

Paraphrase Detection and Sonnet Generation using GPT-2

Stanford CS224N Default Project

Dongyu Jia

Department of Computer Science
Stanford University
dongyu@stanford.edu

Omar Ayoub

Department of Computer Science
Stanford University
omaray@stanford.edu

Abstract

This project explores the dual capabilities of the GPT-2 model in both discriminative and generative natural language processing tasks. Building upon a foundational implementation of the GPT-2 architecture and its fine-tuning for sentiment classification, we implement a robust paraphrase detection system using novel feature fusion architectures and data augmentation, and develop an “inference-time intelligence” pipeline for constrained sonnet generation. Our paraphrase detection model leverages bidirectional data augmentation and hidden state aggregation through a compact fusion architecture, achieving over 90.7% dev accuracy on the Quora Question Pairs dataset. We discover that explicit consistency regularizers are redundant when data is symmetrically augmented, and that complex cross-attention fusion introduces significant optimization challenges. For sonnet generation, we implement a line-by-line sampling system with structural filters and a hybrid scoring mechanism. Our experiments demonstrate that while fine-tuning effectively captures stylistic nuances natively, intelligent sampling that prioritizes domain-specific n-gram overlap across un-fine-tuned models can significantly bridge the performance gap, providing a flexible framework for constrained text generation.

1 Key Information to include

- **TA Mentor:** Simon Kim
- **External Collaborators or Mentor:** No
- **Sharing project:** No
- **Late days:** 6 total (3 days per member)

2 Introduction

While large language models like the Generative Pre-trained Transformer 2 (GPT-2) are inherently designed for generative tasks, their rich internal representations can be powerfully adapted for diverse downstream applications. To establish a rigorous baseline for this dual capability, we first complete the core GPT-2 attention blocks and evaluate standard supervised fine-tuning (SFT) techniques on a foundational sentiment analysis task. Building upon these mechanics, we extend the model into two highly complex domains: a robust paraphrase detection system and an intelligent sampling pipeline for sonnet generation. The former focuses on discriminative fine-tuning to identify semantic similarity between sentence pairs, while the latter constrains generative decoding to meet the rigid structural requirements of Shakespearean poetry. In this report, we detail the architectural modifications required for both tasks, compare custom feature fusion strategies, and evaluate how inference-time interventions compare to pure fine-tuning. We demonstrate that for paraphrase detection, bidirectional data augmentation and simple compact feature fusion outperform more complex attention-based networks and explicit consistency constraints. Furthermore, for sonnet generation, we show that

explicitly weighting domain-specific n-gram overlap during decoding can effectively bridge the performance gap for baseline generative models.

3 Related Work

3.1 Paraphrase Detection

Paraphrase detection has transitioned from traditional feature-based methods to deep learning architectures. State-of-the-art results are typically achieved using bidirectional encoders like BERT [1], RoBERTa, and MPNet, which excel at capturing nuanced contextual relationships between sentence pairs. Sentence-BERT (SBERT) [2] further optimized this by producing fixed-size embeddings suitable for efficient cosine similarity comparison.

While GPT models are primarily generative, their pre-trained representations are powerful for downstream classification tasks through specialized heads and feature fusion. Our foundational experiments with fine-tuning for sentiment analysis demonstrated that applying a linear classification head directly to the model’s final token establishes a reliable, albeit naive, approach. While current baselines for more complex tasks like paraphrase detection often continue to rely on these simple linear projections of the final token, we propose a more robust approach in this work by extracting and pooling hidden states from the final four layers of the Transformer, employing both compact concatenation and multi-head attention mechanisms to fuse these features for enhanced performance.

3.2 Sonnet and Poetry Generation

Automated poetry generation requires balancing linguistic creativity with structural constraints (rhyme, meter, and form). Early work like “Deep-speare” [3] utilized deep learning to capture rhythm and rhyme. With the advent of Transformers, GPT-2 [4] has become a standard foundation for creative AI. Recent approaches, such as “The Mechanical Bard” [5], utilize GPT-2 alongside part-of-speech templates or constrained decoding to ensure the output adheres to the 14-line Shakespearean form.

4 Approach

4.1 Sentiment Analysis

To establish a foundational model and classification pipeline, we complete the core GPT-2 architecture - comprising the embedding layer, multi-head self-attention blocks, and multilayer perceptrons - and append a task-specific classification head. Specifically, we extract the contextualized hidden state of the final sequence token, apply dropout for regularization, and project it via a newly initialized linear layer into the requisite number of class logits. The network is optimized using a custom implementation of the AdamW algorithm.

We evaluate this architecture on the SST and CFIMDB datasets under two training paradigms: a “last-linear-layer” approach (freezing base GPT-2 parameters) and a “full-model” approach (end-to-end supervised fine-tuning). Our implementation achieves results consistent with the reference baseline accuracies provided for this task. Beyond sentiment analysis, this simple token-projection method serves as the naive conceptual baseline against which we compare our advanced feature fusion architectures in the subsequent paraphrase detection task.

4.2 Paraphrase Detection

Our approach leverages a pre-trained GPT-2 model with custom feature fusion architectures.

- **Base Architecture:** We use GPT2Model as the backbone, freezing only the embedding layer while fine-tuning all transformer blocks by default.
- **Hidden State Retrieval:** For discriminative capability, we extract the hidden states from the **last four layers** of the GPT-2 model. This is inspired by the feature-based approach in BERT [1], which demonstrated that aggregating the last four layers provides a richer linguistic representation than using only the final layer. Let $H_i \in \mathbb{R}^{L \times D}$ be the hidden state of layer $i \in \{L - 3, L - 2, L - 1, L\}$.

- **Pooling Strategy:** For each of the last four layers, we compute mean and max pooling:
 - $\mu_i = \text{MeanPool}(H_i \cdot \text{Mask})$
 - $m_i = \text{MaxPool}(H_i \text{ with Masked Substitution})$
- **Feature Fusion:** We aggregate these indicators alongside the final token embedding e_{last} .
 - **Attention Fusion:**
 1. We construct a sequence of 9 vectors: $V = [\mu_{L-3\dots L}, m_{L-3\dots L}, e_{last}] \in \mathbb{R}^{9 \times D}$.
 2. We apply a Multi-Head Attention (MHA) block with residual connection and LayerNorm (LN):

$$Z = \text{LN}(\text{MHA}(\text{LN}(V)) + V)$$
 3. The flattened result $f = \text{reshape}(Z) \in \mathbb{R}^{9 \cdot D}$ is passed to the classification head.
 - **Compact Fusion:** A more efficient variant that reduces dimensionality before classification:
 1. Compute "global" aggregates: $\bar{\mu} = \text{mean}(\mu_i)$, $\bar{m} = \text{max}(m_i)$.
 2. Construct the feature vector $f = [\bar{\mu}; \bar{m}; e_{last}] \in \mathbb{R}^{3 \cdot D}$.
- **Classification Head:** The classification logits are computed as:

$$\text{Logits} = W_2 \cdot \text{GELU}(\text{LN}(W_1 \cdot f + b_1)) + b_2$$

- **Consistency Training:** To enhance robustness and ensure symmetry, we employ a symmetric training objective. For a pair of sentences (S_1, S_2) , the model processes both permutations to ensure the representation is invariant to order. **We are inspired by R-Drop [6]**, which uses bidirectional KL divergence to regularize output distributions for consistency training. The total loss is:

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \mathcal{L}_{KL} + \beta \mathcal{L}_{Cosine}$$

- **Symmetric KL Divergence ($\mathcal{L}_{KL}$):** Measures the distributional consistency between the two permutations [6]. Given probability distributions P_1 and P_2 , it is defined as:

$$\mathcal{L}_{KL} = \frac{1}{2} [D_{KL}(\text{Softmax}(P_1) || \text{Softmax}(P_2)) + D_{KL}(\text{Softmax}(P_2) || \text{Softmax}(P_1))]$$
- **Cosine Embedding Loss ($\mathcal{L}_{Cosine}$):** Ensures the aggregated feature vectors f_1, f_2 are geometrically similar in the embedding space, a concept fundamental to Siamese Network architectures for text similarity [2]. It is computed as:

$$\mathcal{L}_{Cosine} = 1 - \cos(\theta) = 1 - \frac{f_1 \cdot f_2}{\|f_1\| \|f_2\|}$$

4.3 Sonnet Generation: Inference-Time Intelligence

The core innovation of our sonnet generation system is the **Inference-Time Intelligence** pipeline. Rather than relying on simple autoregressive decoding, we implement a multi-stage, line-by-line generation process that enforces external structural constraints while optimizing for both fluency and training-set adherence.

- **Fine-tuning:** The GPT-2 model is fine-tuned on a corpus of sonnets to internalize the stylistic and linguistic patterns of the genre.
- **Line-by-Line Sampling:** The model generates the sonnet one line at a time. This allows for intermediate intervention and correction, ensuring that errors in one line do not catastrophically derail the structure of the entire poem.
- **Structural Filtering (Rejection Sampling):** For each line, we generate a large batch of candidates (e.g., $N = 40$). Each candidate is passed through a "Physical Form Filter" that rejects lines failing to meet sonnet criteria:
 - **Length Constraints:** Lines must be between 18 and 65 characters.
 - **Punctuation Rules:** Lines cannot start with punctuation and must use a valid subset of Shakespearean-appropriate punctuation.
 - **Indentation:** Couplets (lines 13-14) must be correctly indented.

- **Terminator Detection:** Candidates must end with a valid newline or EOS token without hitting maximum step limits mid-line.
- **Top-20% Weighted Selection:** From the survivors of the filtering stage, we select the best candidate using a hybrid scoring mechanism:

1. **Scoring:** Each candidate c is scored by a weighted blend of normalized Log-Likelihood (LL) and chrF score (n-gram overlap with the training set):

$$\text{Score}(c) = \omega \cdot Z(\text{LL}_c) + (1 - \omega) \cdot Z(\text{chrF}_c)$$

The inclusion of n-gram overlap (chrF) is critical for sonnet generation as it rewards candidates that closely mirror the Shakespearean training set. We believe that for this task, preserving the specific "Old English" cadence and vocabulary is more important than pure variety, as it ensures the generated text remains within the stylistic bounds of the genre.

2. **Truncation:** We keep only the **top 20%** of candidates based on this score.
3. **Softmax Sampling:** We apply a softmax over the top 20% scores and sample the final line:

$$P(c) = \frac{\exp(\text{Score}(c)/\tau)}{\sum_{j \in \text{Top20\%}} \exp(\text{Score}(j)/\tau)}$$

- **Intra-Sonnet Repetition Penalty:** To ensure thematic progression, we maintain a history of n-grams generated within the current sonnet and penalize candidates with high "self-overlap," preventing the model from becoming trapped in repetitive loops.

5 Experiments

5.1 Sentiment Analysis

To establish our baselines, we evaluated our implementation on two distinct datasets: the Stanford Sentiment Treebank (SST) for 5-way fine-grained sentiment classification, and the CFIMDB dataset for binary sentiment classification. Our primary evaluation metric was classification accuracy, mirroring the reference baseline criteria.

Table 1 details our hyperparameter configurations and final development set accuracies. As expected, full-model fine-tuning significantly outperforms the frozen last-layer approach across both datasets. This aligns with our hypothesis that updating the entire network allows GPT-2’s deep contextual representations to adapt specifically to the linguistic distributions of movie reviews. Furthermore, we observed that the full-model approach was highly susceptible to overfitting; increasing the dropout probability from 0.3 to 0.4 was necessary to stabilize training and achieve the reported dev accuracies.

Dataset	Tune Mode	Batch	LR	Epochs	Drop Prob	Baseline	Accuracy
SST	Last Layer	64	1e-3	15	0.3	0.462	0.465
CFIMDB	Last Layer	64	1e-3	15	0.3	0.861	0.865
SST	Full Model	32	1e-5	10	0.4	0.513	0.516
CFIMDB	Full Model	32	1e-5	10	0.4	0.976	0.980

Table 1: Summary of Sentiment Analysis experimental hyperparameters and results.

5.2 Paraphrase Detection Setup

- **Paraphrase Dataset:** Quora Question Pairs (QQP) dataset.
- **Model Config:** GPT-2 (Small) using BFloat16 precision and Flash Attention for efficiency.
- **Training:** Optimized using AdamW and a OneCycleLR scheduler with a maximum learning rate of $2e-5$.

5.3 Paraphrase Detection Comparison

We conducted three primary experiments for paraphrase detection to evaluate different aspects of our model’s performance.

5.3.1 Experiment 1: Naive Full 12-Layer Fine-Tuning

In this experiment, we tried to use full 12-layer fine-tuning on GPT-2 small with a linear head, to test out if a "data augmentation approach" for pairs (A,B) to (B,A) is helpful. We used 25 epochs, a batch size of 16, and a OneCycleLR learning rate schedule (max lr 5e-5, initial lr 1e-5). As shown in Table 2, the data-augmented experiment generated a higher dev accuracy of 90.6%, compared to the non-augmented baseline which achieved at best 90.1%.

Table 2: Experiment 1: Naive Full 12-Layer Fine-Tuning Results.

Experiment Name	Batch Size	Epochs	Learning Rate	Dev Accuracy
Non-Augmented Baseline	16	25	OneCycleLR (max 5e-5, init 1e-5)	90.1%
Data-Augmented (A,B $\rightarrow$ B,A)	16	25	OneCycleLR (max 5e-5, init 1e-5)	90.6%

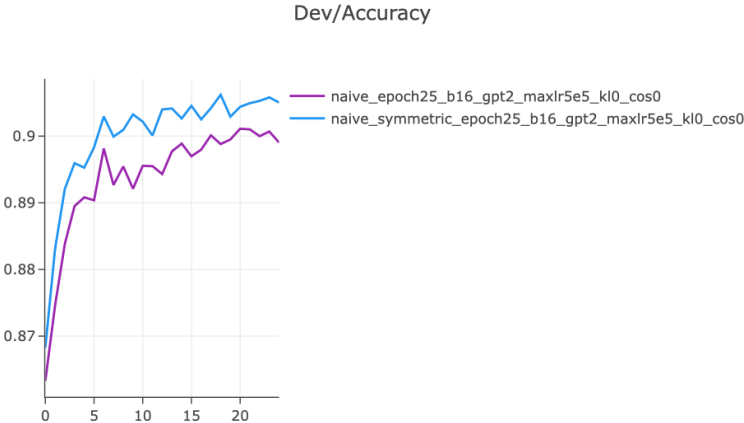


Figure 1: Dev accuracy comparison for Experiment 1.

5.3.2 Experiment 2: Loss Function Comparison

In Experiment 2, we explored the performance under the Compact Fusion setup using different loss function combinations: cross-entropy (CE) loss only versus CE combined with Cosine Embedding loss and KL Divergence loss. For all runs in this experiment, data augmentation was enabled. The models were trained for 25 epochs with a batch size of 16 and a OneCycleLR schedule (max lr 5e-5, initial lr 1e-5).

The auxiliary loss weights α (KL) and β (Cosine) were carefully selected to balance their contributions relative to the primary Cross-Entropy (CE) loss. At the onset of training, we observed that the CE loss was approximately **0.3**, while the Cosine loss was around **0.1**. The KL loss, however, was significantly smaller, starting at approximately **0.01**. To ensure that all components exerted meaningful gradients during the optimization process, we set $\alpha = 10$ and $\beta = 1$, thereby aligning the starting magnitudes of the auxiliary losses with the primary classification objective.

As shown in Table 3, using only CE loss achieved a dev accuracy of 90.65%, whereas the addition of the auxiliary Cosine and KL losses led to a dev accuracy of 90.40%.

Table 3: Experiment 2: Loss Function Comparison Results.

Experiment Name	Batch Size	Epochs	Learning Rate	KL Loss α	Cos Loss β	Dev Acc.
CE Loss Only	16	25	OneCycleLR (max 5e-5, init 1e-5)	0	0	90.65%
CE + KL + Cosine Loss	16	25	OneCycleLR (max 5e-5, init 1e-5)	10	1	90.40%

5.3.3 Experiment 3: Architecture Change

In this experiment, we compare our proposed Compact Fusion and Attention Fusion architectures, testing them with fixed auxiliary loss weights (Cosine Loss = 1, KL Loss = 10). Data augmentation was always enabled for these experiments. As detailed in Table 4, the models were trained with varying batch sizes to accommodate their complexity: the Compact Fusion model used a batch size of 64, while the more memory-intensive Attention Fusion model required a reduced batch size of 32 because our GPU could not fit a batch size of 64 due to the large number of parameters introduced by the fusion mechanism.

Table 4: Paraphrase Detection Comparison.

Model / Mode	Learning Rate	Batch	Epochs	Dev Accuracy	Training Loss(end)	KL Loss	Cos Loss
GPT-2 Compact Fusion	Cosine Decay (2e-5 to 2e-6)	64	50	90.72%	0.082	10	1
GPT-2 Attention Fusion	Cosine Decay (2e-5 to 2e-6)	32	70	90.12%	0.091	10	1

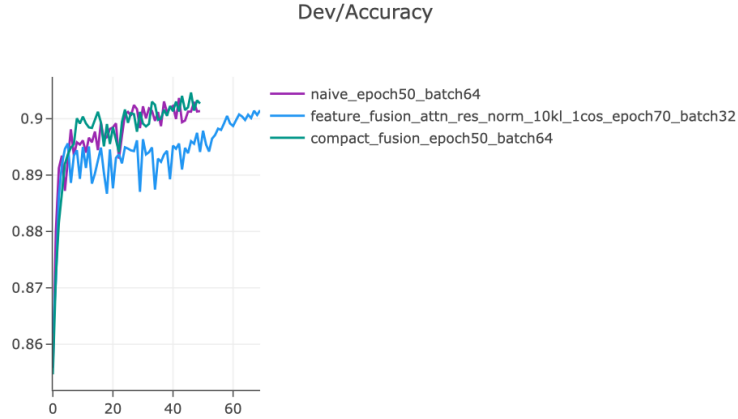


Figure 2: Comparison of development accuracy across different training runs.

5.4 Sonnet Training Dynamics and Baseline Comparison

For the sonnet generation task, we experimented with several model configurations and training parameters, as summarized in Table 5 and illustrated in Figure 3.

Table 5: Comparison of Sonnet Generation Fine-tuning Runs.

Run	Model	Batch Size	Epochs	Learning Rate	Best Dev chrF	Lowest Dev Loss	Lowest Dev Loss Epoch
Run 1	GPT-2 Small	16	100	1e-5	41.785	4.148	22
Run 2	GPT-2 Small	8	20	1e-5	41.724	4.159	12
Run 3	GPT-2 Large	8	25	Cosine Decay (2e-6 to 2e-5)	42.061	3.796	5

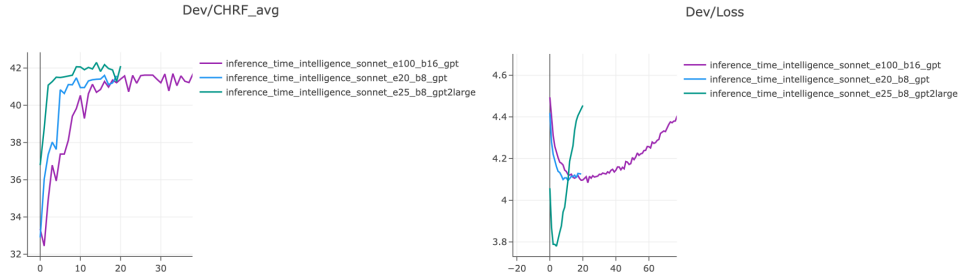


Figure 3: Left: Dev set chrF scores over epochs. Right: Dev set cross-entropy loss over epochs.

5.5 Sonnet Generation: Strategy Selection Results

We evaluated the **Inference-Time Intelligence** pipeline by varying the weight ω between Log-Likelihood (LL) and chrF score. The experiments were conducted with `candidate_num` = 40 and `return_num` = 3.

In Tables 6 and 7, **survival** refers to the number of generated candidates that successfully passed our strict structural filters (proper length, indentation, and punctuation) without being rejected.

To isolate the effects of fine-tuning from the structural constraints of the Inference-Time Intelligence pipeline, we also evaluated the **original, un-fine-tuned GPT** model, which achieved a **Dev Loss: 4.624** and **Dev CHRF avg: 39.580 (std: 1.165)**.

Table 6: Intelligent sampling: Un-fine-tuned GPT.

Strategy (LL_weight / chrF_weight)	Average chrF	Min Survival	Avg Survival
LL 0.0 / chrF 1.0	40.244	4.3	17.9
LL 0.1 / chrF 0.9	40.986	10.9	20.4
LL 0.2 / chrF 0.8	41.523	11.9	23.7
LL 0.3 / chrF 0.7	41.196	13.9	25.0
LL 0.4 / chrF 0.6	40.127	10.3	22.7
LL 0.5 / chrF 0.5	39.751	12.0	25.6
LL 0.6 / chrF 0.4	39.576	9.8	24.1
LL 0.7 / chrF 0.3	39.265	11.0	24.9
LL 0.8 / chrF 0.2	38.678	11.6	24.7
LL 0.9 / chrF 0.1	38.764	14.0	26.5
LL 1.0 / chrF 0.0	38.088	10.2	24.4
Standard top p sampling	39.58	-	-

Table 7: Intelligent sampling: Fine-tuned GPT.

Strategy (LL_weight / chrF_weight)	Average chrF	Min Survival	Avg Survival
LL 0.0 / chrF 1.0	41.292	20.9	37.2
LL 0.1 / chrF 0.9	42.350	22.2	37.2
LL 0.2 / chrF 0.8	41.798	19.5	36.6
LL 0.3 / chrF 0.7	41.364	21.8	37.4
LL 0.4 / chrF 0.6	41.897	26.7	38.2
LL 0.5 / chrF 0.5	41.490	17.8	37.0
LL 0.6 / chrF 0.4	40.938	24.5	37.6
LL 0.7 / chrF 0.3	40.943	18.5	36.8
LL 0.8 / chrF 0.2	41.233	24.9	37.8
LL 0.9 / chrF 0.1	40.450	23.4	37.6
LL 1.0 / chrF 0.0	40.444	24.8	37.9
Standard top p sampling	41.86	-	-

5.6 Leaderboard Scores

Our final models achieved a test accuracy of 0.905 on the Paraphrase detection leaderboard and a 41.968 test chrF score on the Sonnet generation leaderboard.

6 Result Analysis

6.1 Sentiment Analysis

Our experiments revealed distinct optimization requirements between the two training modes. The frozen "last-linear-layer" approach required a higher learning rate (1e-3) and longer training (15

epochs) to reach convergence, easily accommodating a larger batch size (64). Conversely, "full-model" fine-tuning required a drastic reduction in learning rate ($1e-5$) to prevent catastrophic forgetting of pre-trained weights. Unfreezing all parameters also made the model highly prone to overfitting, particularly on the small CFIMDB dataset. To stabilize training, we increased the dropout probability to 0.4 and reduced the batch size to 32. This configuration allowed the full model to converge efficiently in just 10 epochs, effectively balancing capacity and regularization to exceed baseline accuracies on both datasets.

6.2 Paraphrase Detection: Data Augmentation, Loss Functions, and Fusion Architectures

The experimental results across our three experimental setups highlight several key findings regarding data augmentation, auxiliary losses, and representation fusion mechanisms:

Data Augmentation (Experiment 1): Our initial naive fine-tuning revealed that augmenting the dataset with bidirectional context (i.e., generating (B, A) pairs for every (A, B) input) consistently improved generalization. The data-augmented model achieved a higher dev accuracy of 90.6% compared to the non-augmented baseline’s 90.1%. **This confirms that exposing the model to symmetric pair permutations is highly beneficial for paraphrase detection.**

Loss Function Comparison (Experiment 2): When evaluating different loss function combinations under the Compact Fusion setup, we observed that the model trained solely with Cross-Entropy (CE) loss outperformed the model trained with additional auxiliary losses (CE + Cosine + KL), achieving 90.65% dev accuracy versus 90.40%. This provides a critical insight: **because the training data is already symmetric due to our data augmentation strategy, the network implicitly learns symmetric representations.** While KL and Cosine losses explicitly force this symmetry in specific ways, the network likely already achieves its own internal symmetric representation without them. As a result, adding these explicit regularizers is not helpful and may instead act as overly restrictive constraints.

Batch Size Impact (General Observation): Across all our experiments involving varying batch capacities, we highlighted a key trade-off with batch sizes: **while smaller batch sizes can traverse training epochs faster due to more frequent updates per step, larger batch sizes make the optimization steps significantly more stable and consistent over time.**

Fusion Architectures and Convergence (Experiment 3): Comparing the advanced fusion architectures, the Compact Fusion model achieved a higher dev accuracy (90.72%) compared to the more complex Attention Fusion model (90.12%). Furthermore, the Attention Fusion model proved very hard to train. Even after 70 epochs, the training loss remained relatively large (0.091). This suggests that **the complex matrix optimization space introduced by the cross-attention fusion mechanism is likely not smooth and is inherently difficult to optimize effectively compared to simpler pooling mechanisms.**

6.3 Sonnet Generation: Model Selection Strategy and Fine-tuning vs. Intelligent Sampling

Our experiments with the weight distribution between Log-Likelihood (LL) and n-gram overlap (chrF) revealed an interesting phenomenon: **varying the weights did not significantly improve performance for the already fine-tuned model.** While the pipeline ensures that 100% of generated outputs adhere to the sonnet format, the semantic scoring component had limited impact, with scores remaining largely stable between 40.4 and 42.4. We conclude that the fine-tuning phase already effectively "captured" the stylistic distributions, making the external inference-time constraints somewhat redundant. However, across both models, we consistently observed that **higher weights on the chrF score (and correspondingly lower weights on LL) yielded better results.** This occurs because Log-Likelihood fundamentally encourages the model to generate highly probable, safe, and generic text. In contrast, the chrF score explicitly enforces n-gram overlap with the specific target domain (sonnets). The rigid structural and artistic requirements of a sonnet are better captured by forcing this distinct lexical overlap rather than allowing the model to default to generic high-probability English phrasing.

In contrast, the **un-fine-tuned GPT** model, which only achieved a **39.580** average score natively, showed a much more pronounced response to these weight adjustments. With optimal weight selection (LL 0.2 / chrF 0.8), it achieved an average chrF of **41.523**—a score surprisingly competitive

with the fine-tuned results. This finding is significant: it suggests that **intelligent sampling with proper weight selection can bridge the gap between un-refined base models and specialized tuned ones**. While the base model lacks the intrinsic stylistic alignment provided by supervised training, the inference-time constraints can effectively "recover" the desired stylistic and structural form, acting as an online regularizer that compensates for a lack of domain-specific training.

The dynamics of training runs, specifically the balance between development set chrF scores and cross-entropy loss, informed our final model selection.

- **GPT-2 Large Overfitting:** We observed that while GPT-2 Large achieved low cross-entropy loss on dev dataset early on, it often hit its dev loss minimum before reaching its peak chrF score. This behavior suggested that the model began overfitting when it reached high chrF scores, explaining why it was not selected as the final model.
- **Training Duration:** Increasing the number of epochs beyond a certain point (~ 25 epochs) did not yield significant improvements in chrF scores, which stabilized around 41.5. In fact, the development loss tended to increase after this point, signaling the onset of overfitting.
- **Balanced Model Strategy:** To strike an optimal balance between low development loss and high chrF performance, we implemented a strategy where the model was permitted to continue training for 8 additional epochs after the development loss ceased to decrease monotonically. The checkpoint with the highest development chrF score during this window was selected as the final model.

7 Conclusion

This project successfully demonstrates the versatility of GPT-2 across both discriminative and generative applications. For paraphrase detection, we achieved competitive performance (90.72% dev accuracy) by implementing a compact feature fusion architecture paired with powerful bidirectional data augmentation. Crucially, our experiments yielded key insights into regularization: when training data is comprehensively augmented to be symmetric, explicit consistency constraints like KL divergence or cosine embedding loss become largely redundant. We also found that simple, compact fusion structures optimize more efficiently than complex cross-attention mechanisms. Furthermore, our intelligent sampling pipeline for sonnet generation illustrates an effective way to impose rigid structural constraints on autoregressive language models, showing that deliberate inference-time objective weighting (emphasizing domain-specific n-gram overlap over generic log-likelihood) can significantly elevate the structural and stylistic accuracy of un-fine-tuned models.

8 Team Contributions

Dongyu Jia: Conducted various experiments for paraphrase detection (data augmentation, loss functions, architectures) and implemented the intelligent sampling methodology for sonnet generation. Omar Ayoub: Implemented GTP-2 core algorithms and AdamW optimizer. Implemented Sentiment Analysis + ran experiments. Implemented Sonnet Generation using Top-K sampling and Beam Search + ran experiments (we reported on intelligent sampling methodology).

References

- [1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [2] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.
- [3] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia, July 2018. Association for Computational Linguistics.

- [4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.
- [5] Edwin Agnew, Michelle Qiu, Lily Zhu, Sam Wiseman, and Cynthia Rudin. The mechanical bard: An interpretable machine learning approach to Shakespearean sonnet generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1627–1638, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [6] Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Qin Tao, and Chen Wei. R-drop: Regularized dropout for neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.