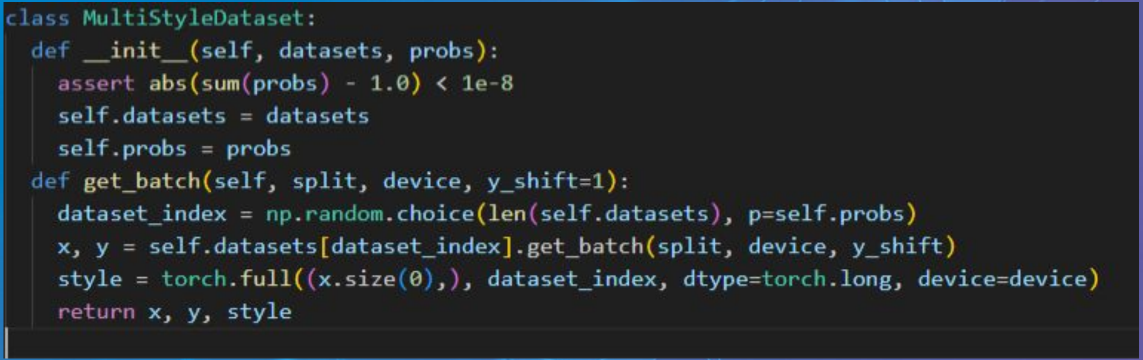
Model Architecture:

Transformer:

1. n_embd=384, n_layer=6, n_head=6, dropout=0.2, context=256
2. Pre-Norm residual: x = x + SA(LN(x)); x = x + FFN(LN(x))
3. Feed-Forward: 4× expansion + ReLU + dropout (simple, fast)

Training Setup:

1. MultiStyleDataset mixes corpora with ratios [0.2, 0.8]
2. Optimizer: AdamW(lr=5e-5)

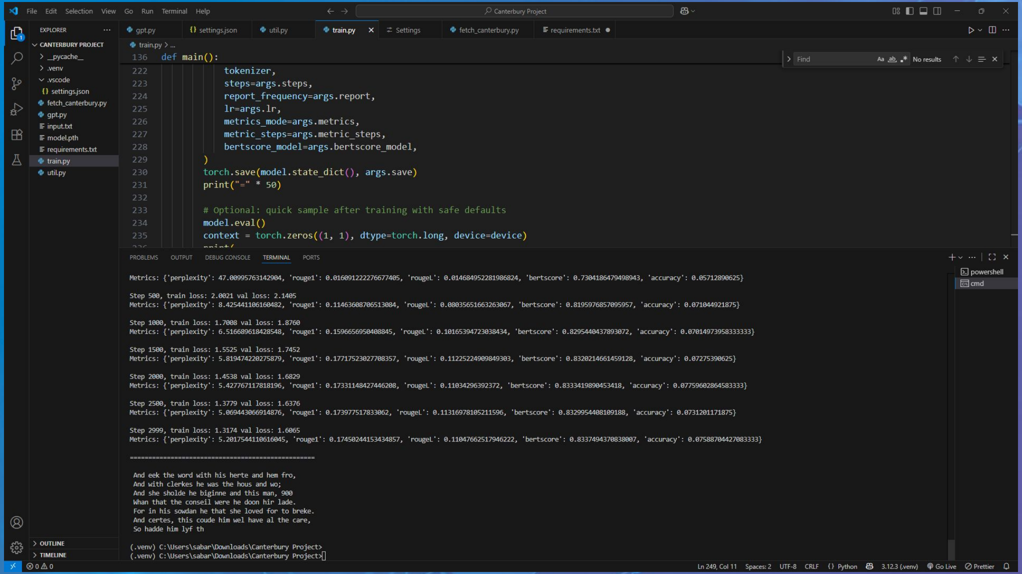


```
class MultiStyleDataset:
  def __init__(self, datasets, probs):
    assert abs(sum(probs) - 1.0) < 1e-8
    self.datasets = datasets
    self.probs = probs
  def get_batch(self, split, device, y_shift=1):
    dataset_index = np.random.choice(len(self.datasets), p=self.probs)
    x, y = self.datasets[dataset_index].get_batch(split, device, y_shift)
    style = torch.full((x.size(0),), dataset_index, dtype=torch.long, device=device)
    return x, y, style
```

Metrics and Evaluation:

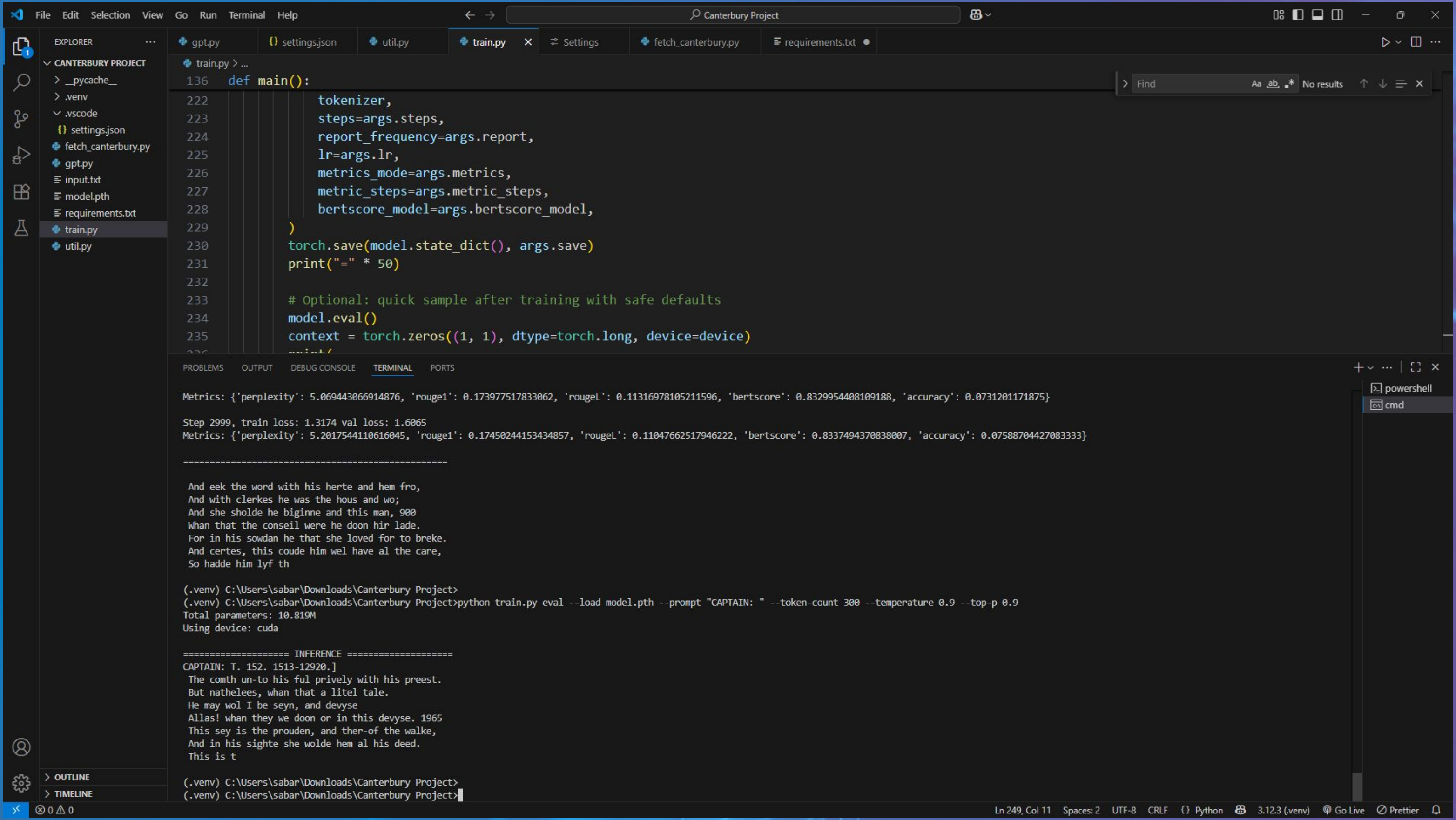
Metrics used are rogue, bertscore and accuracy that were updated in each training step.

1. Perplexity => Model's ability to predict the next word.
2. Rogue-1 => Measures the overlap of unigrams.
3. Rogue-L => Longest common subsequence overlap.
4. BERTScore => Semantic similarity



```python
def main():
            tokenizer,
            steps=args.steps,
            report_frequency=args.report,
            lr=args.lr,
            metrics_mode=args.metrics,
            metric_steps=args.metric_steps,
            bertscore_model=args.bertscore_model,
        )
        torch.save(model.state_dict(), args.save)
        print("=" * 50)

        # Optional: quick sample after training with safe defaults
        model.eval()
        context = torch.zeros((1, 1), dtype=torch.long, device=device)
```

```
Metrics: {'perplexity': 47.00995763142904, 'rouge1': 0.016091222276677405, 'rougeL': 0.014684952281986824, 'bertscore': 0.7304186479498943, 'accuracy': 0.05712890625}

Step 500, train loss: 2.0021 val loss: 2.1405
Metrics: {'perplexity': 8.425441106160482, 'rouge1': 0.11463608706513084, 'rougeL': 0.08035651663263067, 'bertscore': 0.8195976857095957, 'accuracy': 0.071044921875}

Step 1000, train loss: 1.7008 val loss: 1.8760
Metrics: {'perplexity': 6.516689618428548, 'rouge1': 0.1596656950408845, 'rougeL': 0.10165394723038434, 'bertscore': 0.8295440437893072, 'accuracy': 0.07014973958333333}

Step 1500, train loss: 1.5525 val loss: 1.7452
Metrics: {'perplexity': 5.819474220275879, 'rouge1': 0.17717523027708357, 'rougeL': 0.11225224909849303, 'bertscore': 0.8320214661459128, 'accuracy': 0.07275390625}

Step 2000, train loss: 1.4538 val loss: 1.6829
Metrics: {'perplexity': 5.427767117818196, 'rouge1': 0.17331148427446208, 'rougeL': 0.11034296392372, 'bertscore': 0.8333419890453418, 'accuracy': 0.07759602864583333}

Step 2500, train loss: 1.3779 val loss: 1.6376
Metrics: {'perplexity': 5.069443066914876, 'rouge1': 0.173977517833062, 'rougeL': 0.11316978105211596, 'bertscore': 0.8329954408109188, 'accuracy': 0.0731201171875}

Step 2999, train loss: 1.3174 val loss: 1.6065
Metrics: {'perplexity': 5.2017544110616045, 'rouge1': 0.1745024415343857, 'rougeL': 0.1104766251794622, 'bertscore': 0.8337494370838007, 'accuracy': 0.07588704427083333}

==================================================

 And eek the word with his herte and hem fro,
 And with clerkes he was the hous and wo;
 And she sholde he biginne and this man, 900
 Whan that the conseil were he doon hir lade.
 For in his sowdan he that she loved for to breke.
 And certes, this coude him wel have al the care,
 So hadde him lyf th

(.venv) C:\Users\sabar\Downloads\Canterbury Project>
(.venv) C:\Users\sabar\Downloads\Canterbury Project>
```

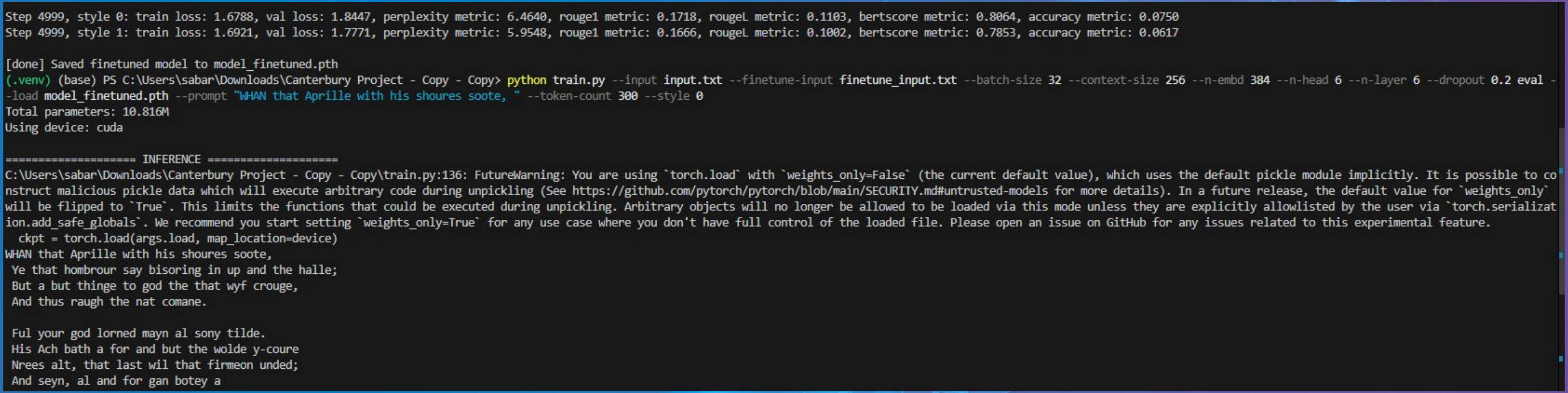


```python
def main():
            tokenizer,
            steps=args.steps,
            report_frequency=args.report,
            lr=args.lr,
            metrics_mode=args.metrics,
            metric_steps=args.metric_steps,
            bertscore_model=args.bertscore_model,
        )
        torch.save(model.state_dict(), args.save)
        print("=" * 50)

        # Optional: quick sample after training with safe defaults
        model.eval()
        context = torch.zeros((1, 1), dtype=torch.long, device=device)
```

```
Metrics: {'perplexity': 5.069443066914876, 'rouge1': 0.173977517833062, 'rougeL': 0.11316978105211596, 'bertscore': 0.8329954408109188, 'accuracy': 0.0731201171875}

Step 2999, train loss: 1.3174 val loss: 1.6065
Metrics: {'perplexity': 5.2017544110616045, 'rouge1': 0.17450244153434857, 'rougeL': 0.11047662517946222, 'bertscore': 0.8337494370838007, 'accuracy': 0.07588704427083333}

==================================================

 And eek the word with his herte and hem fro,
 And with clerkes he was the hous and wo;
 And she sholde he biginne and this man, 900
 Whan that the conseil were he doon hir lade.
 For in his sowdan he that she loved for to breke.
 And certes, this coude him wel have al the care,
 So hadde him lyf th

(.venv) C:\Users\sabar\Downloads\Canterbury Project>
(.venv) C:\Users\sabar\Downloads\Canterbury Project>python train.py eval --load model.pth --prompt "CAPTAIN: " --token-count 300 --temperature 0.9 --top-p 0.9
Total parameters: 10.819M
Using device: cuda

==================== INFERENCE ====================
CAPTAIN: T. 152. 1513-12920.]
 The comth un-to his ful prively with his preest.
 But nathelees, whan that a litel tale.
 He may wol I be seyn, and devyse
 Allas! whan they we doon or in this devyse. 1965
 This sey is the prouden, and ther-of the walke,
 And in his sighte she wolde hem al his deed.
 This is t

(.venv) C:\Users\sabar\Downloads\Canterbury Project>
(.venv) C:\Users\sabar\Downloads\Canterbury Project>
```

Results

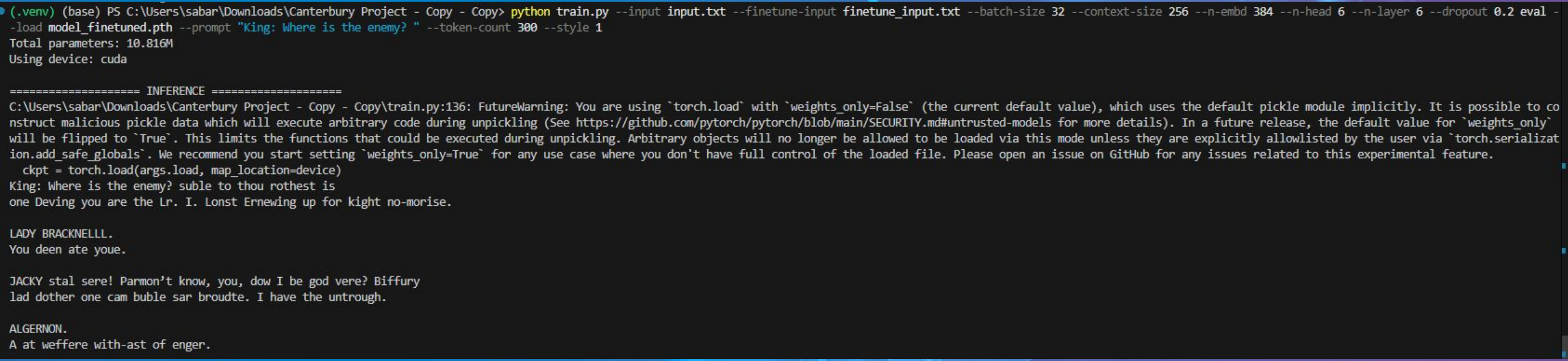
Style 0: Canterbury Medieval Text Generation



```
Step 4999, style 0: train loss: 1.6788, val loss: 1.8447, perplexity metric: 6.4640, rouge1 metric: 0.1718, rougeL metric: 0.1103, bertscore metric: 0.8064, accuracy metric: 0.0750
Step 4999, style 1: train loss: 1.6921, val loss: 1.7771, perplexity metric: 5.9548, rouge1 metric: 0.1666, rougeL metric: 0.1002, bertscore metric: 0.7853, accuracy metric: 0.0617

[done] Saved finetuned model to model_finetuned.pth
(.venv) (base) PS C:\Users\sabar\Downloads\Canterbury Project - Copy - Copy> python train.py --input input.txt --finetune-input finetune_input.txt --batch-size 32 --context-size 256 --n-embd 384 --n-head 6 --n-layer 6 --dropout 0.2 eval -
-load model_finetuned.pth --prompt "WHAN that Aprille with his shoures soote, " --token-count 300 --style 0
Total parameters: 10.816M
Using device: cuda

==================== INFERENCE ====================
C:\Users\sabar\Downloads\Canterbury Project - Copy - Copy\train.py:136: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to co
nstruct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only`
will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serializat
ion.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  ckpt = torch.load(args.load, map_location=device)
WHAN that Aprille with his shoures soote,
 Ye that hombrour say bisoring in up and the halle;
 But a but thinge to god the that wyf crouge,
 And thus raugh the nat comane.

 Ful your god lorned mayn al sony tilde.
 His Ach bath a for and but the wolde y-coure
 Nrees alt, that last wil that firmeon unded;
 And seyn, al and for gan botey a
```

Results

Style 1: The Importance of Being Earnest Text Generation



```
(.venv) (base) PS C:\Users\sabar\Downloads\Canterbury Project - Copy - Copy> python train.py --input input.txt --finetune-input finetune_input.txt --batch-size 32 --context-size 256 --n-embd 384 --n-head 6 --n-layer 6 --dropout 0.2 eval -
-load model_finetuned.pth --prompt "King: Where is the enemy? " --token-count 300 --style 1
Total parameters: 10.816M
Using device: cuda

==================== INFERENCE ====================
C:\Users\sabar\Downloads\Canterbury Project - Copy - Copy\train.py:136: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to co
nstruct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only`
will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serializat
ion.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
  ckpt = torch.load(args.load, map_location=device)
King: Where is the enemy? suble to thou rothest is
one Deving you are the Lr. I. Lonst Ernewing up for kight no-morise.

LADY BRACKNELLL.
You deen ate youe.

JACKY stal sere! Parmon’t know, you, dow I be god vere? Biffury
lad dother one cam buble sar broudte. I have the untrough.

ALGERNON.
A at weffere with-ast of enger.
```