

Parameter-Efficient Fine-Tuning of GPT-2 with LoRA

Stanford CS224N Default Project

Juniper Jeon

Department of Computer Science
Stanford University
cjeon137@stanford.edu

Erick Ramirez

Department of Computer Science
Stanford University
erick25@stanford.edu

Abstract

Large language models are commonly adapted to downstream tasks through full fine-tuning, which requires updating millions of parameters and significant computational resources. Parameter-efficient methods such as Low-Rank Adaptation (LoRA) aim to reduce this cost by introducing small trainable matrices while keeping the majority of the model weights fixed. In this project, we study how different fine-tuning strategies affect the performance of a GPT-2 model across multiple natural language processing tasks. We implement GPT-2-based models and evaluate three tasks: sentiment analysis, paraphrase detection, and sonnet generation. For sentiment classification, we compare last-layer fine-tuning and full-model fine-tuning under both default and tuned hyperparameter settings. For paraphrase detection and sonnet generation, we further compare full-model training with LoRA-based parameter-efficient adaptation. We evaluate models using accuracy for classification and chrF scores for generation, supplemented with qualitative analysis of model outputs. Our results show that full-model fine-tuning significantly improves sentiment classification compared to training only the final layer, particularly after hyperparameter tuning. For paraphrase detection, the model achieves strong performance with development accuracy reaching 0.899, and LoRA achieves comparable results to full fine-tuning. In the sonnet generation task, hyperparameter tuning substantially improves generation quality, producing more fluent and stylistically consistent outputs, though models still struggle to reliably maintain poetic structure such as rhyme schemes. Our findings suggest that while full-model fine-tuning remains effective, parameter-efficient methods like LoRA can achieve similar performance on both classification and generation tasks while requiring far fewer trainable parameters. These results highlight the potential of LoRA as a practical alternative for adapting large language models to new tasks.

1 Key Information to include

- Mentor: Alisa Levin
- External Collaborators: No
- Sharing project: No
- Late Days: No

2 Introduction

Large language models (LLMs) have achieved strong performance across a wide range of natural language processing tasks, including classification, question answering, and text generation [1]. A common approach for adapting these models to new tasks is *fine-tuning*, in which the pretrained model

parameters are updated using task-specific data. While effective, full-model fine-tuning requires updating hundreds of millions of parameters and can be computationally expensive, particularly when adapting models to multiple tasks or domains.

Recent work has proposed parameter-efficient fine-tuning methods that reduce the number of trainable parameters while preserving model performance. One such approach is Low-Rank Adaptation (LoRA) [2], which inserts small trainable low-rank matrices into selected layers of the model while keeping the original weights frozen. This approach significantly reduces memory and compute requirements during training, making it an attractive alternative to full-model fine-tuning. However, it remains an open question how well such parameter-efficient approaches perform across different types of tasks, particularly when compared with standard fine-tuning strategies.

In this project, we investigate how different fine-tuning strategies affect the performance of a GPT-2-based model across several NLP tasks. Specifically, we evaluate sentiment analysis, paraphrase detection, and sonnet generation. These tasks represent a range of problem types, including sentence-level classification, semantic similarity detection, and open-ended text generation. By evaluating multiple tasks within the same model framework, we aim to better understand how training strategies influence both predictive performance and generation quality.

We compare several approaches for adapting GPT-based models. For sentiment analysis, we examine both last-layer fine-tuning, where only the classification head is trained, and full-model fine-tuning, where all parameters are updated. For paraphrase detection and sonnet generation, we additionally compare full fine-tuning with LoRA-based parameter-efficient training. We evaluate models using standard quantitative metrics such as classification accuracy and chrF scores, and complement these results with qualitative analysis of model outputs.

Our experiments show that full-model fine-tuning substantially improves sentiment classification compared to training only the final layer, particularly after hyperparameter tuning. For paraphrase detection, both full-model and LoRA-based training achieve strong performance with similar development accuracy. In the generative sonnet task, tuning generation parameters significantly improves output coherence, while LoRA produces comparable results to full fine-tuning. Together, these results suggest that parameter-efficient adaptation methods can achieve competitive performance across diverse NLP tasks while requiring far fewer trainable parameters.

3 Related Work

Large language models based on the Transformer architecture have become a central approach for many natural language processing tasks. The Transformer model demonstrated that attention-based architectures can effectively model long-range dependencies in text without relying on recurrence or convolution [3]. Building on this architecture, autoregressive language models such as GPT and GPT-2 showed that large pretrained models can learn general language representations from large corpora and be adapted to downstream tasks through fine-tuning [4]. These pretrained models have since become a common foundation for both classification and generation tasks in NLP.

Pretrained language models are frequently fine-tuned to improve performance on specific downstream tasks. While tuning all weights often yields strong performance, it can be computationally expensive and memory intensive, particularly for larger models that may contain millions or even billions of parameters. Parameter-efficient methods aim to achieve comparable results while training far fewer parameters. One prominent approach is Low-Rank Adaptation (LoRA), which introduces small trainable low-rank matrices into attention layers while keeping the original model weights frozen [2]. LoRA has been shown to achieve performance comparable to full fine-tuning on several NLP benchmarks while significantly reducing the number of trainable parameters.

Other parameter-efficient methods have also been proposed. Adapter-based approaches insert small bottleneck layers into pretrained models and train only those additional parameters [5]. Similarly, prompt tuning and prefix tuning adapt models by learning task-specific prompts or prefixes that guide model behavior without modifying the core model parameters [6]. These approaches demonstrate that effective adaptation of large language models can often be achieved with substantially fewer trainable parameters.

Beyond parameter-efficient methods, prior work has explored the application of pretrained language models to a variety of NLP tasks, including sentiment analysis, paraphrase detection, and text

generation. Fine-tuned transformer models have achieved strong results on sentiment classification datasets such as the Stanford Sentiment Treebank (SST) [7]. Similarly, the Quora Question Pairs dataset has become a widely used benchmark for evaluating semantic similarity and paraphrase detection models [8]. For generative tasks, autoregressive models such as GPT-2 have demonstrated the ability to produce coherent long-form text when conditioned on prompts [4].

Recent surveys of large language models highlight both the versatility of these models and the growing interest in efficient adaptation techniques [1]. However, many studies evaluate parameter-efficient methods only on one application or focus primarily on either classification or generation. In this work, we evaluate fine-tuning strategies across multiple applications, including classification, semantic similarity, and text generation, using a consistent GPT-based framework. This allows us to compare the behavior of full fine-tuning and parameter-efficient methods such as LoRA across different types of NLP problems.

4 Approach

We adapt a GPT-based transformer model to three tasks: sentiment classification, paraphrase detection, and sonnet generation. Our experiments compare several training strategies, including full-model fine-tuning, head-only fine-tuning, hyperparameter tuning, and parameter-efficient fine-tuning using LoRA.

Sentiment classification. For sentiment analysis, we use GPT as a sequence encoder and attach a linear classification head to the final token representation. Given an input sequence $x = (x_1, \dots, x_T)$, the transformer produces contextual representations $h_1, \dots, h_T$, and the final token representation h_T is used for classification:

$$\hat{y} = \text{softmax}(Wh_T + b),$$

where $W \in \mathbb{R}^{2 \times d}$ maps the hidden dimension d to the two sentiment classes. We evaluate two training strategies: (1) *last-linear-layer* fine-tuning, where the GPT parameters are frozen and only the classification head is trained, and (2) *full-model* fine-tuning, where all parameters are updated. In addition to the default configuration provided in the CS224N starter code [9], we also evaluate a tuned configuration with modified regularization and learning rate.

Paraphrase detection. For paraphrase detection, we use the same GPT encoder and classification architecture. The model receives a pair of questions concatenated into a single input sequence and predicts whether the two questions are paraphrases. As with sentiment classification, the final token representation is passed through a linear classifier to produce logits for the two output classes.

To study parameter-efficient adaptation, we additionally implement Low-Rank Adaptation (LoRA) [2]. In LoRA, updates to a weight matrix W are parameterized as a low-rank decomposition

$$\Delta W = BA,$$

where $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$ with $r \ll d$. During training, the original GPT weights remain frozen and only the low-rank matrices are updated, reducing the number of trainable parameters while preserving most of the model capacity.

Sonnet generation. For text generation, GPT is trained as an autoregressive language model that predicts the next token given the previous context. For a sequence $x_1, \dots, x_T$, the model predicts

$$p(x_{t+1} \mid x_{\leq t}) = \text{softmax}(Wh_t),$$

where h_t is the hidden representation at position t . Unlike the classification tasks, logits are produced for every position so the model learns the distribution over full sonnet sequences.

We experiment with both the default GPT configuration and a tuned configuration with a larger model and modified sampling parameters. Generation is performed using temperature scaling and nucleus sampling, and we also evaluate a LoRA-based parameter-efficient variant of the model to compare against standard fine-tuning.

5 Experiments

5.1 Data

For sentiment analysis, we use two datasets. The first is the Stanford Sentiment Treebank (SST) [7], which contains 11,855 movie review sentences annotated with fine-grained sentiment labels (*negative*, *somewhat negative*, *neutral*, *somewhat positive*, and *positive*). The project provides the following splits: 8,544 training examples, 1,101 development examples, and 2,210 test examples. In our experiments, the task is sentence-level sentiment classification, where the model predicts the fine-grained sentiment label of a given input sentence. We also use the CFIMDB dataset [9], which consists of 2,434 longer movie reviews labeled as either positive or negative. The project provides the following splits: 1,701 training examples, 245 development examples, and 488 test examples. This task is document-level binary sentiment classification, where the input is a movie review and the output is a binary sentiment label.

For paraphrase detection, we use the Quora Question Pairs dataset [8], which contains pairs of questions labeled according to whether they are paraphrases of one another. The dataset contains approximately 400,000 question pairs in total, and the project provides the following splits: 141,506 training examples, 20,215 development examples, and 40,431 test examples [9]. The task is binary classification: given two questions (s_1, s_2), the model predicts whether they have the same meaning. Following the project setup, each example is formatted as a natural language prompt of the form: “Is ‘ s_1 ’ a paraphrase of ‘ s_2 ’? Answer ‘yes’ or ‘no’: ”, and the model predicts the next token corresponding to the label.

For generation, we use the Shakespeare Sonnet Dataset [9]. This dataset consists of 154 Shakespearean sonnets, each containing 14 lines that follow the rhyme scheme ABAB CDCD EFEF GG. The provided splits include 143 training sonnets and 12 test sonnets.

5.2 Evaluation Method

We evaluate our model on three tasks: sentiment analysis, paraphrase detection, and sonnet generation.

For the sentiment analysis and paraphrase detection tasks, we report development set accuracy. In each experiment, the model is trained on the training split and evaluated on the development split. Accuracy is defined as

$$\text{Accuracy} = \frac{\# \text{ correct predictions}}{\# \text{ total examples}}.$$

For sentiment classification, the model predicts the sentiment label of a sentence or review (e.g., positive or negative). For paraphrase detection, the model predicts whether a pair of questions are paraphrases of each other. These tasks are standard classification problems, making accuracy an appropriate evaluation metric.

For the text generation task, the model is fine-tuned using cross-entropy loss on the Shakespeare sonnet training set. During evaluation, the model is given the first three lines of each held-out sonnet and is required to generate the remaining lines of the poem. Generated sonnets are evaluated using the chrF metric [10], a character-level n -gram F-score that measures similarity between the generated text and the reference sonnet. This metric is commonly used for evaluating generative language models because it captures character-level similarity and is less sensitive to small variations in wording than word-level metrics such as BLEU [11].

5.3 Experimental details

Hyperparameters and model configurations for sentiment analysis, paraphrase detection, and sonnet generation are summarized in Tables 1, 2, and 3. Unless otherwise noted, we use the default settings from the starter code [9]. All models were trained using the AdamW optimizer on a G4 GPU.

5.4 Results

Sentiment Classification. Table 4 summarizes development and test accuracy for sentiment analysis under different fine-tuning settings. *Full-model* fine-tuning consistently outperforms **last-linear-layer** training, particularly on the CFIMDB dataset. With the default configuration, the full-model achieves

Hyperparameter	Default	Tuned (Full Model)
Epochs	10	10
Batch size (SST)	64	64
Batch size (CFIMDB)	8	8
Hidden dropout	0.3	0.73
Learning rate	1×10^{-3}	1×10^{-5}
Weight decay	0	0.1

Table 1: Hyperparameters used for sentiment analysis experiments.

Hyperparameter	Value
Epochs	10
Batch size	8
Learning rate	1×10^{-5}
Hidden size (d)	768

Table 2: Hyperparameters used for paraphrase detection experiments.

Hyperparameter / Component	Default	Tuned
Model	GPT-2	GPT-2 Large
Hidden size (d)	768	1280
Number of layers (l)	12	36
Attention heads	12	20
Epochs	10	10
Batch size	8	8
Learning rate	1×10^{-5}	1×10^{-5}
Dropout	–	0.1
Temperature (generation)	1.2	0.6
Top- p (nucleus sampling)	0.9	0.9
Top- k filtering	–	40
Trainable parameters	Generation head only	Word embeddings + last 2 GPT layers
Max generation length	128	128

Table 3: Hyperparameters used for sonnet generation experiments.

0.910 development accuracy on CFIMDB compared to 0.837 when only the classification head is trained. After hyperparameter tuning, full-model performance further improves to 0.980 development accuracy. In contrast, tuning substantially degrades the last-linear-layer results. The high dropout (0.73) and weight decay (0.1) appear to over-regularize the small classification head, leading to underfitting.

Performance on SST remains relatively low across all configurations, with development accuracy near 0.5. This suggests the model struggles to capture sentence-level sentiment in shorter inputs, possibly due to insufficient training or the difficulty of adapting GPT representations to this dataset.

Setting	SST Dev Acc	SST Test Acc	CFIMDB Dev Acc
Last-linear-layer (default)	0.461	0.470	0.837
Full-model (default)	0.418	–	0.910
Last-linear-layer (tuned)	0.337	–	0.551
Full-model (tuned)	0.507	0.527	0.980

Table 4: Development and test accuracy on SST and CFIMDB under different fine-tuning settings.

Paraphrase Detection. Tables 5 and 6 show training loss and development accuracy for paraphrase detection with and without LoRA. Without LoRA, the model improves rapidly during early epochs and plateaus after roughly six epochs. Training loss continues to decrease slightly after this point while development accuracy decreases, indicating mild overfitting.

The LoRA model achieves similar development performance to the fully fine-tuned model. The best development accuracy without LoRA is 0.899, compared to 0.896 with LoRA. Test accuracy is also comparable (0.875 vs. 0.873). These results suggest that LoRA can match the performance of full fine-tuning while updating far fewer parameters under a reduced training time.

Epoch	Train Loss	Dev Acc	Test Acc	Epoch	Train Loss	Dev Acc	Test Acc
0	0.360	0.872	—	0	0.359	0.869	—
1	0.273	0.884	—	1	0.271	0.876	—
2	0.223	0.892	—	2	0.223	0.885	—
3	0.183	0.894	—	3	0.182	0.893	—
4	0.149	0.895	—	4	0.148	0.890	—
5	0.119	0.895	—	5	0.120	0.895	—
6	0.096	0.899	0.875	6	0.097	0.896	—
7	0.080	0.894	—	7	0.080	0.894	—
8	0.067	0.897	—	8	0.066	0.896	0.873
9	0.056	0.897	—	9	0.057	0.896	—

Table 5: Training loss and development accuracy across epochs on paraphrase detection without LoRA. Table 6: Training loss and development accuracy across epochs on paraphrase detection with LoRA.

Figure 1 visualizes the training dynamics for the two paraphrase models. Both models exhibit nearly identical loss and accuracy curves, further supporting that LoRA provides comparable performance to full fine-tuning in this task.

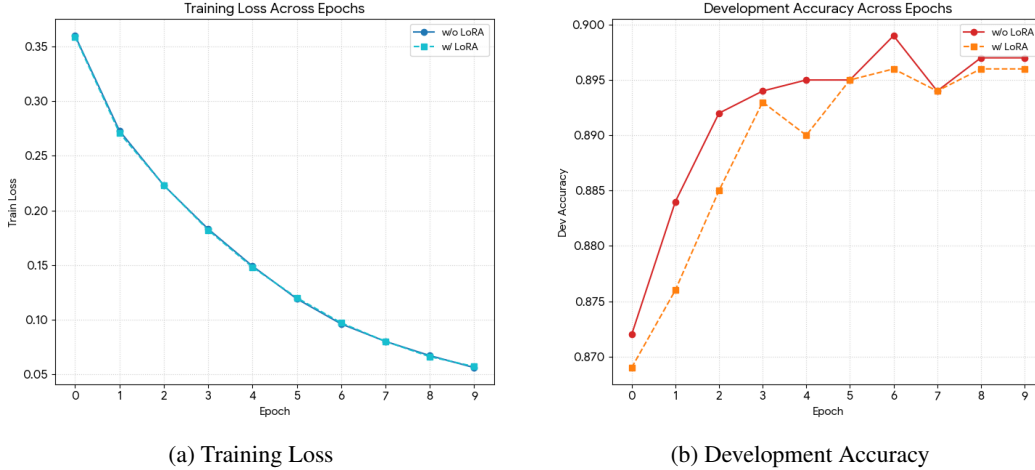


Figure 1: Training loss and development accuracy across epochs on paraphrase detection with and without LoRA.

Sonnet Generation. Table 7 reports chrF scores for sonnet generation under different configurations. Hyperparameter tuning and a larger model substantially improve generation quality, increasing the development chrF score from 37.3 to 41.1. The tuned model also achieves the highest test score of 41.247. In contrast, LoRA slightly reduces performance in both the default and tuned configurations, though the difference remains relatively small. This resembles the reduced but comparable performance of LoRA seen in the paraphrase detection classification task. These results indicate that LoRA is effective for classification and generative tasks.

Parameter Settings	chrF Dev Score	chrF Test Score
Default w/o LoRA	37.273	—
Default w/ LoRA	36.852	—
Tuned w/o LoRA	41.068	41.247
Tuned w/ LoRA	40.039	39.805

Table 7: chrF scores for sonnet generation with and without LoRA under default and tuned configurations.

6 Analysis

Paraphrase Detection. We performed qualitative inspection on the paraphrase detection model by comparing predictions with and without LoRA for selected examples. The model occasionally flips its prediction depending on whether LoRA is applied, even when the semantic similarity is clear.

Example 1:

Sentence 1: How long can you leave cooked chicken wings out at room temp?

Sentence 2: Can you leave raw chicken out all night?

Ground Truth: 0 (not a paraphrase)

Prediction with LoRA: No

Prediction without LoRA: Yes

Example 2:

Sentence 1: Is it possible to see who has viewed my profile page?

Sentence 2: Can you see who viewed your profile on Quora?

Ground Truth: 0 (not a paraphrase)

Prediction with LoRA: Yes

Prediction without LoRA: No

These examples illustrate that while LoRA generally maintains competitive development accuracy, it can introduce inconsistencies on individual examples. This highlights a trade-off between parameter-efficient fine-tuning and stability on edge cases, and suggests that LoRA may sometimes overfit subtle patterns in the training data.

Sonnet Generation. We qualitatively inspected generated sonnets under four configurations: default with LoRA, default without LoRA, tuned with LoRA, and tuned without LoRA. The default models frequently produce incoherent text after the initial prompt lines, often degenerating into unrelated tokens or formatting artifacts. This behavior appears in both LoRA and non-LoRA settings, suggesting that generation quality is more strongly influenced by the model configuration and hyperparameters than by the use of LoRA itself.

Default configuration with LoRA:

Beshrew that heart that makes my heart to groan
For that deep wound it gives my friend and me.
Is 't not enough to torture me alone,
Exec Davlaughter guardeduti levelICES adul Jennifer Dominic...

Default configuration without LoRA:

Beshrew that heart that makes my heart to groan
For that deep wound it gives my friend and me.
Is 't not enough to torture me alone,
gresIPS Sanford Pad Avery jud391 Postedabol IncomeTuesdayphi...

In contrast, the tuned configurations produce substantially more coherent and stylistically consistent outputs. The generated text remains grammatical and maintains a poetic tone for longer sequences, though the model still struggles to consistently enforce Shakespearean rhyme schemes or meter.

Tuned configuration with LoRA:

Beshrew that heart that makes my heart to groan
For that deep wound it gives my friend and me.
Is 't not enough to torture me alone,
And make me cry out in my sleep?
Thy love is more than I can bear
To lose her for evermore...

Tuned configuration without LoRA:

Beshrew that heart that makes my heart to groan
For that deep wound it gives my friend and me.

Is 't not enough to torture me alone,
And make me bleed for a few hours?
I am so far from home, I have no one.
My mother is gone, and my father is dead...

Overall, tuning substantially improves fluency and stability of the generated text. The LoRA and full fine-tuning variants produce broadly similar qualitative outputs in the tuned setting, though the LoRA model occasionally generates shorter or less complete sequences. This aligns with the relatively small differences observed in chrF scores between the two approaches.

7 Conclusion

In this project, we explored fine-tuning GPT-based models for three NLP tasks: sentiment analysis, paraphrase detection, and sonnet generation. We compared multiple training strategies, including last-layer fine-tuning, full-model fine-tuning, and parameter-efficient methods using LoRA. Across these experiments, we evaluated both default and tuned hyperparameter configurations to understand how model capacity and training choices affect performance.

Our results show that full-model fine-tuning performs substantially better than training only the final classification layer for sentiment analysis, particularly after hyperparameter tuning. While the last-layer approach underfit the task, tuning the full model improved SST performance and achieved strong results on the CFIMDB dataset. For paraphrase detection, the model achieved high development accuracy and converged quickly within the first several epochs. LoRA achieved similar performance to full fine-tuning, suggesting that parameter-efficient adaptation can maintain competitive results while reducing the number of trainable parameters.

For sonnet generation, hyperparameter tuning had the largest impact on output quality. Default configurations often produced incoherent text after the initial prompt, whereas tuned models generated substantially more fluent and stylistically consistent lines. Qualitative inspection confirmed that the tuned models better maintained poetic tone, though they still struggled to reliably enforce Shakespearean rhyme schemes and meter. LoRA again produced comparable performance to full fine-tuning, with only minor differences in generation quality.

Overall, this project demonstrated how training strategies and hyperparameter choices influence both quantitative metrics and qualitative behavior in language models. We also observed that parameter-efficient methods such as LoRA can achieve similar performance to full fine-tuning on multiple tasks. The primary limitations of our work include limited hyperparameter exploration and evaluation primarily through aggregate metrics such as accuracy and chrF. Future work could explore more systematic tuning, larger model variants, and more detailed analysis of generated text structure, such as enforcing rhyme schemes or meter during generation.

Team contributions

This project was conducted by Juniper Jeon and Erick Ramirez.

Juniper completed the baseline module files, the optimizer, and the forward functions for the downstream tasks. Collaboratively worked on the final report.

Erick implemented classifier, LoRA, collaboratively finetuned models, and ran the final experiments for sentiment classification, paraphrase detection, and sonnet generation. Collaboratively worked on the final report.

References

- [1] Eman Alomari. Unlocking the potential: a comprehensive systematic review of chatgpt in natural language processing tasks. *Computer Modeling in Engineering & Sciences*, 141(1):43, 2024.
- [2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

- [3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- [4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.
- [5] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mohammad Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, 2019.
- [6] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics*, 2021.
- [7] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, 2013.
- [8] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First quora dataset release: Question pairs. <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>, 2017. Quora Dataset Release.
- [9] Stanford CS224n Course Staff. Cs224n default final project instructions. https://web.stanford.edu/class/cs224n/project/DFP_Instructions.pdf, 2023. Accessed: 2026-02-28.
- [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, Lisbon, Portugal, 2015. Association for Computational Linguistics.
- [11] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191. Association for Computational Linguistics, 2018.