

Structural Line Markers and Multi-Pass Reranking for GPT-2 Sonnet Generation

Stanford CS224N Default Project

Aalaap Hegde

Department of Computer Science
Stanford University
aalaaph@stanford.edu

Mudit Baid

Department of Computer Science
Stanford University
muditb@stanford.edu

Rakshit Kaushik

Department of Computer Science
Stanford University
rakshitk@stanford.edu

Mentor: Qinan Yu

Abstract

We implement GPT-2 from scratch and fine-tune it on two downstream tasks: paraphrase detection over the Quora Question Pairs dataset, and Shakespearean sonnet completion. Paraphrase detection is framed as a cloze-style next-token prediction problem, where the model predicts “yes” or “no” after a natural language prompt. We improve upon the baseline with domain-adaptive pre-training on Quora questions, SMART adversarial regularization, symmetric consistency losses that enforce order-invariant predictions, and data augmentation, achieving a test accuracy of **90.3%** (up from a 86.4% baseline) and dev accuracy of **90.7%**. For sonnet generation, the core challenge is learning 14-line poetic structure from only 143 training examples. Our key contribution is *structural line markers*: explicit positional tags (L01: through L14:) prepended to each line during training, which enable line-by-line generation with guaranteed structural validity. Building on this, we introduce *line-level reranking*, a multi-pass optimization procedure that assembles an optimal sonnet by greedily substituting individual lines from a pool of diverse candidates to maximize chrF. Together, these techniques yield a test chrF of **43.563** and test chrF of **44.69**. We find that line markers are the single most important design decision, and that methods which increase candidate diversity (such as multi-temperature sampling) are more valuable than those that constrain the output space (such as vocabulary biasing).

1 Introduction

Large language models trained on broad internet corpora encode substantial linguistic knowledge, yet adapting them to specialized tasks remains nontrivial. In this project, we implement GPT-2 [1] from scratch and explore how a single pretrained decoder can be repurposed for two fundamentally different problems: binary classification (paraphrase detection) and constrained creative generation (sonnet completion).

Paraphrase detection asks whether two sentences express the same meaning. The standard approach would add a classification head on top of the language model, but the cloze-style formulation [2] offers an appealing alternative. By framing the task as “Is s_1 a paraphrase of s_2 ? Answer yes or no:”, we leverage GPT-2’s next-token prediction capability directly. This avoids introducing new parameters and lets the model exploit its pretrained understanding of natural language inference. The difficulty lies in making the model robust: paraphrase is a symmetric relation, but the cloze prompt

imposes an asymmetric left-to-right ordering. We address this with dual-perspective inference and consistency regularization.

Sonnet generation presents a different set of challenges. Given the first three lines of a held-out Shakespeare sonnet, the model must produce the remaining eleven. Shakespeare’s sonnets follow a rigid ABAB CDCD EFEF GG rhyme scheme across exactly 14 lines, and the training corpus contains only 143 examples. GPT-2’s byte-pair encoding was designed for modern English, and only 6.6% of its 50,257 tokens appear in the Shakespeare training data. A naive fine-tuned model treats sonnets as flat token streams and has no explicit awareness of line boundaries. This frequently produces output with the wrong number of lines or malformed structure.

Our central insight for sonnet generation is that *structural line markers* solve the line-counting problem. By prepending positional tags (L01: through L14:) to each line during training, we give the model explicit awareness of where it is within the 14-line structure. At generation time, we prompt the model with successive line markers and generate one line at a time, guaranteeing exactly 14 lines of output. This simple modification unlocks a second technique: *line-level reranking*. Because every candidate sonnet produced with markers is structurally valid, individual lines become interchangeable across candidates. We generate 50 candidates at varying temperatures and then greedily substitute lines to maximize chrF against the training corpus.

For paraphrase detection, we implement eight extensions including domain-adaptive pre-training, SMART adversarial regularization [3], and symmetric consistency losses, lifting dev accuracy from 86.4% to 90.7%. For sonnet generation, the combination of line markers, GPT-2 medium, and line-level reranking yields our best dev chrF of 44.69, up from 41.7 with an unmarked GPT-2 small baseline.

2 Related Work

GPT-2 and Autoregressive Language Models. Radford et al. [1] introduced GPT-2, a transformer decoder trained on WebText with up to 1.5B parameters. The model uses byte-pair encoding from Sennrich et al. [4] and learned positional embeddings. Our implementation follows the GPT-2 small (124M) and medium (355M) configurations, loading pretrained weights from HuggingFace.

Cloze-Style Classification. Schick and Schütze [2] showed that language models can perform classification by mapping labels to tokens and using the model’s next-token prediction. This pattern-exploiting approach avoids adding task-specific parameters and can be especially effective in low-data regimes. We adopt this formulation for paraphrase detection, predicting “yes” (token 8505) or “no” (token 3919).

Adversarial Regularization. Jiang et al. [3] proposed SMART, a regularization technique that finds adversarial perturbations of input embeddings and penalizes the model for being sensitive to them. This encourages smooth decision boundaries and has been shown to improve robustness in fine-tuning. We apply SMART to the paraphrase detection task, computing symmetric KL divergence between clean and adversarially perturbed predictions.

Decoding Strategies. Holtzman et al. [5] identified the degeneration problem in open-ended text generation and proposed nucleus sampling, which samples from the smallest set of tokens whose cumulative probability exceeds a threshold p . Su et al. [6] proposed contrastive search as an alternative that balances likelihood with distinctiveness from the existing context. We experiment with both strategies for sonnet generation and find nucleus sampling more suitable for creative text.

Poetry Generation. Neural poetry generation has been explored with RNNs by Hopkins and Kiela [7] and with transformers by Liao et al. [8]. The primary challenges are maintaining structural constraints (meter, rhyme) while preserving semantic coherence. Our line-marker approach is related to positional encoding strategies but operates at the line level rather than the token level, providing explicit structural supervision.

3 Approach

3.1 Base Model

We reimplement GPT-2 from scratch, including masked multi-head self-attention with causal masking, pre-norm transformer layers with GELU activations, and weight-tied output projections where logits are computed as $\mathbf{h}\mathbf{E}^\top$ for hidden state $\mathbf{h}$ and embedding matrix $\mathbf{E}$. We also implement AdamW [9] with an efficient bias-corrected learning rate:

$$\alpha_t = \alpha \cdot \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} \quad (1)$$

where α is the base learning rate and β_1, β_2 are the momentum coefficients.

3.2 Paraphrase Detection

Cloze-Style Formulation. Given sentences s_1 and s_2 , we construct the prompt: “Is “ s_1 ” a paraphrase of “ s_2 ”? Answer “yes” or “no”: ”. The model’s prediction is the argmax over the full vocabulary logits at the last token position. We train with cross-entropy loss on the target token IDs (8505 for “yes”, 3919 for “no”).

Domain-Adaptive Pre-Training. Before cloze fine-tuning, we run next-token prediction on all unique questions from the Quora training set for 2 epochs. This adapts GPT-2’s representations to the question domain without requiring labels.

Dual Perspective and Consistency Loss. Paraphrase is symmetric: if s_1 paraphrases s_2 , then s_2 paraphrases s_1 . At inference, we average logits from both orderings (dual-perspective inference). During training, we add a symmetric KL divergence loss between forward and reverse predictions:

$$\mathcal{L}_{\text{cons}} = \frac{1}{2} \left(D_{\text{KL}}(\mathbf{p}_{\text{fwd}} \parallel \mathbf{p}_{\text{rev}}) + D_{\text{KL}}(\mathbf{p}_{\text{rev}} \parallel \mathbf{p}_{\text{fwd}}) \right) \quad (2)$$

where $\mathbf{p}_{\text{fwd}}$ and $\mathbf{p}_{\text{rev}}$ are the softmax distributions over {yes, no} for the (s_1, s_2) and (s_2, s_1) orderings respectively.

SMART Adversarial Regularization. Following Jiang et al. [3], we find the adversarial perturbation δ^* that maximizes the KL divergence between clean and perturbed predictions:

$$\delta^* = \epsilon \cdot \frac{\nabla_{\delta} D_{\text{KL}}(f(\mathbf{x} + \delta) \parallel f(\mathbf{x}))}{\|\nabla_{\delta} D_{\text{KL}}(f(\mathbf{x} + \delta) \parallel f(\mathbf{x}))\|} \quad (3)$$

The SMART loss is the symmetric KL between predictions on clean embeddings $\mathbf{x}$ and adversarially perturbed embeddings $\mathbf{x} + \delta^*$. This encourages locally smooth decision boundaries.

Additional Extensions. We apply several other techniques. Data augmentation doubles the training set by swapping sentence order, directly addressing the asymmetry of the cloze prompt. A cosine learning rate schedule with linear warmup prevents catastrophic forgetting of pretrained weights. Gradient accumulation increases the effective batch size, and weight decay is applied selectively, excluding bias and LayerNorm parameters. Finally, we clean unicode artifacts and fix tokenization issues in the Quora text.

3.3 Sonnet Generation

Training Regularization. Fine-tuning on 143 sonnets is an extreme low-data regime, so we apply several regularization techniques to prevent memorization. Label smoothing softens the target distribution, discouraging the model from assigning all probability mass to a single next token. A cosine learning rate schedule with linear warmup avoids the sharp initial gradient updates that can destabilize fine-tuning from pretrained weights. Gradient clipping and weight decay provide additional stability, and early stopping prevents overfitting to the small training set. Specific hyperparameter values are reported in Section 4.3.

Line Markers. Our key structural contribution. Standard GPT-2 treats sonnets as flat token streams and has no awareness of line boundaries. We prepend positional tags to each line during training:

```
L01: From fairest creatures we desire increase,  
L02: That thereby beauty's rose might never die,  
...  
L14: His tender heir might bear his memory:
```

At generation time, given the first 3 lines as a prompt (with markers L01: through L03:), we generate one line at a time by appending `\nL04:` and sampling until a newline token, then appending `\nL05:`, and so on through L14:. This guarantees exactly 14 lines and gives the model explicit positional information about sonnet structure. The markers also provide implicit couplet awareness: the model learns from training data that L13: and L14: consistently begin with two spaces (the indented couplet), so it reproduces this formatting automatically. After generation, we strip the markers using a regex pass.

Generation and Decoding. We use nucleus sampling [5] with temperature τ and threshold p . A systematic hyperparameter sweep over $\tau \in [0.7, 1.0]$, $p \in [0.8, 0.95]$, and repetition penalty $\in [1.0, 1.2]$ evaluates each combination with 5 random seeds to account for sampling variance. The sweep identified $\tau = 0.85$, $p = 0.95$ as optimal. Repetition penalty did not help, likely because line markers already prevent the model from producing degenerate loops. We also implemented contrastive search [6], which selects tokens that balance model confidence with distinctiveness from preceding context. However, this strategy scored 37.81 chrF compared to 42.08 for nucleus sampling under identical conditions; the degeneration penalty inherent in contrastive search conflicts with poetry’s reliance on thematic repetition and consistent register.

Multi-Temperature Candidate Generation. Rather than generating all candidates at a single temperature, we distribute them across 5 temperatures ($\{0.75, 0.80, 0.85, 0.90, 0.95\}$), producing 10 candidates at each. Lower temperatures yield more conservative, high-likelihood lines that closely match training patterns. Higher temperatures produce more varied and occasionally surprising word choices. This diversity is critical for reranking: if all candidates come from the same temperature, the candidate pool contains many near-duplicate lines, limiting the optimization ceiling.

Line-Level Reranking. Standard reranking generates N candidate sonnets and selects the one with the highest chrF. This is limited because a single weak line anywhere drags down the score of an otherwise strong candidate. Our line-level reranking decomposes the problem. Before scoring, we filter candidates through a coherence gate that rejects sonnets with the wrong number of lines, non-ASCII characters (a sign of GPT-2’s modern vocabulary leaking through), or lines shorter than 10 or longer than 100 characters. We then select the best full candidate as a starting point and run a multi-pass greedy optimization:

1. For each line position 4 through 14, try substituting every unique line from the candidate pool at that position. Keep the substitution that maximizes the full sonnet’s chrF against the training corpus.
2. Repeat for up to 3 passes, since changing one line affects the optimal choice for adjacent lines.

This procedure is only possible because line markers guarantee structural validity of all candidates, making any line a valid substitution at any position.

Additional Extensions. We also explored vocabulary biasing and rhyme scoring. A logit bias tensor built from Shakespeare’s token frequency distribution slightly helped standard reranking (+0.17 chrF) but hurt line-level reranking (−0.51 chrF) by reducing candidate diversity. A suffix-based rhyme scorer evaluating adherence to the ABAB CDCD EFEF GG scheme was not used in our final reranking objective, but served as a diagnostic that confirmed line markers improve rhyme consistency over unmarked generation.

4 Experiments

4.1 Data

Paraphrase Detection. The Quora Question Pairs dataset contains labeled sentence pairs indicating whether two questions are paraphrases. We use the provided train/dev/test splits. With data augmentation (swapping s_1, s_2), the training set approximately doubles.

Sonnet Generation. The training set consists of 143 Shakespeare sonnets (`sonnets.txt`). The dev set contains 12 sonnets (sonnets 132–143) with only the first 3 lines provided; the model must generate the remaining 11. A separate held-out test set is used for the leaderboard.

4.2 Evaluation Metrics

Paraphrase Detection. Accuracy on the dev/test sets, computed by comparing the model’s argmax prediction (over the full vocabulary) to the target token ID.

Sonnet Generation. ChrF [10] (character F-score with $\beta = 2$) as implemented in `sacrebleu`. ChrF computes character-level n-gram precision and recall between generated and reference text, with $\beta = 2$ weighting recall twice as heavily as precision. Whitespace is removed before scoring.

4.3 Experimental Details

All experiments use our from-scratch GPT-2 implementation with pretrained weights loaded from HuggingFace. Training was done on a single NVIDIA L4 GPU (24GB).

Paraphrase Detection. GPT-2 small (124M parameters, 12 layers, 768 hidden, 12 heads). Learning rate 1×10^{-5} with cosine decay and 10% linear warmup over total training steps. Batch size 8 with gradient accumulation over 4 steps (effective batch size 32). Weight decay 0.01, applied only to non-bias, non-LayerNorm parameters. Consistency loss weight $\lambda_{\text{cons}} = 0.5$. SMART regularization weight $\lambda_{\text{smart}} = 0.1$ with perturbation magnitude $\epsilon = 10^{-5}$ and step size 10^{-3} . Domain-adaptive pre-training runs for 2 epochs on all unique Quora questions before cloze fine-tuning begins. The total paraphrase pipeline trains for 10 epochs.

Sonnet Generation. GPT-2 medium (355M parameters, 24 layers, 1024 hidden, 16 heads). Learning rate 1×10^{-5} with cosine warmup schedule (10% warmup steps, minimum LR ratio 0.1). Batch size 8. Label smoothing $\alpha = 0.1$. Gradient clipping at max norm 1.0. Weight decay 0.01. Early stopping with patience 5 and minimum improvement threshold $\delta = 0.001$ on training loss. Training runs for up to 25 epochs. For generation: nucleus sampling with $\tau = 0.85$, $p = 0.95$, no repetition penalty. For line-level reranking: 50 candidates generated across 5 temperatures $\{0.75, 0.80, 0.85, 0.90, 0.95\}$ (10 per temperature), followed by 3 greedy optimization passes. To quantify generation variance, we ran single-shot generation with 5 random seeds; the mean chrF was 42.11 with standard deviation 0.30, confirming that our improvements exceed the noise floor.

4.4 Results

4.4.1 Paraphrase Detection

Table 1 shows the dev accuracy as extensions are added incrementally. Each row includes all extensions from previous rows.

Table 2 shows approaches that did not improve paraphrase detection.

4.4.2 Sonnet Generation

Table 3 summarizes our sonnet generation experiments on the 12-sonnet dev set.

Table 4 summarizes approaches that did not improve performance.

Configuration	Dev Acc. (%)
Baseline (cloze-style, template A, no extensions)	86.4
+ Data augmentation (swap s_1, s_2)	88.7
All extensions combined	90.7

Table 1: Paraphrase detection dev accuracy. The baseline uses vanilla cloze-style fine-tuning with constant learning rate and batch size 8. Data augmentation doubles the training set by swapping sentence order. The full system adds LR warmup/cosine scheduling, gradient accumulation, dual-perspective inference, consistency loss, SMART regularization, domain-adaptive pre-training, and text cleaning.

Configuration	Dev Acc. (%)	Notes
Prompt template D	86.1	Slightly worse than template A

Table 2: Paraphrase detection approaches that did not improve over the baseline.

5 Analysis

5.1 Qualitative Sonnet Evaluation

Table 5 shows a generated sonnet compared to the Shakespeare original. The model captures Shakespearean vocabulary and tone (“thy,” “dost,” “beautified”), and the couplet structure (indented final two lines) is maintained through our generation procedure. However, several failure modes are visible. Lines occasionally devolve into incoherent phrasing, such as “Programmeboards whose sole ornament was hue” in another example, where modern GPT-2 tokens leak through despite fine-tuning. The model also struggles with semantic coherence across lines: while individual lines often sound plausible, the argument of the sonnet frequently loses its thread midway through.

5.2 Sonnet Generation

Line-level reranking is the single largest contributor to our final score, adding +1.28 chrF over standard whole-sonnet reranking. The intuition is straightforward: scoring a full 14-line sonnet averages quality across all lines, so one weak line drags down an otherwise strong candidate. Line-level optimization targets the weakest lines directly, replacing them with better alternatives from the candidate pool. This also explains why standard reranking with $N > 30$ actually *decreased* performance (43.14 for $N = 100$ vs. 43.41 for $N = 30$)—full-sonnet scoring is noisy, and adding weaker candidates dilutes the selection—while line-level reranking scales well because it isolates per-line quality.

The unifying theme across our negative results (Table 4) is that anything reducing candidate diversity hurts line-level reranking. GPT-2 large overfits more aggressively on 143 sonnets, producing less varied output. The ensemble mixes stylistic tendencies across model sizes, so lines from GPT-2 small clash with adjacent lines from GPT-2 large. Vocabulary bias narrows the token distribution, producing homogeneous candidates. In each case, the greedy optimizer simply has fewer distinct lines to work with. Contrastive search failed for a different reason: its degeneration penalty is counterproductive for poetry, where thematic repetition and consistent register are desirable rather than problematic.

Scaling from GPT-2 small to medium helped (+0.68 chrF), but scaling further to large hurt. The hyperparameter sweep also did not improve over default parameters once we had sufficient model capacity. Both observations point to the same conclusion: for this task, the bottleneck is candidate diversity at generation time, not model expressiveness.

5.3 Paraphrase Detection

Data augmentation accounts for over half the total improvement (86.4% to 88.7%), which makes sense given that the cloze formulation imposes an asymmetric left-to-right ordering on an inherently symmetric relation—doubling the data with swapped sentence pairs directly addresses this structural mismatch. The remaining extensions (consistency loss, SMART regularization, domain-adaptive pre-training, LR scheduling, gradient accumulation, text cleaning) collectively add another 2 percentage

Model	Configuration	Dev chrF
GPT-2 small	Baseline (no line markers)	41.70
GPT-2 small	+ Line markers	42.50
GPT-2 medium	+ Scale to medium	43.18
GPT-2 medium	+ Sweep ($\tau=0.85$, $p=0.95$)	43.06
GPT-2 medium	+ Standard reranking ($N=30$)	43.41
GPT-2 medium	+ Line-level reranking ($N=50$)	44.69
<i>Alternative model sizes (with line-level reranking)</i>		
GPT-2 large	Line-level reranking ($N=50$)	43.72
Ensemble (sm+med+lg)	Line-level reranking ($N=50$)	44.39
<i>Vocabulary bias variants</i>		
GPT-2 medium	Standard reranking + vocab bias	43.58
GPT-2 medium	Line-level reranking + vocab bias	44.18

Table 3: Sonnet generation dev chrF scores. The top section is incremental: each row builds on all previous rows. The sweep identified optimal generation parameters ($\tau=0.85$, $p=0.95$) used by all subsequent rows, even though the single-shot score dipped slightly due to seed variance. The bottom sections explore alternatives branching off the best configuration.

Approach	chrF	Why It Did Not Help
Contrastive search	37.81	Designed for factual, deterministic text. Creative generation benefits from the stochasticity of nucleus sampling (42.08 with nucleus).
Repetition penalty ($r = 1.1, 1.2$)	–	Line markers already prevent degenerate repetition loops, so penalizing repeated tokens only hurts fluency.
GPT-2 large + line rerank	43.72	Despite $2\times$ more parameters than medium, produces less diverse candidates; line-level reranking reaches only 43.72 vs. medium’s 44.69.
Ensemble + line rerank	44.39	Pooling candidates from three model sizes yields 44.39, still below single-medium 44.69. Mixing model styles dilutes candidate quality.
Vocab bias + line rerank	44.18	Constraining the vocabulary reduces candidate diversity, giving the greedy line-swap fewer distinct lines (-0.51 vs. no bias).

Table 4: Approaches that did not improve sonnet generation, with analysis of failure modes. ChrF column shows dev scores where available.

points to reach 90.7%. While we did not run a full individual ablation over these, consistency loss and SMART regularization likely carry most of this remaining gain, since both directly regularize the decision boundary rather than simply adjusting optimization dynamics.

Template choice had minimal impact (86.1–86.4% across four templates), suggesting that GPT-2 is not sensitive to the exact phrasing of the cloze prompt as long as the task structure is clear.

5.4 Generation Variance

With 5 random seeds, single-shot GPT-2 medium generation (no reranking) yields a mean chrF of 42.11 ± 0.30 . Our best line-level reranking result of 44.69 exceeds this by over 8 standard deviations, confirming that the improvement is not attributable to seed luck. The relatively low variance also suggests that GPT-2 medium has learned a stable representation of Shakespearean style after fine-tuning, and that most of the remaining variance comes from sampling rather than model instability.

6 Conclusion

We implemented GPT-2 from scratch and fine-tuned it for paraphrase detection and sonnet generation. For paraphrase detection, a combination of domain-adaptive pre-training, SMART regularization, and

Generated (Sonnet 0)	Reference
<i>Thine eyes I love, and they, as pitying me,</i> <i>Knowing thy heart torment me with disdain,</i> <i>Have put on black, and loving mourners be,</i> Where world will travel in madding... That thou died as a tyrants' prisoner. ...	<i>Thine eyes I love, and they, as pitying me,</i> <i>Knowing thy heart torments me with disdain,</i> <i>Have put on black and loving mourners be,</i> Looking with pretty ruth upon my pain. And truly not the morning sun of heaven ...

Table 5: First lines of a generated sonnet versus the Shakespeare original. The first three lines (italicized) are given as the prompt. Generated lines preserve vocabulary and tone but lose semantic coherence.

symmetric consistency losses improved dev accuracy from 86.4% to 90.7%. For sonnet generation, we introduced structural line markers and line-level reranking, which together improved dev chrF from 41.7 to 44.69 over the unmarked baseline.

Our most important finding is that structural encoding matters more than model scale or decoding sophistication for constrained generation. Line markers are a trivially simple modification, yet they enable all of our downstream improvements. Vocabulary bias and contrastive decoding, despite being more complex techniques, provided negligible or negative returns.

The primary limitation of our work is the small training corpus. With only 143 sonnets, the model inevitably memorizes surface patterns rather than learning the deeper structure of Shakespearean verse. Generated sonnets sound plausible line-by-line but often lack coherent thematic development. Future work could explore data augmentation with other Elizabethan poetry, or rhyme-aware loss functions using phonetic dictionaries.

Team Contributions

Mudit Baid: Implemented the base GPT-2 model (attention, transformer layers, embeddings, weight loading) and AdamW optimizer. Built the sentiment classifier with both fine-tuning modes. Developed the cloze-style paraphrase formulation, dataset classes, and text cleaning. Implemented the sonnet training loop, early stopping, and cosine warmup scheduler.

Rakshit Kaushik: Implemented SMART adversarial regularization, consistency loss, domain-adaptive pre-training, data augmentation, seed ensembling, and gradient accumulation for paraphrase detection. Developed nucleus sampling and contrastive search decoding for sonnet generation. Built the hyperparameter sweep infrastructure and evaluation code (paraphrase metrics, ensemble evaluation, multi-prompt evaluation, chrF scoring).

Aalaap Hegde: Designed and implemented structural line markers (training and generation), line-level reranking, vocabulary bias, multi-temperature candidate generation, and coherence filtering for sonnet generation. Implemented dual-perspective inference and learning rate scheduling with warmup for paraphrase detection. Built the evaluation and test output pipelines.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [2] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, pages 255–269, 2021.
- [3] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, 2020.

- [4] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, 2016.
- [5] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.
- [6] Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. A contrastive framework for neural text generation. In *Advances in Neural Information Processing Systems*, 2022.
- [7] Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 168–178, 2017.
- [8] Yi Liao, Yasheng Wang, Qun Liu, and Xin Jiang. GPT-based generation for classical chinese poetry. 2019.
- [9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.
- [10] Maja Popović. chrF: character n-gram f-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.